

Semantic Modeling Methodology

Towards interoperability of systems

Andries van Renssen

Edition 2

Copyright © 2015 Dr. Ir. Andries van Renssen.

Gellish.net, Zoetermeer, The Netherlands

www.gellish.net

All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the Author.

ISBN 978-1-329-65448-8

Table of content

1	Introduction.....	10
2	Semantic information modeling.....	14
2.1	What are semantic information models?.....	14
2.2	Models of individual things versus models of kinds.....	18
2.3	The purpose of semantic information models.....	22
2.4	Kinds of semantic information models.....	23
2.5	Knowledge Libraries.....	27
2.6	Benefits of integrated models.....	28
3	Formalized natural languages.....	31
3.1	Formal language defining ontologies.....	31
3.1.1	Vocabulary.....	31
3.1.2	Dictionary.....	32
3.1.3	Taxonomy-ontology.....	33
3.1.4	Relation types.....	35
3.2	Correct formal language expressions.....	37
3.3	Unique identification and naming.....	40
3.3.1	Numbers.....	46
3.3.2	Date-times.....	47
4	Architecture of semantic information models.....	49
4.1	Outline of an Information Model Architecture.....	49
4.2	Phases in the information modeling process.....	54
4.3	Definition models.....	55
4.3.1	The taxonomic dictionary-ontology.....	55
4.3.2	Terminology management.....	61
4.3.3	Synonyms, homonyms and unique identifiers.....	61
4.3.4	Multiple languages.....	62
4.3.5	Domain dictionaries and dictionary extensions.....	63
4.3.6	Definition models.....	65
4.4	Knowledge models.....	68
4.4.1	The expression of knowledge.....	68
4.4.2	The general structure of knowledge models.....	73
4.4.3	Composition trees.....	74
4.5	Requirements models.....	75
4.6	Product type models.....	80
4.7	Document models and 3D models.....	83
4.8	Product and process information models.....	84
4.8.1	Modeling of information about individual things.....	85
4.9	Relations between models.....	88
4.10	Usage of modeled knowledge and requirements.....	90
4.10.1	Knowledge and requirements consultation.....	90
4.10.2	Example usage by a designer.....	91
4.10.3	Example usage by a cost estimator.....	91

5	Product Information Modeling.....	93
5.1	The product model creation process.....	93
5.2	Step 1, Create product composition hierarchies.....	96
5.3	Step 2, Create document models.....	102
5.3.1	Step 2.1 - Specify meta-data about the information.....	103
5.3.2	Step 2.2 - Relate documents to product components.....	109
5.3.3	Step 2.3 - Relate documents to files at addresses.....	109
5.3.4	Step 2.4 - Compose documents.....	111
5.4	Step 3, Specify activities and processes (functions).....	112
5.5	Step 4, Specify aspects of products and components.....	115
5.5.1	Properties and qualities.....	115
5.5.2	States and phases.....	117
5.5.3	Structure - Process material & radiation.....	118
5.5.4	Material of construction.....	120
5.5.5	Batches & streams and their states and properties.....	121
5.5.6	Concentration of elements in mixtures.....	123
5.6	Step 5, Relate fabricated items to designed items.....	125
5.7	Modeling of physical objects.....	126
5.7.1	Facilities and systems and their roles.....	126
5.7.2	Equipment, machines, components and spare parts.....	128
5.7.3	Networks and piping systems.....	132
5.7.4	Electrical.....	143
5.7.5	Instrumentation & Control systems.....	143
5.7.6	Infrastructure and Buildings.....	144
5.7.7	Waves and energy.....	144
5.7.8	Signals.....	146
5.7.9	Holes and spaces.....	149
5.7.10	Features.....	150
5.8	Verification rules for correctness of product models.....	151
5.9	Summary of relation types for product models.....	154
6	Application system usage examples.....	158
6.1	The Gellish Communicator.....	158
6.2	Functionality.....	159
6.2.1	Import, combination and verification of information.....	160
6.2.2	Formulating queries with conditions.....	161
6.2.3	Logic conclusions and answers.....	164
6.2.4	Queries for product and process models.....	165
6.2.5	Views on product models.....	168
6.2.6	Exchange of information.....	173
6.2.7	Editing of information models.....	175
6.2.8	Knowledge and requirements based proposals and verification.....	177
6.3	The Gellish Search Engine.....	180
7	Process information modeling.....	187
7.1	Modeling of activities, processes and events.....	188
7.2	Sequences of activities.....	189

7.3	Graphical representation of activity sequences.....	190
7.4	Physical objects involved in occurrences.....	192
7.4.1	Kinds of involvements.....	192
7.4.2	Properties of physical objects.....	194
7.4.3	Physical objects subjected to activities.....	194
7.5	Sequences of activities in IDEF0.....	197
7.6	Duration, begin and end, where and when.....	199
7.7	Cause and effect.....	200
7.8	States and state changes.....	201
7.9	Process designs and control.....	204
7.9.1	Processes.....	204
7.9.2	Batch processes & state transitions.....	205
7.9.3	Process cases.....	207
7.9.4	Energy transport and conversion processes.....	207
7.9.5	Process Design & Process Simulation.....	207
7.9.6	Process Control.....	209
7.9.7	Control logic and function blocks.....	212
7.10	Fabrication and assembly.....	214
7.11	Construction & commissioning.....	215
7.11.1	Commissioning and start-up activity definitions.....	216
7.11.2	Relations between activities.....	217
7.11.3	Duration and resources.....	218
7.11.4	Special tools, temporary facilities and consumables.....	218
7.11.5	Procedures to execute activities.....	219
7.12	Maintenance.....	219
7.13	Scope and lay-out of diagrams of activities.....	220
7.13.1	Interacting with models via schematic diagrams.....	221
8	Observations and measurements modeling.....	222
8.1	Measured properties.....	222
8.2	Observing activities and observations.....	222
8.2.1	What is an observation and a measurement?.....	222
8.2.2	Modeling of observations and measurements.....	223
8.2.3	An observation device.....	227
8.3	Recording of textual or pictorial results.....	229
8.3.1	Collections of formal language expressions.....	230
8.3.2	Information on information carriers.....	230
8.3.3	Relations between resulting information and described objects.....	231
8.4	Measurement of location dependent properties.....	232
9	Integration of 1D, 2D and 3D product models.....	235
9.1	Levels of information integration.....	235
9.2	Characteristics of 1D, 2D and 3D product models.....	239
9.3	Coordinate systems specification.....	240
9.3.1	Cartesian coordinate systems.....	242
9.3.2	Cylindrical coordinate systems.....	245
9.3.3	Spherical coordinate systems.....	246

9.3.4	Geographical coordinate systems.....	247
9.3.5	Standard individual coordinate systems.....	249
9.3.6	Positioning in 3D coordinate systems.....	250
9.4	Shape aspects of Physical Objects.....	254
9.4.1	Shapes.....	254
9.4.2	Kinds of shapes.....	255
9.4.3	Representation of shapes - Geometric Models.....	256
9.4.4	One dimensional shape aspects (1D).....	257
9.4.5	Geometric models of 3D shapes.....	258
9.4.6	Specification of components.....	264
9.4.7	Specification of components with standard 3D shapes.....	265
9.4.8	Positioning of component shapes in a 3D coordinate system.....	269
9.5	Drawings and views.....	270
9.5.1	Kinds of drawings.....	270
9.5.2	Graphical elements.....	273
9.5.3	Schematic graphical elements.....	277
9.5.4	Positioning of graphical elements in 2D.....	279
9.5.5	Assemblies of graphical elements.....	286
9.5.6	Standard or typical graphical elements.....	287
9.5.7	Annotation elements on drawings.....	289
9.5.8	Schematic diagrams.....	291
10	Modeling of definitions.....	318
10.1	Definitions and definition models.....	318
10.2	Definitions of related concepts.....	319
10.3	The structure of a formal taxonomic dictionary.....	320
10.4	Domains of concepts (kinds).....	322
10.4.1	Collections of kinds.....	325
10.4.2	Standardized product types and catalogue items.....	327
10.4.3	Kinds of relations.....	328
10.4.4	Composition and connection relations between members of kinds...	330
10.4.5	Kinds of physical objects.....	332
10.4.6	Kinds of roles.....	337
10.4.7	Identification of new kinds and their hierarchy.....	339
10.4.8	Kinds of properties, qualities and encoding aspects.....	340
10.4.9	Kinds of aspects of physical objects.....	341
10.4.10	Kinds of substances - process materials & materials of construction	343
10.4.11	Kinds of signals, waves and energy.....	344
10.4.12	Kinds of occurrences.....	344
10.4.13	Kinds of organisms, biological & biochemical objects.....	346
10.4.14	Templates and product models (forms and data sheets).....	346
10.4.15	Scales and currencies.....	347
10.4.16	Qualitative concepts.....	347
10.4.17	Standard individual things.....	348
10.5	Formal dictionary development.....	348
10.6	Patterns for definition models.....	350

10.6.1	Pattern for the definition of physical objects.....	350
10.7	Definition model development procedure.....	352
10.7.1	Stage 1: Allocate a UID.....	354
10.7.2	Stage 2: Specify names, synonyms and homonyms.....	355
10.7.3	Stage 3: Specify a direct supertype.....	362
10.7.4	Stage 4: Add a textual description.....	368
10.7.5	Stage 5: Specify defining components.....	370
10.7.6	Stage 6: Specify defining intrinsic aspects.....	374
10.7.7	Stage 7: Define the nature of intrinsic aspects.....	378
10.7.8	Stage 8: Define values of intrinsic aspects.....	379
10.7.9	Stage 9, Specify functions and define occurrences.....	382
10.7.10	Stage 10: Geometric representation of objects.....	385
10.7.11	Stage 11: Other kinds of relations.....	386
10.7.12	Stage 12: Add contextual facts.....	386
10.7.13	Stage 13: Information about a concept.....	386
10.7.14	Stage 14: Documenting definition models in a standard format.....	388
10.8	Definitions through usage of templates.....	388
10.9	Implementation of taxonomic dictionaries.....	391
10.10	Quality rules for taxonomic dictionaries.....	391
10.11	Summary of relation types for definition models.....	395
11	Knowledge modeling.....	399
11.1	Introduction to knowledge modeling.....	399
11.2	Expression of knowledge.....	400
11.2.1	Role of objects in relations.....	402
11.2.2	Cardinalities.....	403
11.3	Contexts.....	404
11.4	The knowledge models creation process.....	405
11.4.1	Stage 1: Specify possible compositions.....	405
11.4.2	Stage 2: Specify possible aspects.....	406
11.4.3	Stage 3: Specify knowledge about functions and occurrences.....	407
11.4.4	Stage 4: Specify other possible relations.....	408
11.5	Usage of knowledge for creation of design proposals.....	409
11.6	Knowledge-aided design proposals.....	411
12	Requirements modeling.....	414
12.1	Objectives and scope of requirements models.....	414
12.2	Inheritance of requirements.....	416
12.3	Context dependent requirements.....	417
12.4	Project requirements.....	419
12.5	Implementation of requirements models.....	420
12.6	Verification of delivered information.....	420
12.7	Conditional requirements.....	420
12.8	Integration of requirements models in Information Models.....	423
12.9	Requirements model creation procedure.....	423
12.9.1	Step 1, Definition of a subtype concept.....	424
12.9.2	Step 2, Create a collection of requirements.....	425

12.9.3	Step 3: Required components of an assembly.....	426
12.9.4	Step 4, Required kinds of aspects.....	430
12.9.5	Requirements for multiple relations of the same kind.....	434
12.9.6	Requirements for units of measure.....	434
12.9.7	Step 5, Allowed values for required aspects.....	435
12.9.8	Step 6, Textual requirements and rules.....	435
12.9.9	Relate documents to files and addresses.....	438
12.9.10	Detailed modeling of textual requirements and rules.....	439
12.9.11	Step 7, Specify requirements about kinds of documents.....	442
12.9.12	Step 8: Requirements for kinds of occurrences.....	445
12.10	Verification of product models against requirements.....	445
12.11	Summary of relation types for modeling of requirements.....	447
13	Modeling of product types.....	451
13.1	Product libraries and electronic product catalogues.....	451
13.2	Product type information models.....	452
13.3	Categories of product types.....	453
13.3.1	Product families.....	454
13.4	Procedure for creating object libraries & product catalogues.....	456
13.4.1	Step 1: Streamlining object libraries & product catalogues.....	456
13.4.2	Step 2: Determine characterizing aspects and components.....	458
13.4.3	Step 3: Define kinds of components, aspects and allowed values....	458
13.4.4	Step 4: Create product family requirements models.....	459
13.4.5	Step 5: Define product types.....	461
13.4.6	Step 6: Specify values for the definition of product types.....	462
13.4.7	Step 7: Create definition models for product types.....	463
13.4.8	Step 8: Specify scales and units of measure.....	467
13.5	Usage of product types for classifying individual items.....	468
13.6	Summary of relation types.....	468
14	Modeling principles and guidelines.....	470
14.1	Classification and specialization guidelines.....	470
14.2	Assembly, compositions and collections.....	472
14.3	Aspects of parts with product decomposition.....	473
14.4	Aspects of parts without product decomposition.....	474
14.5	Subtypes of aspects and roles of aspects.....	476
14.6	Properties of properties (not allowed).....	477
14.7	Collections of aspects.....	478
14.8	Correlations between aspects.....	478
14.9	Dictionaries and taxonomies versus knowledge models.....	479
14.10	Functions and locations of physical objects in assemblies.....	481
14.11	Type of thing: Requirements to classify things.....	482
14.12	How to add relation types.....	483
14.13	Information about a thing versus a thing itself.....	485
14.14	Roles of aspects and roles of physical objects.....	485
14.15	Individual things versus kinds of things.....	486
14.16	Imaginary objects versus installed, materialized objects.....	487

14.17	Functions as roles of activities.....	490
14.18	Product specification versus process specification.....	491
14.19	Life cycle approach of products.....	493
14.20	Modeling of information from source documents.....	494
14.21	Roles of objects (classification by role and by nature).....	497
14.22	Inheritance of facts.....	498
14.23	Quantifiable versus qualifiable and literal properties.....	499
14.24	Scales and UoM's.....	500
14.25	Synonyms versus preferred terms.....	501
14.26	Generic concepts, types and manufacturer's models.....	502
14.27	Criterion to define a new class or specialization.....	504
14.28	Connections and ports.....	504
14.29	Collection and assembly.....	506
14.30	Parts of more than one whole.....	507
14.31	Inverse relations.....	508
14.32	Cardinality constraints.....	508
14.33	Homonyms.....	508
14.34	Kinds versus characteristics.....	508
14.35	Characteristics required for activities.....	509
14.36	Requirements for detailed kinds of Systems.....	510
15	Certification.....	511
16	Managing the formal language definition.....	512
16.1	Rules for acceptance of proposals for extensions.....	512
16.2	Return proposals to the language manager.....	514
16.3	Version management of Gellish languages.....	514
17	References.....	515
18	Glossary.....	517

1 Introduction

There is a growing interest in expressing and exchanging knowledge and information via the World Wide Web in such a way that it is computer interpretable and enables interoperability of systems. In other words, there is a requirement that information can be exchanged between systems of various parties without the need for data conversions.

The need for interoperability of systems and for integrating data from different sources implies requirements on the expressions that are exchanged between different systems of various parties. It also implies requirements on the data and data structures for storing integrated data in databases or in cooperating databases. Interoperability and integration requires the harmonization and standardization of the data and their definitions, which implies the need for the use of a common terminology and a common way to express information in an unambiguous and system independent way.

The Gellish Semantic Modeling Methodology and its accompanying family of formal languages provide an effective and powerful methodology and languages to address the above-described needs. The methodology provides guidance for a process of growing towards the use of common formally defined languages.

The methodology promotes that ideas and facts are expressed as relations between things, whereas each relation is classified by a kind of relation. The core of each expression is thus a triple. Each triple is accompanied by at least an intention and several contextual facts. The triple structure is shared with the subject-predicate-object method that is applied in the Resource Description Framework (RDF), which is intended to be a framework for representing information in the Web.

To be computer interpretable the components and structure of expressions shall be unambiguously defined. Such unambiguous definitions of expressions therefore shall be expressed in a formally defined language. As such a language must also be human interpretable it follows that a formally defined language has preferably strong similarity with natural languages. The Gellish family of formal languages, among which Formal English and Formal Dutch (Formeel Nederlands), addresses these requirements and provides such languages. Gellish formal languages are formalized subsets of natural languages that are interpretable by computers as well as by humans. The formal languages are described in the book ‘Semantic Information Modeling in Formalized Languages’ [Ref. 2].

A computer interpretable formal language definition includes an electronic taxonomic dictionary that is computer interpretable. Such a dictionary is a very practical instrument for making definitions and knowledge available in a structured, standardized and transparent way via computer programs, possibly via the World Wide Web. Therefore, the languages in the Gellish family of formal languages enable communicating in a Semantic Web that go far beyond basic languages such as RDF/OWL.

The definition of formal languages consists of a language defining ontology, which mainly comprises a network of Definition Models of concepts. Once a formal language is defined, it can be used for expressing information and knowledge. Using a high quality formal language enables developing, maintaining and interpreting semantic information models, including (additional) Definition Models, Knowledge Models, Requirements Models and Product and Process Information Models. Formally expressed models can be reused and combined in other applications and can be exchanged and unambiguously interpreted by other systems, without the need for costly data conversions.

This book provides guidance on how to develop such semantic information models according to a consistent Information Model Architecture. The methodology also includes processes for developing and extending formalized languages and their smart taxonomic dictionaries with varying terminologies. Such developments and extensions, customizations and mappings are mechanisms to grow from various current situations towards the use of a common formalized language and their adaptation to the needs of particular businesses and organizations. A formal language definition in a taxonomic dictionary forms the basis for the development and interpretation of information models.

Collections of knowledge models form knowledge libraries, also called object libraries or product catalogues. Such libraries are typically used as knowledge source during designs of new items, systems and facilities, or product selections.

General requirements about kinds of things or procedures are conventionally expressed in best practice guides and product quality standards. They are typically used as standard specification of requirements by industrial parties. In their modeled form, as requirements models, they contain computer interpretable expressions of general requirements and can then be used as guidance by the designers as well as for computer-assisted support for verification whether delivered products and systems satisfy specified general requirements.

Finally product and process models are described as a powerful mechanism for information storage and retrieval about products and facilities and their operation and maintenance as well as for modeling of human activities.

The first section of this book presents a general Information Model Architecture. An understanding of the Information Model Architecture is important for insight in expressing and interpreting information in formal expressions. It is a prerequisite for developing sound taxonomic dictionaries, for creating powerful knowledge models and for making optimal use of the available knowledge.

The Information Modeling Methodology and the Information Model Architecture assumes that a formal language is used for modeling knowledge and information. Therefore, a brief introduction is given into formal languages and in particular into the ISO standards based Gellish family of formal languages. After that the Information Model Architecture is described. The family of languages consists of Formal English, Formal Dutch (Formeel Nederlands), etc. Formal English is used throughout this book in the example expressions. A more extensive description of that family of languages can be found in Semantic Information Modeling in Formalized Languages [Ref. 2]. The electronic, computer interpretable formal language defining ontology (usually denoted as ‘the (formal) dictionary’) is provided in the Formal English Taxonomic Dictionary-Ontology that is available as a database with search engine via the Gellish webshop [Ref. 3]. Its source can be acquired via Gellish.net. The fundamentals of the formal language and methodology are discussed in Formalized Natural Languages [Ref. 1].

The subsequent sections provide dedicated guidance for specific modeling tasks in the total process of expressing information and knowledge during the process of creating and extending information models, smart domain dictionaries, knowledge libraries and collections of requirements. They also provide guidance on using modeled knowledge and requirements in design systems and in information storage and retrieval systems.

A library of knowledge models builds on definitions of kinds of things and provides knowledge about those defined kinds of things. It is a source for knowledge about individual things of those kinds (also called concepts or classes). Such formally expressed knowledge can therefore be used, for example, for the design of individual things, by deriving ideas about individual things (instances) from the general knowledge about the kinds of things. Such knowledge can also be used for guiding the search for other knowledge about kinds of things or for documents and other information about individual things.

Individual organizations or branch organizations can extend the general dictionary and knowledge bases that are available from a support organization for developing their proprietary knowledge bases. With such knowledge bases they can support their design processes and facilitate storage and retrieval of information and knowledge and they can arrange standardized exchange of system independent data and knowledge with their business partners.

Furthermore, the uniformity of information and knowledge bases that are expressed in a formal language provides a basis for interoperability and knowledge sharing in larger communities and for the creation of widely applicable tools that support design, storage, retrieval and exchange processes in those communities.

Using knowledge bases can significantly increase the speed of design processes of individual objects in projects and of the recording of information about those objects and processes during or during operation. It will result in quality improvements so that projects can be executed faster and at lower costs.

Finally, this book describes how designing products and verification of designs against requirements can benefit from usage of knowledge and requirements models in projects and in operational situations. Especially how such models can simplify and improve the business processes or creating product and process information. Furthermore, guidelines are provided for developing powerful Facility or Product and Process Information Models that describe real world facilities, products and processes and their documentation, which can act as a flexible and effective means for storage, retrieval and exchange of information.

To facilitate the understanding and application of the Gellish Modeling Methodology, a glossary with the most important concepts is provided as the last chapter of this book.

2 Semantic information modeling

2.1 What are semantic information models?

Traditionally information about products, facilities and processes is recorded in documents. It started with only paper documents. In modern businesses and organizations such information is typically contained in a mixture of electronic documents and information models that consists of computer interpretable ‘data’ or expressions in databases.

Conventionally the data is stored as ‘instances’ of ‘entities’ and ‘attributes’ of various ‘data models’. Data models represent carefully designed data structures that have a role as predefined templates for expressions. Those data structures are implemented in the form of database tables or data exchange files. The data are filled-in in those template data structures via a user interface, thus making expressions by filling the rows in database tables or data exchange files. In this conventional approach the data models act as limited and application system specific languages, whereas the knowledge about the data models is essential for a correct interpretation of the data, thus for interpreting the meaning.

In Semantic Information Modeling this is different. Semantic Information Models are computer interpretable collections of formal expressions in which the information that is required for a correct interpretation is included in the expressions themselves. This interpretation possibility is provided because the information is expressed in a formalized language. This requires that such a formalized language is predefined. Therefore, this book includes sections that explain how to create taxonomic dictionaries-ontologies that define formal languages.

Thus, when information is expressed in semantic information models, then the models consist of a collection of expressions in a computer interpretable formal language. Such an information model may reflect for example products, their components and their operation or usage and their properties. Other information may be expressed in document files and data files in various formats. Expressions in semantic information models can includes not only terms that denote products and processes, but also identifiers that represent natural language texts (sentences, paragraphs, etc.), as well as names and addresses of electronic document files. When all such information is integrated, then all such texts, document files and data files are integrated by relations with other elements in the information model.

A product or process information model is a collection of *expressions of ideas or facts* about one or more *individual things and/or occurrences*. Those individual things and occurrences and their components and aspects are classified by kinds of things (concepts), whose definitions are captured in a dictionary that is a component of the definition of a formalized language.

Information usually consists of pieces of text or spoken sentences in natural language. However, information is also expressed in the form of drawings, in 3D models and in tabular form, such as tables in documents, but also tables in databases. Interoperability of systems requires that all such information is integrated and interpreted via computers. This requires that it is partly or completely transformed into formal expressions in a formal language. This is possible, because it appears that all information can be expressed as a collection of binary relations between things. Such collections of expressions of binary of relations are called semantic information models. Thus:

A semantic information model is a collection of formal expressions that include what is required for its unambiguous interpretation.

The core of each formal expression is a binary relation between things, whereas ‘things’ can denote anything. The related things on their turn are related with other things, so that semantic information models that are expressed in a formal language in fact form one integrated network of relations between things, whereas the language defining ontology network is also integrated in that network. A graphical representation of such an integrated network is illustrated in Figure 1.

A textual tabular computer interpretable version of such a network is called a semantic information model. The textual model consists of a collection of expressions that represent binary relations between things. Various overlapping cuts out of the network form partial models, so that it is also possible to describe a formal semantic information model as a collection of overlapping models.

A typical example of an information model is presented graphically in Figure 1.

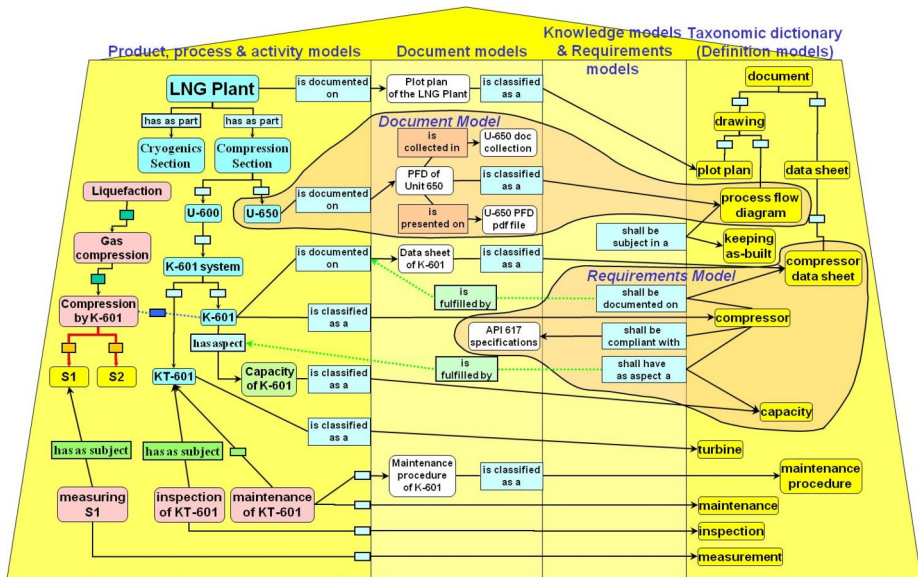


Figure 1, A semantic information model

The example that is used in Figure 1 is a about a complex object, being a process plant.

The larger boxes with rounded corners illustrate individual things and concepts that are components in the models. The small and large rectangles with sharp corners are situated in lines between the boxes and illustrate the relations between the components in the model, whereas different colors indicate different types of relations. These relations include not only relations between the components in the product models, but also relations between the components and the processes, activities, properties and documents.

A typical Semantic Information Model as is illustrated in Figure 1 consists of the following sections:

1. A formalized language definition.

This includes a taxonomic dictionary-ontology in which the formalized language is defined. It covers the definition of concepts and terminology as well as the definition of standard kinds of relations and their representation by phrases. It defines that language in which information, knowledge and requirements is expressed. The language definition is illustrated on the right hand side of Figure 1 (note that only a few

example relations between the concepts are drawn). The dictionary provides definitions (*definition models*) of all the concepts and of the kinds of relations. In most information models only a subset of those concepts is used. The taxonomic dictionary is the fundament for interpretation of any Information Model and forms an integral part of them.

2. Product models and process models of individual things and their relations.

The left hand part of Figure 1 illustrates a product composition or breakdown structure, together with processes that take place in the products and activities on its components, as well as properties of those components. Each component is described in its own product model, whereas the components are usually connected to other components and assembled in larger assemblies, together forming higher aggregation product models. Each individual product, component, activity and each aspect of them is classified by a classification relation with a kind of thing in the dictionary to specify what kind of thing it is. These classification relations define the individual things and make them elements of the applied formal language. In fact those expressions extent the formal dictionary with individual things and thus they extent the vocabulary of the formal language. This means that the individual things become known in the formal language and are allowed to be used in other expressions. The collection of names and identifiers of individual things need to be managed in order to maintain uniqueness of denotations and to avoid ambiguity.

3. Document models related to products or processes.

The second part includes the document repository with electronic files and models of documents about the products and their operation and maintenance. Each individual document and file is classified by a classification relation with a concept in the dictionary, such as a kind of document or kind of file.

The Product models and Document models together form the core of an Information Model.

4. Knowledge models and Requirements models.

The ‘shall have...’ and ‘can have...’ relations in the third box express general requirements specifications and knowledge about kinds of things. Those requirements and knowledge or standard specifications are therefore expressed as relations between concepts (kinds of things) or

documents, whereas the definitions of those related kinds of things are included in the dictionary section.

Thus Figure 1 illustrates how to integrate knowledge and requirements as well as information about individual product and documents in one information model.

In summary: Semantic Information Models are collections of expressions that are formulated in a formal language, which definition and interpretation rules form an integrated whole with the expressions. The scope of Semantic Information Models is only constrained by the available concepts of the formal language, i.e. its semantic richness. Because of its extensibility the scope is basically unlimited. Thus it may include data and documents and may describe products and processes, their operation and maintenance, etc. And when each individual thing and each relation in the models is classified by at least one concept (a kind of thing or kind of relation) that is defined in the dictionary of the formalized language, then the meaning of the expressions can be interpreted from the expressions.

Semantic Information Models as well as the models by which the formal language is defined can be stored in databases and files with a very simple and uniform structure. That structure, called the Gellish Expression Format, is described in the free available document ‘Gellish syntax and contextual facts - Definition of universal semantic databases and data exchange messages’ [Ref. 4] and is also included in the book ‘Semantic information modeling in formalized languages’ [Ref. 2].

2.2 Models of individual things versus models of kinds

Before we discuss information models in further detail we need to clarify a few terms and phrases to denote basic concepts in formal languages. The meaning and distinction between those concepts need to be clearly understood and is therefore discussed here in more detail.

Things

In the context of this book the term ‘thing’ can denote anything, including physical objects, aspects such as properties and qualities, values, units of measures, roles, functions, activities and occurrences, documents, symbols, etc., etc., and also relations, such as classification relations and specialization relations. Thus the term ‘kind of thing’ is synonymous with the term ‘kind’.

Kinds or concepts

Dictionaries and knowledge libraries contain definitions and knowledge about kinds, also called concepts or classes.

Thus, what is a kind or concept?

A concept or kind is something that is the common nature of a number of things. In other words, it is a category of which the members commonly have a number of similarities, such as aspects or composition.

Thus the term ‘concept’ and ‘kind’ and ‘kind of thing’ are used as synonyms in this book. The term ‘class’ could also be used as a synonym, but it may cause confusion, because in information science classes typically correspond with particular data structures for ‘specific data about kinds’. In semantic modeling kinds do not correspond with particular data structures. Therefore the term ‘class’ is usually avoided in this methodology and ‘kind’ or ‘concept’ is used instead.

An example of a concept is ‘centrifugal pump’, which is defined as: a pump that is intended to increase the pressure of a liquid by means of one or more rotating impellers.

Note that this concept definition does not imply any data structure for data about centrifugal pumps.

The definition of a concept can also be described as follows (all definitions are provided in the Formal English Taxonomic Dictionary-Ontology):

A concept is something that is a commonality between things, rather than an individual thing.

A concept has an explicit or implied basis for inclusion or exclusion from being of that kind.

In other words, a concept has a definition that implicitly or explicitly defines constraints that form a basis for determining whether a thing belongs to or does not belong to that kind.

These criteria for inclusion or exclusion can relate to all aspects of things, including their behavior and their composition.

A concept or kind can classify a *single thing*, but it can also classify a *plural thing*. The difference between a single thing and a plural thing is that the elements in a collection (a plural thing) don't have a particular position relative to each other, whereas the parts of a single assembly do have a particular position relative to each other, possibly explicitly expressed by a connection relation.

For example, the concept 'collection' can be used to classify individual collections and the concept 'bolt' can be used to classify each element in a collection of bolts. Note that for that latter classification the term 'bolt' is still singular as in a formal language kinds are normally denoted by terms in singular.

A single object can be either a *single component object* or it may be a *composite object* that is composed of multiple parts. A composite object can be either an *assembled object* or an *arranged object*. The difference is that the components in an arranged object don't have connection relations with each other, whereas in an assembly the parts are normally connected. For example, a flange is a single component object, a pump is an assembled composite object, composed of many parts, a village is an arrangement of houses and a stock of valves is a collection of things.

Kinds of objects

The term 'object' is often used to denote *physical* objects, which are things that obey the laws of physics and that possess aspects, but are not aspects themselves, and that can play roles in occurrences. Examples of kinds of (physical) objects are: tree, mountain, country, stream of water, building, road, car, ship, refinery, pump, bolt, person, etc. For example, an object library typically contains definitions and knowledge about kinds of physical objects.

Individual things

We use the term individual thing (or individual object) when we denote an example or exemplar that is of a kind. Individual objects in reality are for example, the Eiffel tower, the M1 motorway, my car and John Doe. An individual object may also be an imaginary individual object. For example, a new motorway that is being designed and is called the A45 motorway. Such an imaginary object 'exists' only in the designer's mind, whereas information about it is stored in a database and in documents. The concept 'individual thing' has a wide meaning as it also includes not only real and imaginary individual physical objects, but also individual real and imaginary historic

and future activities, individual aspects (properties) of individual objects, individual relations, etc.

A design process, for example the design of a building, a factory or some of their components, results in information models about individual objects. Those individual objects are assembled and thus together they form the designed object. This means that the integrated set of information models of the components of a whole assembly, such as a building or a process plant, form an integrated information model, such as a building information model or a plant information model of the whole object and possibly includes also activities and other occurrences related to the objects.

Thus an information model may contain information either about real world individual objects, or about imaginary individual objects or about both.

Note: There is a difference between a *kind* and a *typical individual thing*. Typical things are individual things that have a particular status of ‘being typical’ (for others). They are normally defined for the purpose of reference or for copying of their specifications when an equivalent specific individual thing is created. For example, a typical individual object can be a prototype individual, which can be intended as a “template” or “master copy” for the derivation of one or more specific individual objects, such as a (typical) symbol in a symbol library. A typical individual can also be representative for a the individual elements in a collection. Typical individual objects are defined in the same way as specific individual objects. On the other hand, kinds are abstraction things, whereas kinds cannot have specific locations, nor can they have connections to specific individual objects. Therefore kinds are defined in a different way.

Classification relations

Each individual thing can be related to a concept of kind of thing that classifies the individual thing. For example, the A45 <is classified as a> motorway. A classification relation thus specifies that the individual thing is classified by a kind of thing, which means that it is declared that the individual thing satisfies the definition of that kind. In a formal language it should be the case that a definition of each used kind is included in the dictionary that defines the concepts and vocabulary of the formal language. Furthermore, knowledge models might add knowledge about those kinds of things to the definitions. For example, knowledge models may contain optional design solutions, or requirements for kinds of things.

Later chapters discuss in more detail why and how objects and other things are classified by explicit classification relations.

Conclusion

It is important to clearly distinguish when information about individual things is modeled and when knowledge about kinds of things is modeled. This distinction is fundamental as dictionaries and knowledge models are about kinds of things whereas product and process information models are about individual things, whereas both domains use different kinds of relations to express ideas and facts about the things.

Note that conventional data modeling requires that models of kinds are created before data entry is possible for creating models of individual things as ‘instances’ of the patterns about the kinds. Furthermore, the models of kind constrain the possibilities for entering data about individual things. In semantic information modeling that uses formal languages there is no obligation to create models of kinds, nor should such models precede or constrain models of individual things. Models of kinds may be helpful for expressing knowledge and requirements, but they are no prerequisite.

2.3 The purpose of semantic information models

The purpose of Semantic Information Models for facilities, products or processes is to provide easy access to data and documents about individual objects, their components and their use, operation and/or maintenance as well as about definitions, knowledge and requirements for kinds of such things. Furthermore, their purpose is to increase the quality and consistency of the information by verification of the compatibility between information about the individual things and the knowledge and requirements about them. And finally their purpose is to enable interoperability of multiple systems that contain parts of the integrated models and that share the formal language definition. For example, a distributed collection of Information Models may be exchanged partly or completely between parties and may together contain documents and data about a building (then it is called a Building Information Model or BIM), a ship or a process plant and may integrate all information about the design, manufacturing, operation and maintenance of them.

There are three categories of information models:

- Document oriented information models
- Data oriented information models
- Combined documents and data oriented information models.

A document oriented information model includes a model of individual objects which primarily consists of a breakdown of one or more modeled

objects in parts, parts of parts, etc. and that may contain other relations between the parts, its functions, processes, etc. The model also includes documents and relations between those documents and the components of the modeled objects about which the documents provide information. Such an information model forms the content of a document management system that enable either to search directly on document titles or first to navigate through the object model to search for a component of the object after which the system can display documents about the selected component. Selecting the required document may launch a browser to view the selected document.

A data oriented information model will also include a model of individual objects. However in addition to that it includes properties and other aspects about the objects and their components, processes, etc. For example, it may contain diameters and lengths of components, operating temperatures of fluids, activities and processes in which the objects are involved, etc.

A combined information model integrates documents in a data oriented model.

2.4 Kinds of semantic information models

Information can be expressed in many forms. Such forms can be small sentences or larger pieces of text in natural language (including also complete documents), drawings, multi-media files, etc. Semantic modeling of such information means decomposing the information in components (basic semantic units) and converting those components into expressions in a formal language. However, it may be that some of such natural language texts, drawings etc. is not modeled but nevertheless it is possible to include them as pieces of information or ‘information objects’ (binary large objects, ‘blobs’) in information models. However, be aware that normally such information is then not expressed in the formal language and may not be computer interpretable.

The nature of the information may be that it expresses knowledge, requirements or definitions or that it expresses information about individual things or combinations of those. Depending on that nature the resulting models will be knowledge models, requirements models, definition models, product or process models or integrated information models.

Knowledge models

A *knowledge model* is a coherent collection of expressions of possibilities about things of a particular kind. It does not include modeling of ideas or

facts about individual things. A knowledge model expresses *what can be the case* for things of such a kind. It is optional whether information (expressions) about individual things or values of their aspects is recorded. For example, a knowledge model might contain a statement that a car can have a (some) mass. This is an example of factual *knowledge* about what can be the case for things of a particular *kind*. However, the concept ‘car’ is neither *defined* by its mass, nor by the fact that every car has *some* mass. A kind of physical object is only *defined* by a property when the *value* of the property is fixed for every object of that kind.

A kind of thing about which knowledge is modeled can be a kind of physical object or it may be a kind of occurrence or any other kind. A knowledge model about a kind consists of one or more relations between the kind and other kinds. For composed kinds, the knowledge model of a kind that acts as a component, is often considered to be part of the knowledge model of the composed whole, although strictly speaking component knowledge is not knowledge about a composed whole.

Thus, a semantic knowledge model is a model of that makes knowledge about a kind of thing explicit and computer interpretable by expressing that knowledge in a formal language, which means that it is expressed as classified relations between concepts. All the concepts that are used in knowledge models and all the kinds of relations are defined explicitly and in an unambiguous way in a formal dictionary. This enables that computer software can interpret the meaning from the model and the language definition and can reuse the knowledge by acting on the content of the model.

This implies that knowledge models do not define their own concepts, but they rely completely on the definitions of the concepts in the formal dictionary in which definition models are collected.

Definition models

A *definition model* is a special kind of knowledge model. A definition model contains only expressions of *what is by definition the case* for every individual thing of the defined kind. Thus those expressions define such a kind.

Definitions include expressions that specify the necessary and sufficient constraints by which kinds (concepts) are defined. Those constraints are intended to be used as the formal criteria to determine whether something belongs to the kind or not. There may be other ideas that are possibilities or that are typically the case, or are the case for well formed exemplars of the defined kind, but are not necessarily the case for things of the defined kind.

Such other possibilities do not belong to definition models, although they can be used as informal criteria. For example, the fact that a (well formed) person (normally) has two arms is a strong indication that something with two arms is a person. However, if somebody has only one or no arms it does not mean (s)he is not a person. Therefore, the constraints shall be necessary and sufficient in order to belong to a definition.

Statements about individual things do not belong to definition models, but they belong to product and process models of individual things. For example, the fact that ‘my car’ has a measured maximum speed of 165 km/h is factual *information* about what is the case for an *individual* object and the formal expression of such information does not belong to a definition model of a car, it belongs to the model of an individual thing, such as ‘my car’, which model is not called a knowledge model, but a product model.

Definition models include models that define generic concepts as well as detailed product types.

Product type models, such as standardized components models and manufacturer’s models specify the definition of detailed subtypes of more generic kinds. They usually leave a number of degrees of freedom for individual things of the defined kind. For some aspects they may specify a constraining list of fixed value options or bounding constraints, such as minimum and maximum values and for other aspects they may specify no constraints at all.

Requirements models

A *requirements model* is another special case of a knowledge model. A requirements model contains expressions of *what shall be the case* according to a particular party or standard.

Requirements are statements that express that a particular party declares that something must be the case for things of a particular kind, when it is or becomes under his or her authority. This includes also proprietary, industry, national or international standards. Many of them express requirements that are only applicable if some authority declares that and when they are applicable.

Requirements may include fixed value requirements (with a tolerance) and bounded requirements that include constraints that specify ‘allowed values’. For example, they specify that a value may be within a range or must be selected from a collection of values that constrain the values of the aspects of that kind in that context.

Requirements that describe what a party wants about *individual* things are outside the scope of requirements models. Examples of imaginary individual things are the designed individual facilities or the described individual prototypes that are to be used as example exemplars according to which other copies are intended to be made. The modeling of such individual things result in product and process models as will be described below.

For clarity we will use the term knowledge model only for models that express possibilities and not definitions or requirements. However, formally, kinds of relations that can be used to express *what shall be the case* for things of a kind are subtypes of kinds of relations that can be used to express *what can be the case* and kinds of relations that can be used to express *what is by definition the case* are subtypes of kinds of relations that can be used to express *what shall be the case*.

Models of individual things

Models of individual things are models that consist of relations between individual things and individual parties, or relations between individual things and parties and kinds. For example *product and process models* are models describe individual products and individual occurrences in which individual things and parties are involved. Individual things and parties include not only physical objects, such as products and processes, but also individual aspects and individual relations, as well as individual persons, organizations, locations, countries, documents, etc. These models include compositions of the individual things and their properties and other aspects and relations between components as well as relations between components and processes in which they are involved. They may also include models of those components and the processes, activities and events in which they are involved.

Integrated information models

Integrated information models are models that are combinations of expressions of knowledge, definitions and requirements, as well as models of individual things and parties. Such models are a mixture of definitions, options and obligations.

The information in a semantic information model about individual things can be verified on its compliance with knowledge models or with requirements models, whereas definitions are used to enable the interpretation of the expressions. It is also possible to use the knowledge in knowledge models or

requirements models to create information models about individual things by derivation as will be described later.

Information can be:

- Explicitly modeled in formalized languages, and thus being interpretable by a computer;
- Information that is not explicitly modeled, but is contained in documents and drawings that are not computer interpretable or written in another (natural or formal) language;
- Partly modeled and partly included in the models as ‘information objects’.

2.5 Knowledge Libraries

A knowledge library is a collection of knowledge models that express possibilities about kinds of things. Thus the knowledge in a knowledge library is captured and expressed in distinct knowledge models. A knowledge library and its knowledge models consist of relations between concepts, whereas those concepts shall be (pre)defined explicitly in definition models. Those definition models are contained in a taxonomic dictionary-ontology that defines a formal language.

Thus a database that contains knowledge models and uses definition models can be called a knowledge base or knowledge library.

A library of knowledge models (a knowledge library) and a dictionary together can be called a modeled encyclopedia, because it contains more knowledge about the concepts than a dictionary.

The kinds of things in a knowledge library, about which knowledge is contained, form nodes in networks of knowledge. Those nodes can be used as a kind of framework for attaching further knowledge and requirements about those kinds of things or as classifiers for individual things.

A knowledge library consists of the following parts that are described in more detail later in this document:

- Definition models, each of which expresses a definition of a kind of thing;
- Knowledge models, each of which expresses a particular amount of knowledge about a kind of thing;

- Terminology to denote the kinds of things, such as by terms, symbols, names, abbreviations, codes, etc. as well as names, codes, etc. of individual things.

Each definition of a kind of thing, as well as knowledge about a kind of thing can be expressed in a number of expressions.

A knowledge library contains a collection of knowledge models about kinds of things that are known and used by a wide community, but it may also contain collections of knowledge models that are specific for particular subject discipline areas. That is the reason why the development of knowledge libraries is typically organized per discipline area and is typically coordinated by an interdisciplinary body.

To make knowledge in a knowledge library usable for any kind of user and system it is required that domain specialists model their knowledge in an unambiguous way. Therefore, knowledge libraries should be system independent.

System independency means that the modeling method is not dependent on particular tools (such as particular software or databases) and that the resulting models can be migrated from one system to another without the need to make dedicated transformations. In other words, the models need to be captured only once and should not be needed to be recreated or transformed when in future other systems will be applied.

2.6 Benefits of integrated models

Integrated models are a source of definitions, knowledge and requirements about kinds of things (classes of things or concepts), whereas information about kinds of things is by definition applicable to individual things that can be classified by such kinds. The information that is contained in integrated models can therefore be used in various ways. For example, it can be used during the design of individual objects, or to verify whether designs are in accordance with general knowledge and specifications of requirements or to provide knowledge for students and inexperienced users or to harmonize the content of various databases or it can be used to disambiguate terminology in various systems and data exchange files.

In order to express that an individual thing is of a particular kind, the individual thing is classified, which means that we create a classification relation between the individual thing and the kind of thing. For example:

B-23 is classified as a road

Classification of individual things ensures standardization and eliminates ambiguity within projects and companies. Adherence to those standards causes that everybody uses the same terms for the same concepts.

Figure 2 illustrates how a classification relation for road B-23 provides access to knowledge in a knowledge library.

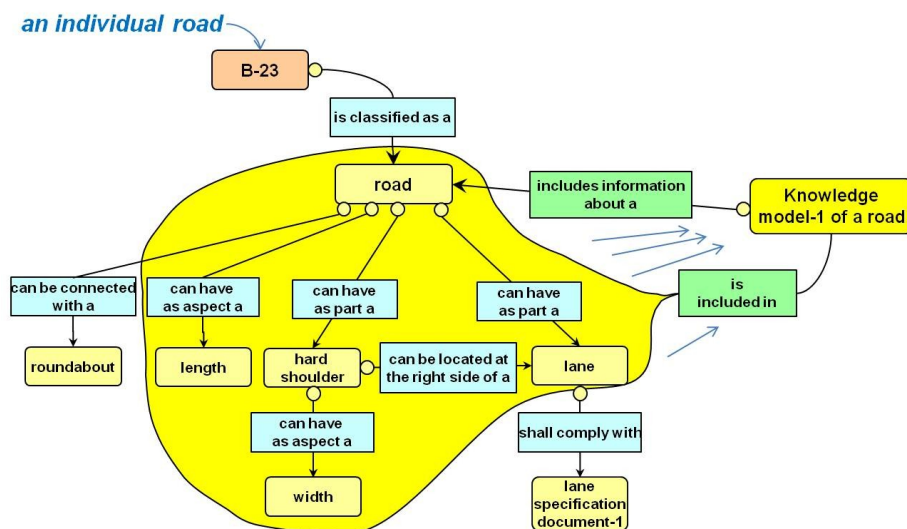


Figure 2, Classification as an access path to knowledge

In Figure 2 the object B-23 at the top of the figure represents for example a new road that is going to be designed. The fact that B-23 is a road is expressed by the above-mentioned classification relation between the individual object B-23 and the concept road that is selected from the dictionary of the formal language.

The yellow area in Figure 2 indicates a knowledge model, which consists of a collection of expressions about the concept road, which in this example includes knowledge about the kinds of parts shoulder and lane. The knowledge model is represented itself as collection of expressions (a graph in RDF) called 'Knowledge model-1 of a road'. The box with the phrase 'is included in' represents a number of inclusion relations between the knowledge statements and Knowledge model-1, which relations specify that those statements are included in the collection. For completeness, it is also specified by a relation that the knowledge model <includes information about a> road.

Relations that express knowledge, but are nevertheless outside a 'cloud' that defines a knowledge model indicate that there exists other knowledge about

the concept road that does not belong to knowledge model-1. For example the statement that a road can be connected to a roundabout, as is expressed at the left hand side of the figure

So, by classifying the individual object B-23 by the concept road that appears in the formal dictionary a computer can determine that there is a knowledge model about a road available in a Knowledge Library. In that way knowledge about that concept is made available for the design of individual objects, such as B-23. The same holds for the availability of requirements and definitions. How such knowledge, requirements and definition can be converted into proposals for possible facts about individual things is described in chapter 5.

The process of classification is explained in further detail in par. 5.2.

3 Formalized natural languages

3.1 Formal language defining ontologies

As mentioned before, Semantic Information Models should use a formalized language for making expressions, such as statements, requirements and modeled definitions. The main advantage of using a formal language is that such a language provides formally defined and computer interpretable concepts and expression structures and provides a common basis for data exchange and interoperability between systems of various parties.

The definition of a flexible, extensible and system independent formalized natural language is briefly described in this chapter.

Formal languages are naturally derived from natural languages (although for formal languages in mathematics and logic that is not a requirement). The concepts and expression structures can be also based on definitions provided in ISO, IEC and other vocabulary and data model standards.

We distinguish between the definition and the usage of formal languages. Formal languages can be defined in language defining ontologies that are accompanied by a syntax specification. A language defining ontology includes a vocabulary with the specification of the terminology of the language, a taxonomic dictionary with the definitions of the concepts that are denoted by the terms in the vocabulary and that are arranged in a taxonomic hierarchy and additional relations between concepts that specify what are formally correct expressions.

3.1.1 Vocabulary

A formal language definition includes the specification of its vocabulary, which consists of the terms and phrases that are allowed in expressions in the formal language. The vocabulary does include not only terms and phrases that denote allowed concepts but also terms for denoting individual things. Furthermore, the terminology should not only include ordinary natural language terms, synonyms and abbreviations, such as found in conventional dictionaries. The terminology should also include composed terms to represent more specialized concepts that are used in application area's (domains) that are to be covered by the dictionary and language. The vocabulary should even include phrases that represent concepts that are kinds of relations, such as <is a kind of> or <is a part of>. These examples illustrate

that although phrases denote one concept they are nevertheless expressed in English by a number of words in a sequence. Composed terms, such as ‘centrifugal pump’, are typically not included in ordinary dictionaries. However in normal business practice and database content there are many concepts in use that are denoted by such composed terms.

The vocabulary of a formal language does not only consist of terms for kinds of things, it should also cover names/terms for generally known individual things. For example geographic objects, such as countries, cities, etc. could be included in the vocabulary.

When people express information about individual things in a formal language, they often introduce new names for the individual things that they are describing or are referring to. This means that on-the-fly they add new terms to the vocabulary of the language. Quite often this implies that different people or different organizations use their own terminology, thus adding synonyms and homonyms to the formal language. The use of various terminologies by various organizations is enabled by representing each thing (idea or phenomenon) by a unique identifier (UID) and to allow that various organizations allocate their own terms to those UIDs. This supports interoperability, while maintaining variation in terminology. The mechanism is further described in par. 3.3.

People should be aware that this process of extending the language with new terms should be managed, especially to ensure that things should not be re-created by different people and thus to avoid multiple identities for the same things. Furthermore, the method for adding new terms should satisfy rules from the methodology to extend the language as described in chapter 4.3.

3.1.2 Dictionary

The vocabulary forms an essential part of an electronic dictionary for a formal language. The dictionary adds primarily textual definitions that describe the meaning of the concepts that are represented by UIDs and denoted by the names, terms and phrases. An advanced formal dictionary also contains modeled definitions as is described in chapter 10.

Expressions in a formal language not only use terms for concepts and individual things, but also phrases that denote kinds of relations (or ‘predicates’) for classifying relations between things. Therefore, a formal language dictionary should not only include definitions of concepts, but primarily include definitions (definition models) of kinds of relations (‘relation types’) that are necessary for the creation of expressions in the formal language.

3.1.3 Taxonomy-ontology

A formal language can be formally defined in a language defining ontology. Such an ontology includes definitions of the concepts that define the language and their meaning, including also definitions of the kinds of relations that are required to make expressions, but such an ontology should not include specifications of knowledge (possibilities) or requirements about kinds of things. In addition to that, a formal language definition includes a syntax that defines how the components of expressions should be arranged for making expressions. In this book we will use as syntax the Gellish Expression Format.

The usage of natural language concepts and kinds of relations in formal languages means that the knowledge and information that is expressed in a formal language is insensitive for changes in information technology. It also means that the semantics are stable and that the expressions can be transformed in a simple way to other or newer technologies. For example, a formal language can be expressed by using SQL as well as RDF/XML or OWL technology.

Thus a formal taxonomic dictionary is not just a conventional dictionary but its concepts are arranged in a taxonomic hierarchy and it is an ontology because of the use of other kinds of relations to complete the language definition. Therefore the dictionary is denoted as an electronic taxonomic dictionary¹.

A taxonomic dictionary that defined a formal language should satisfy the following criteria:

1. The dictionary contains definitions per concept, each represented by a unique identifier (UID; a natural number). Note that ordinary dictionaries usually provide multiple different definitions of a single term, whereas it is unclear whether those definitions are alternative definitions of the same concept or whether they are definitions of different concepts. A formal dictionary explicitly distinguishes homonyms (the same term for different concepts) and explicitly specifies which terms are used as true synonyms, including also abbreviations, codes, etc.
2. The dictionary is itself expressed in the formal language of which it contains defined concepts. For example a database may include a

¹ http://en.wikipedia.org/wiki/Gellish_English_dictionary

language defining taxonomic dictionary that is stored as or can be imported and exported as files in the Gellish Expression Format.

3. The dictionary is completely arranged as a taxonomy, which is a subtype-supertype hierarchy of concepts. This means that each concept is defined as an explicit subtype of one or more supertype concepts by specialization relations. This enables inheritance of definitions, knowledge, and requirements.
4. The dictionary includes also concepts with multi-term names. For example: line shaft centrifugal pump. Ordinary dictionaries will only define the separate terms, from which it cannot always be inferred what the multi-term means.
5. The dictionary defines kinds of relations, also called relation types or fact types, as being special kinds of concepts. Such kinds of relations do not appear in ordinary dictionaries, because they are word-based and do not recognize standard phrases. However, relation types represent semantic concepts that are used by everybody to make sentences. Because in a formal language these relation types are standardized and unambiguously and formally defined in a computer interpretable way, further expressions in the formal language also become computer interpretable. The meaning (the semantics) of relation types should distinguish between various categories of statements. Therefore the taxonomic dictionary includes kinds of relations that are specific for the kinds of statements that represent knowledge, whereas other kinds of relations are specific for expressing requirements, others for definitions, and again others for information about individual things. The relation types have names and aliases that consist of standardized phrases. For example the phrase <is a part of> and the phrases <can have part a> and <has by definition as part a>.
6. The dictionary includes explicit relations between concepts, using the above mentioned standardized relation types to express what is by definition the case about the defined concepts and kinds of relations. Those relations make that the taxonomic dictionary is also an ontology. If a user of an application system classifies an individual object by a concept in the taxonomic dictionary, then a computer system can infer that the knowledge and requirements about the concept are applicable to that individual object, and it can generate design proposals and can make the knowledge and requirements available to that user. This includes also the knowledge and

requirements that are available about the supertypes of that concept, because such knowledge and requirements are inherited by the subtype of those supertype concepts, conforming to the subtype-supertype hierarchy (the taxonomy).

7. The dictionary enables automatic translation and search in databases in other languages. This is possible because each concept is represented by a language-independent UID, whereas multiple terms (names) in various languages are allowed to denote the same concept. This makes it possible for facts that are expressed in one language to be automatically presented by a computer in any other language for which a formal dictionary is available. It also makes it possible to answer queries in one language using database systems that contain facts that are expressed in other languages and then present the results in the language in which the query was formulated.
8. The dictionary can be extended by private and proprietary concepts and terms. For example, company specific codes and proprietary knowledge can be added as required. Guidelines for the "proper definition of a concept" are provided in the documentation.
9. The dictionary is computer interpretable and system independent.

3.1.4 Relation types

In the Gellish Modeling Methodology, each atomic expression (a ‘unit of meaning’) consists of a binary relation between two related things. Any higher order or N-ary expression can be modeled as an N-ary relation that is related to N involved objects via N atomic binary involvement relations. Pieces of text can be included as blobs (binary large objects) to which relations are made, or they can be decomposed and converted into atomic expressions. We call this decomposition process explicitation of a text. Each resulting atomic expression should be an expression of an atomic idea or fact. Thus, each expression of an idea or fact is represented by one or more binary relations between things.

An idea about a topic can be expressed as a statement about a fact or a statement about a possible fact, but it can also be a question, a confirmation, a denial, a promise, a command, etc. As a simplification we describe the modeling methodology as if we only model facts, whereas a fact is defined as ‘that which is the case’. This means that everything ‘that is the case’ can be expressed according to the basic pattern. However, the methodology is also applicable for the expression of other intentions. Actually the methodology is

also suitable to express what is thought to be the case, what is wanted to be the case or for a question whether something is the case, etc.

The upper ontology section of a formal language definition contains primarily definition models of (standard) relation types, that are required to create any expression in the formal language and thus to create any of the models in the information architecture. Some of the kinds of relations are required to create definition models for new concepts that are extensions of the dictionary. Other concepts and relation types are defined in the upper ontology that enable expressing knowledge and again others to enable expressing requirements about kinds of things whereas again other kinds of relations represent the semantics that is required to express information about designs and real world individual objects or to relate objects to information objects, such as documents and drawings.

For example: the fact that a road can have as part a lane can be expressed according to the basic pattern in a formal language as follows:

road can have as part a lane

The brackets in this 3xample expression are added to illustrate that the phrase that represents a part-whole relation type is one unity. This example illustrates that the expression according to the basic pattern is an ordinary English sentence. It is simple and unambiguously interpretable, not only for human beings, but also for computers. As you will reckon this expression is also a correct formal language expression. The expressions are typically stored in a tabular form as follows:

Name of left hand object	Name of relation type	Name of right hand object
road	can have as part a	lane

Other examples of similar expressions about kinds of things are:

viaduct	is a subtype of	civil item
viaduct	can have as part a	pillar
viaduct	has by definition as part a	bridge deck
pillar	can have as aspect a	height

Examples of expressions about individual things are:

V-1	is classified as a	viaduct	
Pillar-1	is a part of	V-1	
Pillar-1	is classified as a	pillar	
Pillar-1	has as aspect a	H-P1	

H-P1	is classified as a	height	
H-P1	is quantified on scale as	4.3	m

The middle parts of the sentences are called relation types, because they denote the kind of relation between the related objects.

There are different categories of relation types used for expressing knowledge, requirements and information in order to express differences in meaning as is expressed in knowledge libraries and in product models or process models.

By expressing facts according to the same structure, using the same standard relation types and using the same concepts and definitions from the electronic formal dictionary it become possible to integrate knowledge and information from different sources in one consistent language and in various RDF datasets or in tables that are all in the same Gellish Expression Format. This enables computers to search for knowledge and information in various databases, combine the search results and present the integrated results to a user. It also enables that the knowledge and information is interpreted and used by computers, without the need to tune application software to conventional data models.

The use of an unambiguous formal language prevents problems that occur with many search engines, such as the fact that a query result often presents many irrelevant search results. It also significantly increases the chance that you find what you are looking for, whereas conventional representations of information use a large variety of terminology that is often not mentioned in the search criteria.

3.2 Correct formal language expressions

Information in a semantic model that is expressed in a formal language comprises a collection of expressions, presented in a syntax that is suitable for the formal language. Correct formal language expressions in a semantic model are expressions that comply with the following rules and guidelines.

Rule 1: Expression components.

A formal language expression consists of a uniquely identified relation. The relation shall be classified by one prime formally defined kind of relation (relation type) and possibly by a number of secondary qualifying kinds of relations, such as a scale (unit of measure), a probability or a severity. The relation shall be related by different classified binary involvement relations to related things that

are involved in the relation. Each kind of involvement relation expresses a kind of role that is played by the related thing in the relation.

Binary relations may be abbreviated by replacing two involvement relations by one equivalent relation, which for readability reasons is then denoted by a phrase. The justification of this is discussed in [Ref 2].

Each binary relation shall be accompanied by a number of contextual facts, as required.

Explanation:

Assume that it should be expressed that area A is separated from area B by fence S. In a formal language any expression has the form of a relation. Assume that a ternary order relation is called R3. Rule 1 then states: R3 shall be classified, and R3 shall be related by different involvement relations to A, B and S. This will result in 4 binary relations (1..4), for example expressed as follows:

1	R3	is classified as a	separation relation
2	A	is involved as separated from in	R3
3	B	is involved as separated by in	R3
4	S	is involved as separator in	R3

When it is only specified that A is separated from B, without specifying a separator, it is possible to use a binary relation R2 instead of the above ternary relation R3. Then R2 is classified by a kind of relation that is denoted by a phrase, such as <is separated from>. The abbreviated expression then becomes:

R2 A is separated from B

Rule 2: Usage of formally defined concepts.

Each expression of a main fact relates only individual things and kinds of things (concepts or classes) that are:

- Either selected from a formal language dictionary,

or are

- Properly defined subtypes of formal language dictionary concepts,

or are

- Individual things that are properly classified by formal language dictionary concepts or their subtypes (via individual classification relations),

whereas the used individual things and kinds of things are related by individual relations that are classified by kinds of relations (relation types) that are also selected from the formal dictionary.

Rule 3: Supertype(s) of a concept.

A subtype of a concept is properly defined by a definition model that includes at least one subtype-supertype relation (a specialization relation) with a direct supertype concept. A definition model shall satisfy the requirements for definition models that are expressed in chapter 10, which describes how a formal dictionary can be developed and extended.

Rule 4: Classification of an individual thing.

An individual thing is properly classified if it has at least one classification relation with a kind of thing that is selected from the formal dictionary or from its formally defined subtypes.

Rule 5: Roles in relations.

Any relation shall relate things that play roles of kinds that are required by the kind of relation that classifies the relation and that are defined in the Upper Ontology section of the dictionary.

Example 1: assume that A is a performer of B. This implies that A shall be a physical object, because the ‘is a performer of’ relation type requires a first role (performer) that can only be played by a physical object. Note that the second role (performed) is a role that can only be played by an occurrence. So B must be an occurrence.

Example 2: assume that C is a kind of D. This implies that C is a concept, because the ‘is a kind of’ relation requires a first role that is a subtype, which role can only be played by a concept (or a subtype of concept).

Rule 6: Quantification on a scale.

A relation between an aspect and a number may be classified by a scale, in which case the qualitative scale (unit of measure) shall be selected from the dictionary or from an extension that is defined conform the rules of the formal language.

3.3 Unique identification and naming

To ensure unambiguity and enabling the use of synonyms and homonyms, every individual thing or idea and every concept is represented in a (family of) formalized language(s) by a persistent unique identifier (UID). This holds for everything that is identifiable, such as physical objects as well as their aspects, for concepts and individual things, and for all other things, such as occurrences, relations and kinds of relations, ideas, etc.

Those identifiers are intended to be unique across languages and are therefore called (globally) unique identifiers. There is an ISO standard with guidelines for such identifiers (ISO/IEC 11179-5). The Gellish UIDs are compliant with that standard.

A UID should just represent a bare concept (a ‘Ding an sich’) throughout its history and should be language independent. In semantic modeling all information should not be hidden, but modeled explicitly. Therefore UIDs should not contain any implied or coded information about the represented things, contrary to what is the case with concatenated codes. That is what makes that any thing is always represented by the same UID, even in different languages.²

This is, because there is a difference between a concept and the collection of expressions about that concept. Those expressions specify relations between the concept and other concepts (a semantic network). A collection of expressions can have its own UID, if required, and there may be more than one collection of expressions about the same concept. This difference is recognized in the Gellish family of formal languages, but is often ignored in other methodologies.

In natural languages as well in formalized languages, every individual thing or kind of thing, including every kind of relation, can be denoted by a variety of terms, such as names, symbols, synonyms or codes. For an unambiguous reference to something it is insufficient to use only such a term. In addition to the term it is necessary that it is indicated from which language and language community the term originates. The language community should be as

² The only things that have no unique identifier are the terms themselves, because terms in a formal language, where all concepts are represented by UIDs, are just character strings that are already unique from themselves. For example, the term (character string) ‘bank’ can be used to denote different concepts; the string even appears in different (natural) languages. Nevertheless the character string b-a-n-k as used in a formal language is just a unique and unambiguous sequence of four characters from the Latin alphabet. The string has no meaning in itself. The UID of a concept bears its meaning by representing something.

precise as required to ensure that within such a language and language community each term is unique, which means that within a language community no homonyms should be allowed.

Although things as well as ideas are represented by language independent UIDs, they can be denoted by multiple terms or phrases in multiple languages. This enables software to automatically express ideas in any language for which a dictionary is available. This implies that expressions can be automatically translated.

For example:

The relation type that is represented in a Gellish language by UID 1146 is denoted in Dutch as <is een soort> and in English as <is a kind of> or <is a subtype of> or <is a specialisation of>. For a computer the UID 1146 is sufficient for indicating the meaning of the concept in a language independent way, whereas for a person who defines or uses a dictionary or library the terms as well as the language and the language communities are essential for a correct interpretation of expressions.

The definitions and terms used in various standards, for example those used in ISO 15926-4 and ISO 16354, are included in the Gellish English dictionary, whereas others can be included as new concepts in a formal domain dictionary or can be included as synonyms, using the same UIDs (with those standards as language community context for the synonym terms). Similar synonym relations are and can be made for terms and definitions in proprietary database implementations, such as for various database implementations in various companies. The language wide UIDs form the basis for interoperability of systems through communication (exchange of data) of messages, queries and files that are expressed in a language from the family of formalized languages.

Various organizations want to use their own terminology for denoting concepts and individual things, thus forming language communities. Furthermore, organizations that are involved in international cooperation or business are confronted with various terms and translations to and from foreign languages. Formalized languages can support the freedom to use one's own terminology and a variety of languages by allowing that concepts and individual things are denoted by various aliases, such as synonyms, abbreviations, codes, symbols, translations, etc. by specifying the language community where each term originates and by relating their terminology to the appropriate UIDs.

Formalized languages can also allow for the use of homonyms, being the same term denoting different things. However, allowing for synonyms, homonyms and translations implies that terms alone do not unambiguously denote concepts. Therefore it requires a separation between a unique identifier (UID) and various denotations. A unique identifier uniquely represents something within the family of formalized languages, whereas vocabularies contain terms that are denotations used in various languages and language communities, whereas they may also be enabled in the formalized language.

For example a concept that is represented by a UID may be related to a name that is the preferred name in a particular language community or discipline and language, whereas other names, codes etc. are related to the same UID as the preferred name in other language communities. This can be expressed in the formal language by various kinds of alias relations as is indicated in Table 1.

Language	Left hand UID	Left hand object name	Relation type name	Right hand UID	Right hand object name
English	130058	centrifugal pump	is a kind of	130206	pump
English	130058	centr. pump	is an abbreviation of	130058	centrifugal pump
English	130058	centrifical pump	is a synonym of	130058	centrifugal pump
Deutsch	130058	Zentrifugalpumpe	ist eine Übersetzung von	130058	centrifugal pump

Table 1, Identification and synonyms of objects

The following expressions from a formal dictionary illustrate that UIDs distinguish homonyms, independent of language:

language	language community	UID of concept	term
English	geography	990152	bank
English	finance	700140	bank

The distinction between UIDs and terms makes it necessary that a formal dictionary includes relations between UIDs and the various denotations in various languages and language communities, as is illustrated in the table above. When organizations want to allow for the use of aliases and homonyms, especially in their communication with others they should

implement terminology management for systems that implement the formal language. Terminology management includes registration and harmonization of terms, names, codes, etc. that are used to denote those things and it includes that UUIDs are allocated to individual things and to kinds of things that are used in those systems.

The above method of distinguishing between unambiguous UUIDs and (ambiguous) terms causes that expressions can unambiguously be formulated as sequences of UUIDs, whereas the formal dictionary is used to allocate terminology to the UUIDs in the language and language communities that the user prefers in the user interface.

In order to avoid overlap between the identification of things that are referenced in formal language expressions, including things in projects and things in dictionaries it is important to ensure that each thing is identified by a Unique Identifier (UUID) which uniquely identifies that particular thing in any expression within the family of formal languages³.

Therefore, terminology management and management of the allocation of unique identifiers are required for the language defining dictionary and for the concepts and individual things within businesses and projects.

Guideline

Everything in a system shall have a unique identifier (UUID) that represents the thing in the family of formal languages. This identifier may not be changed during the life of the thing. There may not be another thing that has the same identifier. A UUID is independent of the contexts in which it is used.

To ensure such uniqueness among various independent parties that use the same language and dictionary, it is necessary that a UUID consists of two parts: a prefix that indicates the party that allocates identifiers, and a suffix that is a alphanumeric string that uniquely identifies the concept within a domain as chosen by the allocating party. The prefix and the suffix are separated by a separator (ref. 15), such as a colon (:). For example various parties use Digital Object Identifiers with DOI as prefix (ref. 16). When working within a controlled environment it is feasible that the prefix and colon may be omitted. For example, in the Gellish dictionary the concept denoted as ‘anything’ has UUID 730000; however outside the Gellish context this concept is represented by UUID gel:730000.

³ Two different things may not have the same UUID and two different UUIDs may not represent the same thing.

Within an RDF implementation an IRI may act as a UID.

For example, <http://www.formalenglish.net/dictionary#730000> is a correct identifier of the concept ‘anything’ in a formal language in an RDF context.

The Gellish UID of anything in its dictionary, except strings, is a character string that is a natural number (positive integer), possibly preceded by the prefix and colon Gel:. The range of UIDs below 100.000.000 is reserved for the language definition. UIDs above that number are free, as described below. Within the Gellish family of languages and their definitions the reserved range is subdivided in ranges as specified below (the upper values are not included in the ranges).

1-100: Unknowns.

This range is reserved for the identification of objects (or collections of objects) that are ‘unknowns’. Thus, the use of a UID in this range means that the object is unknown. Typically such unknowns are used in formally expressed Queries in which the identity of such a thing or things is requested. For example, the UID of the left hand object in the following question is 1 with the name ‘what’. That ‘object’ or collection of objects stands in this example for an unknown collection of pumps (with their real UIDs) as will become known in the answer to the question.

UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
1	what	1225	is classified as a	130206	pump

100-1000 Testing.

This range is reserved for testing and demo purposes.

1000-30.000.000 Language definition.

This range is reserved for the definition of common concepts in Gellish taxonomic dictionaries. UIDs in this range are allocated by a Gellish language manager.

Strings

A string is a sequence of characters that is unique in its own right. In Gellish formal languages strings do not directly represent things, because all things are represented by UIDs. Strings only denote UIDs in a particular language and language community context. Therefore strings themselves do not need numeric UIDs. Note that a character string may have various

roles to denote different things (homonyms). Thus a character string is not a Gellish UID.

Examples of UIDs

Examples of UIDs of some formal language concepts are the following:

Gellish UID	Name in English	Name in Dutch
1.225	is classified as a	is geclassificeerd als een
130.206	pump	pomp
551.564	capacity (mass flow rate)	capaciteit (massastroomsterkte)
#50E-1	5.0	5,0
570.449	dm3/s	dm3/s

Note that the same UID is denoted by different names or phrases in English and Dutch. Even the decimal notation is different in the two languages (5.0 versus 5,0), whereas only the terms for the unit of measure are identical, as they are international symbols.

User defined things might have for example the following UIDs:

Gellish UID	Name in English
40.000.001	P-6501
40.000.002	capacity of P-6501
101.001.001	valve type A
501.001.002	nominal diameter of valve type A
1.501.001.002	idea 1501001002

Reserved ranges

Particular ranges of UIDs can be reserved for particular applications via the Gellish language manager. The UIDs in the above example are below 10 billion and thus they assume that the user organization has acquired the corresponding ranges of UIDs.

For example, an organization may request and receive the following reserved range of Unique Identifiers (UIDs) for the expression of information, knowledge and requirements for a particular project:

Object UIDs 108000000 - 109000000

Idea UIDs 208000000 – 209000000

Within those ranges the organization for which the range is reserved is free to define subsets. For example, following subsets might be defined:

	Object UIDs	Idea UIDs
Kinds of equipment items and kinds of aspects	108000000 – 108200000	208000000 – 208200000
Kinds of documents	108300000 - 108400000	208300000 - 208400000

A rule for applying one of the Gellish families of formal languages is that the organization that acquired a range is responsible for the definition of the concepts that are represented by the UIDs within that range. And no other organization may use the UIDs in reserved ranges for representing other things.

3.3.1 Numbers

Mathematics has developed various notations for numerical values. Herewith we propose a method by which integer numbers (positive and negative whole numbers including zero) as well as rational numbers (fractions) can be unambiguously represented by unique identifiers (UIDs) that are derived from their mathematical meaning. The method uses UIDs that are a concatenation of the prefix ‘#’, which indicates that the concept is a arithmetic number, followed by a string that specifies the significant digits in a decimal system. When optionally followed by the character E and a string that denotes a whole number that specifies an exponent. This is applied for example in formalized English as presented in Table Four: Bron van verwijzing niet gevonden.

UID of idea	UID of left hand object	Name of left hand object	UID of kind of relation	Name of kind of relation	UID of right hand object	Name of right hand object	Name of UoM	Date/ time of start of validity
12	101	T-1	1225	is classified as a	550038	temperature		
13	101	T-1	5025	is quantified on scale as	#2	2	deg C	t1
14	101	T-1	5025	is quantified on scale as	#30E-1	3.0	deg C	t2
15	102	d-1	1225	is classified as a	550005	duration		
16	102	d-1	5025	is quantified on scale as	#30E-1	3.0	s	t2

Tabel 2: Quantification of physical properties by a number on a scale

For example, Table Tabel 2 illustrates that numbers can be used to quantify physical properties on a scale, e.g. a scale of C (Celcius) or mm (millimeter) or on a scale of s (seconds), etc. The first line in Table Tabel 2 expresses the idea that T-1 is a temperature. The second line expresses that T-1 has a value of 2 on a Celcius scale, where #2 is the UID of the number 2 and t1 denoted a moment in time that the value was quantified (see below). The third line expresses that that same temperature T-1 on time t2 has a value of 3.0 on a Celcius scale, where #30E-1 is the UID of the number 3.0. The fifth line expresses that the duration d-1 is quantified on a scale of seconds by the same numeric value 3.0. on the time t2.

- Notes: 1. There is only one property T-1 which has two values that are valid on two moments.
 2. The UIDs are independent of the denotation ('names') of the numbers. For example, the concept with UID #30E-1 is denoted as 3.0 (in an American context), but as 3,0 (with a comma instead of a dot) in a German context.

3.3.2 Date-times

Periods in time are represented either by a single identifier, being a date-time, or the period is specified by a start date-time and an end date-time. Date-times are themselves periods in time that can be represented by UIDs that are derived from their quantification on a time scale as follows: the UID is a concatenation of the prefix 'dd:' and numbers that represent a year, a month, a day, an hour, a minute and seconds, including decimal part of a second on a time scale. The period has by default a duration that terminates at the end of the last digit. Table xx presents the example of an event that took place on a specific date-time.

E-1 has as date-time of occurrence

UID of idea	Name of left hand object	Name of kind of relation	UID of right hand object	Name of right hand object	Name of scale	Additional textual definition of left hand object
17	John	has as birth date-time	dd:19810309	9 mar 1981	UTC	
18	The first world war	has as occurrence date-time	dd:19	the 20th century	UTC	
319	Event-1	has as occurrence date-time	dd:20230619202615.1	the 19th of June 2023 15.1 second past 8:26 pm	UTC+01	
20	UTC	is a quantitative subtype of	571769	time scale		according to the Gregorian calender at the 0 meridian.
21	UTC+01	is a quantitative subtype of	571769	time scale		that is one hour later than UTC.

Tabel 3: Example of quantifications of periods

UID dd:20230619202615.1 on scale GMT+1. This is the identifier of the 19th of June 2023 at 8.26 pm 15.1 second past 8:26 o'clock on a scale that is one hour later than Greenwich Mean Time scale. If there is not an end of validity specified, then the duration lasts until 15.2 second past 8:26.

Note that the scales are separate concepts that are defined in the taxonomic dictionary as well.

4 Architecture of semantic information models

To bring consistency in the process to create and use knowledge models and information models it is important to use a systematic approach. Therefore we first present an overall Information Model Architecture for storing information in a semantic database. That Information Model Architecture is composed of several coherent parts or layers as is explained in the following paragraphs.

4.1 Outline of an Information Model Architecture

An architecture of integrated information models according to the Gellish Modeling Methodology consists of seven main sections as illustrated in Figure 3.

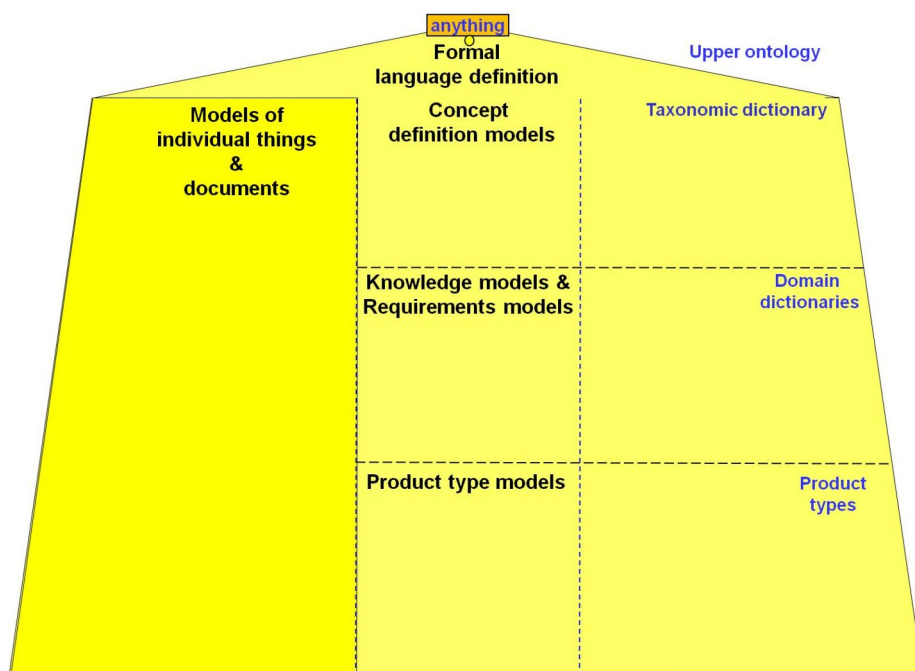


Figure 3, Semantic Information Model Architecture outline

The seven sections are:

Section 1: The formal language defining ontology

Formal languages are intended to be computer interpretable. Therefore, the languages need to be formally defined. The formal taxonomic dictionary-ontology (further called ‘the dictionary’ for short) with its various domains is intended to provide the semantic part of such a formal definition of the language and its concepts. Concepts are also defined in ordinary dictionaries and standards, but concepts in a formal dictionary are modeled and they are themselves expressed in expressions that comply with the definition of the formal language.

Formal language expressions consist of relations between things, whereas those relations are classified by standard kinds of relations, also called relation types. The formal language definition part of the architecture includes an ‘upper ontology’ that contains definition models of the standard kinds of relations. Those definition models are required to make expressions in the formal language computer interpretable. The upper ontology also contains definition models of the concepts that are used in the definition models of the kinds of relations. A definition model of a kind of relation consists of the following expressions:

- The definition of the kind of relation as being a subtype of one or more supertype kinds of relations. This is expressed as one or more relations between the kind of relation and a supertype kind of relation. This forms a taxonomy of kinds of relations.
- The specification of the kinds of roles that shall be played by role players in such a relation.
- The specification of the kinds of things that can play roles of such kinds.

The upper ontology also contains a taxonomy of subtype-supertype relations between kinds of roles and a taxonomy of kinds of things that can play such roles.

The collection of kinds of relations determines the expression capabilities of a formal language such as Formal English, Formal Dutch, etc. The upper ontology of definition models for kinds of relations forms the upper ontology section of the taxonomic dictionary. The upper ontology kinds of

relations includes also generally recognized kinds of relations as appear is several ISO and non-ISO standards⁴.

The upper ontology that defines the core of the Gellish formal languages is extensively described in the book ‘Semantic Information Modeling in Formalized Languages’ [Ref. 2]. The taxonomic dictionary, including the language defining ontology is available via the Gellish website [Ref. 3].

Section 2: Definition models – the taxonomic dictionary

The definition models section of the architecture includes the definition of concepts that are subtypes of the concepts in the upper ontology. Together they form a general purpose electronic taxonomic dictionary of the formal language. The definitions are based on various ISO, IEC standards as well as other standards⁵. Together with the ‘Gellish Syntax and Contextual Facts’ [Ref. 4] and a format, such as the Gellish Expression Format, they define the formal language. The syntax specifies the structure of expressions in the formal language. The contextual facts provide additional context that is required for the correct interpretation of the expressions.

The formal taxonomic dictionary (‘the dictionary’ for short) consists of two parts: a definitions part and a terminology part. The definitions part contains definitions of concepts (kinds of things, which kinds are represented in the formal language by unique identifiers, UIDs) in the form of definition models. The terminology part contains names, synonyms, abbreviations, codes, symbols, etc. by which the concepts can be presented to people. The dictionary also contains relations between the concepts and between the concepts and the terms.

Section 3: Knowledge models

This section consists of a collection of knowledge models in which knowledge is captured about the defined kinds of things. This knowledge is automatically integrated with definitions of the concepts that are used in the knowledge models and may be integrated with specifications, requirements, standardized designs, etc. (see below).

⁴ Especially concepts that are also defined in ISO 10303-221, ISO 15926, ISO 16354, RDF and OWL.

⁵ Especially concepts that are also defined in ISO 31, ISO 1000, ISO 1998, ISO 14224, ISO 16323, ISO 16789, IEC 50, ISO/IEC 1131-3, VDI/VDE 3696, ISO 10303-42, 50, 225, 227, NORSOK and many other standards.

Section 4: Requirements models

This section consists of requirements models that express requirements about defined kinds of things, which requirements are only valid within a particular validity context, such as a particular business or standard. For example, requirements that are expressed in standards or in best practice guides of a company and that thus are valid when such standards and guides are applicable.

Note: mixed models are also possible. Mixed models include knowledge as well as requirements, options for design solutions, etc.. They are sometimes called specification models or requirements models, but include more than just that.

Section 5: Product type models and process type models and procedures

This section consists of specialized definition models that express specifications of standardized product types and types of processes or activities. The latter are typically called procedures. For example it contains models of product types and procedures that are standardized in national, international or company standards.

Section 6: Document models

This section consists of models that express information about documents and files, including also the relations to kinds of information and kinds of documents and relations to things about which the documents contain information. Document models may also include information about graphical symbols and their coordinates on drawings as well as detailed relations with and between them and the things they symbolize.

Section 7: Product and process information models

This section of the architecture consists of coherent models about individual things and individual processes and activities. Each model of an individual things may include among others a composition hierarchy and related properties. Each component in that hierarchy and each aspect, etc. has a classification relation to a concept in the formal dictionary that classifies that component. This section has two subsection: one for models about real world individual things and another subsection for models about imaginary individual things such as design.

These seven sections represent seven categories of models. They are closely related to each other reason, because expressions of knowledge, requirements and information about individual things can only be interpreted by using

definitions of kinds of relations that belong to the upper ontology section and by using definitions of concepts that belong to the dictionary section. For example, a knowledge library, being essentially a library of knowledge models, expresses knowledge as relations between concepts. Those concepts are defined in definition models that are included in one of the domain dictionaries or in the generic section of the dictionary. Therefore, a knowledge library will necessarily interoperate with definition models in a dictionary. The same holds for requirements models. Therefore, the sections together typically form one interoperating information model, and only the dictionary section will sometimes be used in isolation.

Furthermore, an available dictionary will not always contain every concept that appears in new models. Thus the creation of e.g. knowledge models will often introduce some new concepts for which definitions are required. Those new concepts and their definition models will form extensions of an existing formal taxonomic dictionary by adding subtype-supertype relations with between new concepts and their supertype concepts. And last but not least, organizations may use proprietary terminology and dictionaries with concept definitions that specify extensions to an existing formal dictionary or even as a replacement of one or more of its domains.

Collections of knowledge models and requirements models may be developed and offered by various knowledge providers. For example, branch organizations and standardization organizations conventionally publish knowledge and requirements in textual documents, whereas they can transform them into computer interpretable models and thus make them directly usable e.g. by computer augmented design systems. The concepts in such knowledge models and requirements models should be part of the language definition of the used formal language. Therefore those concepts should be chosen from a wider used formal dictionary or should be defined as subtypes of concepts that belong to an existing domain dictionary. This means that those models should not be developed as independent or isolated models. Such use of a common formal language makes that knowledge libraries always interoperate with the language definition section in the information model architecture. Such integration or interoperation enables the semantic verification of the correctness of expressions in the models.

Similarly, product type definitions that specify standard components and should also use the language defining ontology and definitions. The product and process information models about designs (imaginary things) as well as about real individual things should also be expressed in the same formal language and may refer, especially by classification relations, to the concepts in the dictionary.

The upper ontology section with the formal language definition of the information model architecture is extensively described in the book ‘Semantic Information Modeling in Formalized Languages’. The next paragraphs describe the various other components of the information model architecture in more detail.

4.2 Phases in the information modeling process

The Gellish Modeling Methodology includes a structured approach to modeling. First it assists domain experts to model knowledge and requirements in a structured way. Secondly, it assists users and software developers in the creation of information models of individual things and of tools for their computer assisted development. The approach consists of a number of the following stages, assuming that the basic formal language is available:

1. Create concepts and their definitions and names (denotations)

The concepts (kinds of things) that are required in expressing knowledge or requirements and are not yet present in the dictionary are given a definition in the form of a definition model. Such definition models consist of one or more relations of standard types between concepts, according to a fixed pattern. Those models will extend the formal language dictionary with additional entries and make it wider applicable for expressing information.

2. Add knowledge

Knowledge about these kinds of things is documented in the form of knowledge models. This makes knowledge accessible and available for reuse in designs and in verification processes. Such knowledge models consist of collections of expressions that are composed of relations between concepts that are defined in the dictionary. Those relations are composed of expressions that use a limited set of standard relation types. All used concepts shall be selected from the language defining dictionary.

3. Add requirements

Requirements for kinds of things are added in the form of requirements models. This makes a company’s standard requirements available for reuse and inclusion in product models and procurement processes. These models also use special kinds of relation types from the Upper Ontology section of the dictionary.

4. Add product types and process types

This includes the creation of definition models of product types. Product types represent product portfolios, catalogue items and standard designs or typical configurations. For example product types that are on the list of available product types of a supplier or types of assemblies for which cost figures are available. Process types or procedures represent standard ways of performing activities and best practices. Thus their modeling makes a company's standards available for computer supported access and use.

5. Create product and process models

In this stage individual products and processes are modeled using the predefined formal language and its concept definitions. The previous stages make the creation of models of individual products and processes more efficient, by enabling the reuse of knowledge and requirements.

6. Create document models

Document models relate information objects, such as documents, drawings and files or graphical symbols to modeled things. This enables searching of documentation via the objects about which documents provide information or searching for information about objects via the symbols and text on drawings. Document models may also be created to enable the production of drawings. By including standardized symbols and nomenclature for drawings in the formal dictionary, this supports the use of those standards.

Each kind of knowledge model requires its own set of relation types. It is also possible that requirements are modeled in various ways. For example, models may contain requirements at a qualitative level, whereas it is also possible that models contain requirements that are quantitative.

4.3 Definition models

4.3.1 The taxonomic dictionary-ontology

The second layer of the Information Architecture consists of an upper ontology, which includes a formal vocabulary and taxonomic dictionary with definitions of generic concepts as well as domain dictionaries with definitions and terminology of more detailed concepts and product types. The domain dictionaries further define the concepts and vocabulary of the formal

language. The upper ontology determines the scope and expression power of the formal language.

Ordinary dictionaries usually provide for each *term* (lemma) one or more definitions. The user of the dictionary is then left with the task to determine whether those definitions are different definitions for the same concept or that the definitions describe multiple concepts. In other words ordinary dictionaries are usually not explicit in the distinction between synonyms and homonyms. Furthermore, the definitions in ordinary dictionaries are sentences in which other terms are used, but those dictionaries do not provide explicit or computer linked relations to the definitions of the concepts that are denoted by those other terms.

A taxonomic dictionary of a formal language does not just contain a list of concepts (terms) and their definitions as in conventional dictionaries, but it consists of computer interpretable *definition models* of related concepts. Such definition models are collections of formalized expressions that explicitly relate the components of the expressions to each other by computer interpretable relations in a data structure. Definition models include concepts and their names and synonyms, symbols, illustrations, codes, abbreviations, etc. as well as relations between the defined concept and related concepts. Such relations are the case for all things of such kinds. Only then the relations are defining relations.

A formal language should also enable referencing and expressing facts about detailed specialized concepts and product types. Therefore, a formal dictionary should be extensible with such detailed concepts and product types. This can be done by extending the subtype-supertype hierarchy with models of standard product types and even with such things as proprietary company specific catalogue items. Collections of such detailed kinds of things are typically included in specialized domain dictionaries. The relation between domain dictionaries and the existing public domain ones is illustrated in Figure 4.

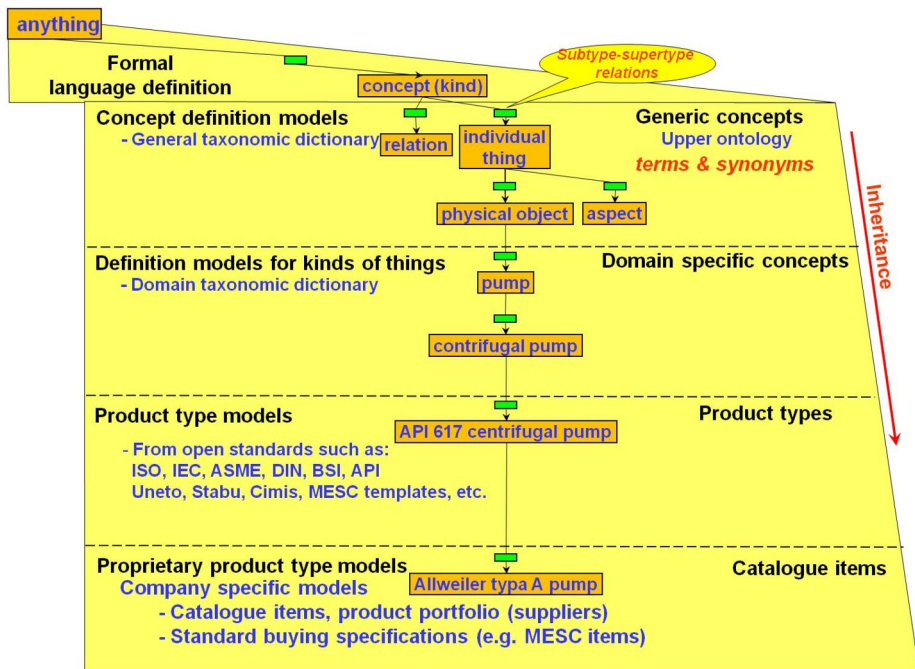


Figure 4, Domain dictionaries as extensions of a general dictionary

Examples of reasons for creating taxonomic domain dictionaries are:

- To create a dedicated electronic formal domain dictionary for a particular application system that has functionality to use inheritance in subtype-supertype hierarchies. For example, a civil technology domain dictionary for systems for designing civil items.
- To extent an existing taxonomic dictionary that is in use with definitions of additional concepts. For example in order to cover equipment types and property types that are used in a specialized environment. (The process of extending and adding synonyms is often called 'mapping').
- To create a taxonomic dictionary in another language than English.
- To harmonize systems by mapping the names and codes used in a particular context to the names of the concepts in an existing formal dictionary.

A formalized language can be defined in a special kind of ontology that has as only purpose to define the semantics of a formal language. Such a language defining ontology can then be used as a 'smart dictionary' for

expressing information and knowledge in that defined language. Such an ontology has the following characteristics:

Distinction between concepts and terms

The dictionary-ontology distinguishes between a concept and the terms with which the concept can be denoted. Therefore, it provides definitions of concepts (and not definitions of terms). Each concept is represented by a different unique identifier (UID). The concept of UIDs is further discussed in paragraph Fout: Bron van verwijzing niet gevonden. Each concept may be denoted by various terms, such as names, codes, abbreviations, etc. in different languages, but those terms are all distinguished from the concept itself and they all refer to the same concept UID. This means that a smart dictionary makes an explicit distinction between homonyms (the same term for different concepts) and allows for many alias terms (synonyms) for the same concept.

When another concept exists that is denoted by the same term (a homonym), then that other concept will (shall) have a different unique identifier (UID). In that respect a smart dictionary differs from a conventional dictionary, because a conventional dictionary is based on words (terms), whereas for each term there are given several definitions, whereas the user is left with the task to determine whether the definitions are valid for the same concept or for different concepts.

Arrangement in a taxonomy

The concepts in the dictionary are arranged according to a particular subtype-supertype hierarchy or taxonomy. That hierarchy expresses specialization-generalization relations between the concepts in the form of a hierarchical network structure. Often such a structure is called a tree-structure: a specialization tree. However, strictly speaking it is not a tree structure, because the hierarchy includes not only multiple subtypes, but occasionally also multiple supertypes.

The hierarchy can be used by computers for the inheritance of knowledge or requirements and aspects from a supertype concept to all its subtype concepts. The hierarchy can also be used to facilitate searching for things of a particular kind, because the computer can find not only all things of a specified kind, but also the things that are classified by any of the subtypes of that kind.

Every concept in the dictionary is a subtype of a more generic concept. At the very top of the hierarchy there is only one concept that covers everything and is thus called 'anything'. The application of a formal

language can have an unlimited scope as the language covers in principle everything, because the language is extensible, whereas any new concept can be defined as a subtype of a concept that already exists in the language and thus inherits the expression capabilities of all its supertype concepts.

The more specialized concepts are located lower in the hierarchy. This holds in particular for the concepts that are defined in knowledge libraries (object libraries) for specialized application domains and in various standards. At the lowest level of the hierarchy there are concepts located that represent standardized types of products, such as product catalogue items and manufacturer's models. The latter ones are also subtypes of more general concepts.

Definitions of kinds of relations

The dictionary includes not only definitions of concepts in all categories of concepts, including also definitions of relation types. Those relation types largely determine the expression capabilities of the formal language.

Definition models as well as textual definitions

Concepts are defined by Definition Models, whereas a Definition Model may contain a sentence with a textual definition, but in addition to that, a Definition Model contains explicit relations between the defined concept and other concepts. Those relations express facts that are always true for the concept. Those relations are thus used to make the definition. In that way a Definition Model consist of a collection of relations, which forms a computer interpretable network of related concepts. Each relation with another concept in a model is classified by a standard relation type, which shall be selected from the formal dictionary (the standard relation types are also defined in the Smart formal dictionary). This makes a smart dictionary an ontology. An ontology can extent these relations beyond the dictionary facts to encyclopedic knowledge about the concept. Both kinds of knowledge about the concepts (dictionary definitions and encyclopedic knowledge) can be used by computers, because it is explicitly modeled. Furthermore, such knowledge about a concept is inherited by all the subtypes of the concepts in the subtype hierarchy (taxonomy).

Unique identifiers for identification of concepts

The dictionary uses one language independent unique identifier (e.g. a natural number) for each represented concept. This enables that facts that

are expressed in one language can be automatically presented by a computer in any other language for which a smart dictionary is available.

Extensible domain taxonomic dictionaries

The dictionary can be extended by private and proprietary concepts and terms. For example, company specific codes and proprietary knowledge. Some instructions are given in 'Proper definition of a concept'.

The dictionary is computer interpretable and system independent

The definition models are expressed in the formal language itself, using only a very small 'bootstrapping' set of concepts. This is contrary to ordinary data models that use a meta language for the expression of the models.

A large number of definitions and terms that are considered international common concepts and terms are already included in the basic taxonomic dictionary. That also holds for the definitions and terms for concepts in many specialist disciplines.

A flexible formal language is an open language and anybody can extend the formal dictionary with an unlimited number of definitions of subtypes of concepts and relation types from his or her own discipline or organization on any level in the hierarchy and with any number of synonyms. However, users should be aware that the definitions of their concepts should be communicated with others. Therefore extensions should satisfy the rules of the formal language, especially also the rules for allocating UIDs. To facilitate wide interoperability it is recommended to propose extensions for inclusion in the formal dictionary in order to enable others to recognize the concepts and know their definitions.

Developers of knowledge libraries or users of the formal language can always verify whether their concepts already exist in the dictionary. If not, then they can simply add them by defining new concepts as subtypes of existing more general concepts in the dictionary. This also holds for the standardized relation types, which can be used for the effective recording of knowledge about kinds of things or for the description or specification of individual things or their designs. It also holds for the standardized allowed values for characteristics and scales (units of measure) that are included in the dictionary. The way to specify such extensions is described in chapter 10.

4.3.2 Terminology management

It is a rule in formal languages that only terms and phrases may be used that are defined in its formal dictionary. Therefore a formal dictionary shall have a terminology part that specifies the vocabulary of the formal language. The terminology specifies how concepts may be denoted or called in expressions in the formal language. As described before, those denotations include names and various kinds of aliases, such as synonyms, abbreviations, codes and symbols. For short, we will call all these denotations together ‘terms’, which act as ‘names’ for the concepts. The distinctions between various kinds of terms for concepts in the terminology part of the dictionary are facilitated by various relations of special kinds. Such relations relate a unique identifier of a concept to the terms by which it can be denoted in various language community contexts. For example, the concept represented by UID 130370 is denoted in English in a general technology language community context as ‘pump’, whereas that same UID may be denoted within a particular company X by an abbreviation as ‘P’. The specification of such naming relations can be expressed in a naming table as follows:

Language	Language community	UID of object	Name of relation type	Name of object
English	technology	130370	is called	pump
English	Company X	130370	is abbreviated as	P

Unique identifiers and their related terminology can be specified as public or proprietary extensions of a formal dictionary. The concepts that are defined in that way can be used in the same way as all other concepts that exist in the formal language. This is independent of the question whether the extensions are stored in a central database or in a separate database on another location. When such extensions are defined according to the rules of the Gellish Modeling Methodology the combination of those extensions and the general formal language defining dictionary-ontology together will behave as if it is one dictionary-ontology.

4.3.3 Synonyms, homonyms and unique identifiers

Names of concepts are associated with the concept which they denote via an naming relation. This allows for multiple names for the same kind, i.e. synonyms. It also means that a concept itself is in principle sufficiently identified by a unique identifier and can even be without a name. Therefore, it is intended that all kinds in formal languages will have a global unique identifier. These unique id’s (UIDs) are basically arbitrary numbers within

particular ranges.

The unique identifiers provide among others the capability to use homonyms (the same name for different kinds), whereas homonyms are distinguished by relating the ‘*context* for reference’, which is the context in which a homonym is primarily applicable to the relation between the object and its name. Homonyms can be distinguished in formal languages by their naming context, usually referred to as the ‘language community’ that is the home base for the term.

The prime names of kinds can be called the ‘preferred names’ in a particular language in general. Synonyms are then alternatives that are preferred in particular language communities. Names are preferably applicable in a wide language community context. Some terms are even internationally accepted in standards. Their language can be called ‘international’.

4.3.4 Multiple languages

A family of consistent formal languages provides different names in various languages and may contain codes according to various ‘naming systems’ or ‘coding conventions’, whereas multiple denotations denote the same concepts.

Any fact may thus be expressed in different languages. However if multiple expressions express the same fact, then that single fact shall have only one unique identifier. Thus the UIDs of identical facts shall be identical.

It is also possible to add translation relations that express how the name of something in one language is translated to another language.

For example:

Language	UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
English	1	network	4	1146	is a specialization of	3	system
Nederlands	1	netwerk	4	1146	is een specialisatie van	3	systeem
English	1	network	5	4691	is a translation of	1	netwerk

It is also allowed to define the reverse translation as well, as follows:

Nederlands	1	netwerk	6	4691	is een vertaling van	1	network

Two expressions express the same information (express the same ‘fact’) when they are valid at the same time according to their status and validity period (start and end date-time) and when the following UUIDs are identical:

- the UUID of the left hand object,
- the UUID of the right hand object,
- the UUID of the relation type,
- the UUID of the validity of the fact (if present),
- the UUID of the intention of the expression (if present),
- the UUID of the extent to which it is the case (if present),
- the UUID of the probability that the expression is true (if present),
- the UUID of the unit of measure for the quantification of an aspect (if present).

Two identical facts expressed in different languages implicitly define that the names of the related concepts are each other’s translations (synonyms). For example, fact 4 in the above table implicitly defines that network (English) is a translation of network (Dutch).

If an explicit translation relation is present for things that also appear in identical facts (expressed in different languages) then software should verify whether the names of the concepts in the translation relation is consistent with the names used in the corresponding identical facts.

It is recommended to define translations via a translation of the specialization relation, because it shows the taxonomy (subtype – supertype hierarchy) also in the other language.

4.3.5 Domain dictionaries and dictionary extensions

Knowledge models or requirements models that are expressed in a formal language are models that relate concepts that are defined in a formal dictionary. Therefore, software should support modelers in finding the required concepts and knowledge about them by searching with a search term in the vocabulary of the dictionary and by presenting possible alternative terms. If the concept is found, although the concept is not denoted in the dictionary by the term that is preferred by the modeler, then the modeler can decide to extend the dictionary by specifying an alias term for the concept, while maintaining the UUID. When a concept is not found in the dictionary of the formal language, then a new concept should be added, before it can be used in the model and in other models. Each model can thus initiate a requirement for extension of the dictionary with new concepts.

For example, an architect may get an order to design a type of ecoducts for the city of Frankfurt, whereas the modeler makes use of a formal German

dictionary and a particular formal extension: the Frankfurter domain dictionary. The latter contains Frankfurter local terminology as synonyms for other terminology as well as some proprietary defined concepts. It may be that the concept ‘ecoduct’ does not yet appear in either of the two, whereas new subtypes, such as ‘Ecoduct type A’ and Ecoduct type B’ are required to be added to the Frankfurter domain dictionary anyway. Then the first one could be added with a temporary UID, with a feedback proposal to add the concept to the general formal dictionary, whereas the latter two are added with a proprietary language community context.

Organizations may decide to develop a complete new proprietary or public domain dictionary, for example because they want to manage a complete set of terminology that is used in their own organization.

Extensions of the dictionary of the formal language (or even replacements of parts of it) should fit in the overall architecture. This implies e.g. that all new concepts shall be defined by using explicit specialization relations with existing concepts, in the same way as existing concepts are defined. Some new concepts will be subtypes of existing concepts, with or without themselves having further subtypes. Others need to be inserted in an existing hierarchy, so that an existing specialization relation should be replaced by new ones.

Somebody who wants to add a concept to the dictionary of a formal language and thus to extent its vocabulary usually has a short term need. Therefore, such a person usually will arrange that the concept is added on short notice to a local proprietary domain dictionary. This may be the case for the ecoduct in the above example. This can be done provisionally by allocating a temporary UID or permanently with a UID from a reserved range. For generally applicable concepts it is recommended that a proposal for addition (or modification) of concepts and/or synonyms is also issued for addition to the general formal language dictionary.

In both cases she can arrange that a new definition model is developed by the maintenance organization of his local domain dictionary by the Architect’s own organization. The difference is whether the new definition model is approved and included in a local domain dictionary only or is (also) approved and included in the international general dictionary for the formal language. In the latter case all other parties that are involved in this and in future projects that use the formal language will automatically share the common definition.

Whoever will define the concept, he or she has to ensure the quality of the definition, so that there is reasonable consensus in the domain community

about the definition. This includes that the concept is defined as a subtype of an existing supertype concept (e.g. for ecoduct this might be ‘bridge’) and shall verify whether all facts about the subtype are indeed valid when the concept ecoduct is defined as a subtype of that supertype in the specialization hierarchy of concepts. In other words: the defining party has to convince himself (and others) that the new concept (e.g. ecoduct) indeed inherits all defining aspects from its supertype concept (e.g. ‘bridge’).

Finally it is important that the defining party records the references to sources of information that are used to create the facts in the definition model.

An extensive description of this modeling method for definitions is described in chapter 10. The next paragraph (4.3.6) describes the characteristics of definition models.

4.3.6 Definition models

Formal dictionaries with computer interpretable definitions consist of definition models about the defined concepts. A Definition Model is basically a model that consists of a coherent collection of expressions of facts about the defined concept, which facts are always valid for every example of that kind in the real world. Such facts relate the defined concept to other (existing) concepts.

Definition models may be models of definitions of any kind of concept, such as:

- kinds of physical objects;
- kinds of occurrences, including kinds of activities, processes and events. For example: kinds of testing or maintenance activities;
- kinds of functions and other kinds of roles;
- kinds of aspects, such as characteristics, properties, qualities;
- qualitative aspects or allowed values for aspects of kinds;
- kinds of relations;
- etc.

All these kinds together are called concepts.

The definition of concepts in a formal dictionary is illustrated in Figure 5.

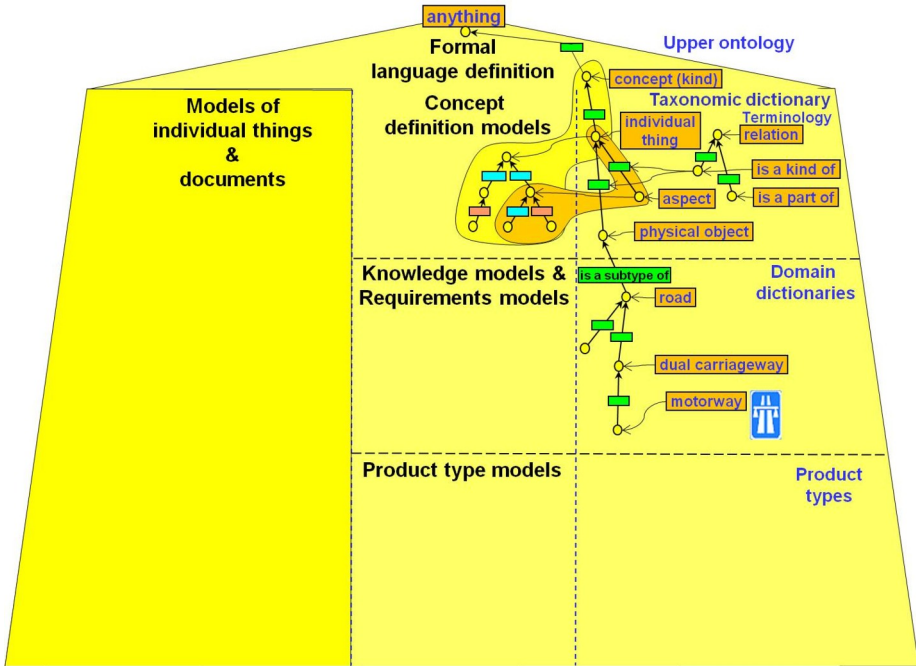


Figure 5, Architecture section 2: Definition models.

The dots (small circles) in Figure 5 represent (UIDs of) concepts and the rectangular boxes with various colors represent (UIDs of) relations of various kinds. The structure at the right hand side of Figure 5 presents a part of the taxonomy (the subtype-supertype hierarchy) of concepts as discussed above. For example the hierarchy includes an expression that states ‘physical object is a subtype of individual thing’. This is depicted in the figure as follows: the dot that represents a concept with the name ‘physical object’ is related by an <is a kind of> relation to the dot that represents a concept with the name ‘individual thing’.

The upper middle section shows a ‘cloud’ which represents a definition model that consists of a network of expressions that together define a concept. Each expression in the network expresses a (binary) relation between two concepts. Those concepts are symbolized by the dots that act as the nodes in the network. Each of the concepts in the definition model network appears also in the taxonomy. The relation between a concept in the model and the same concept in the dictionary is denoted by arrows from dots in the dictionary to dots in the model. In that way each concept in a model is a node in a branch in the taxonomy.

The example illustrates the definition model of the concept ‘individual thing’. Therefore the model includes the relation with the supertype of ‘individual thing’ called ‘concept’. Furthermore, in the definition model the concept ‘aspect’ appears as one of the related concepts. Therefore, the model that defines ‘individual thing’ implies that there exists a definition model of the concept ‘aspect’. This is illustrated by a second ‘cloud’.

The ‘Domain dictionaries’ section in Figure 5 also contains a few defined concepts, among others the concept road, dual carriageway and motorway, which are all related by subtype-supertype relations. Each of them is also defined by a definition model. The most generic concepts in a domain dictionary are related by a subtype-supertype relation to concepts in the generic taxonomic dictionary. This is required for inheriting aspects from the definition of their supertypes and inheriting in which kinds of relations they may be used.

A definition model of a concept consists of three parts:

1. A unique identifier (UID) that represents the concept in a language independent way. This is represented in the figure by a dot (a small circle). For example, the number 40153 is the UID of the concept road.
2. One or more terms, synonyms, translations, abbreviations, codes, or symbols that are related to the concept (by a naming relation) and that form various visual and/or audible representations of the concept in a particular language and/or language community. Examples in the figure are the English terms for the object types: physical object, road and the phrase for the relation type <is subtype of>, which is a phrase that is the name (in English) of a specialization relation (a subtype-supertype relation).
3. A collection of definitional facts that specifies the supertype(s) of the defined concept, that provides a textual definition and that specify the distinguishing aspects or components that the defined concept by definition has. The figure illustrates a definition model as a collection of relations between concepts that are represented by lines that connect circles via rectangles, whereas the rectangles represent relation types.

Each fact in a definition model pattern is basically build-up as a collection of expressions of binary relations.

Every definition model of a concept consists of binary relations between the defined concept and other concepts and possibly relations between those other concepts. Together those relations express what is *by definition* the case for aspects and for parts of the defined concept.

For example:

The facts that define the concept ‘viaduct’ should include only the facts that are by definition always true for any viaduct, of whatever subtype of viaduct it is a member. Thus those facts are valid for all things that may be classified as a viaduct. This means that a definition model does not cover facts that are sometimes true, under some conditions only or for some subtypes only. The latter kinds of facts are part of a knowledge model about what *can be* the case for some viaducts or of definition models of particular subtypes of viaduct.

The concepts that are defined through definition models are an extension of the taxonomic structure (specialisation tree) of concepts in the dictionary. So together with the concepts that are already defined in the dictionary they should form one consistent subtype-supertype hierarchy.

A definition model follows a typical pattern. Details of that pattern and detailed guidelines for the creation of definition models and their integration in a formal dictionary are given in chapter 10.

4.4 Knowledge models

The third section of the Information Architecture consists of a network of knowledge models. A collection of related knowledge models forms a knowledge library.

4.4.1 The expression of knowledge

A knowledge model about a particular kind of thing is a collection of statements that express something that may be the case for individual things that are of that particular kind. Thus it expresses knowledge about a kind of thing. Those statements (possible facts) are expressed in a formal language as relations between concepts that already exist in the dictionary. Thus each of the relations relates kinds of things. The meaning of a relation that relates two kinds differs from the meaning of a similar relation between two individual things. For example, the meaning of ‘an A can be a part of a B’ and ‘an A is by definition a part of a B’ differ semantically from ‘A is a part of B’, where in the first and second expression A and B must be kinds whereas the relations essentially do not express information about kinds, but they express information about individual things of those kinds. On the other hand, the third relation expresses information that is directly applicable to the individual things A and B. Therefore, a formal language shall have relation

types for expressing facts that describe knowledge, which relation types differ from relation types that specify information about individual things.

The nature of relations in knowledge models is that they express possible facts, being facts that can be the case, but that are not (by definition) always in all circumstances the case for things of those kinds.

A knowledge model can be depicted as a network of related concepts around the concept that represents the concept about which knowledge is provided. This was illustrated for example in the knowledge model of a road in Figure 2 on page 29.

In addition to expressing knowledge it is possible to express requirements and constraints that are valid for individual things that are of a particular kind, which requirements and constraints are only applicable in a particular context. For example requirements for roads, which requirements are only applicable when the road is located in a particular country or state, where the government determines the applicability of the regulations. Such categories of facts belong to models that are called requirements models. In many cases organizations tend to include requirements and specifications of standard product types in knowledge models. For example, they treat typical compositions, standard requirements and typical costs figures as 'knowledge'. However, requirements do not include generally valid knowledge about kinds of things, because requirements belong to the context dependent requirements and specifications that are discussed in the next section of the Information Architecture.

The kinds of things that are related in any knowledge model must have been defined in definition models in the taxonomic dictionary section of the Architecture, before they are used. This means that this section is concerned with knowledge that is added to existing and already defined concepts. Such additional knowledge expresses what *can be the case* and it enriches the knowledge that is contained in the definitions of the concepts, which express what *is by definition the case*. It will be clear that expressing knowledge does not extent or narrow those definitions, although it may be a trigger for improving the definitions or for the addition of new definition models.

For example:

A dictionary model expresses what a viaduct is. It contains only criteria that can be used to decide whether an object can be classified as a viaduct or not.

A knowledge model of a viaduct will contain a lot of generally valid knowledge about viaducts, but the definition of how a viaduct is defined does not change because of all that knowledge.

Thus, knowledge models build on the concepts that are defined by the definition models and they extend the networks of the definition models with encyclopedic knowledge about what is possible. Those extensions have the form of additional relations of various kinds with other already defined concepts. Those relations do not repeat relations that are included in definition models.

Each concept in a dictionary forms a node in a network. The relations of definition model networks are relation with those nodes, whereas knowledge can be attached to the same nodes by specifying additional relations with those nodes, in principle without adding new nodes. The collections of knowledge relations about the nodes form knowledge model networks. The total collection of relations forms one integrated network of relations.

A knowledge library (also called an object library) is a database of knowledge models that are integrated with definition models. Organizations may develop such knowledge libraries and offer services to extract knowledge models from the library for inclusion in application systems or as a guide to develop or design individual objects.

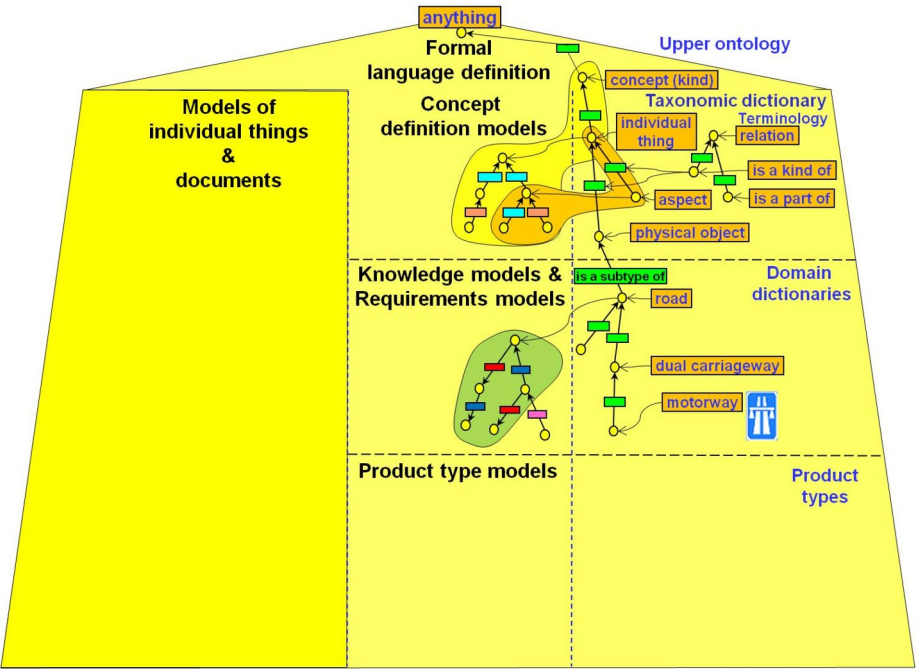


Figure 6, Architecture section 3: Knowledge models

Figure 6 presents a knowledge model, symbolized by a ‘cloud’ that consists of a collection of relations between related concepts, which include among others the concept about which the model provides information. Each knowledge model only relates concepts that are already defined in definition models and thus knowledge models always build on definition models. In other word, knowledge models form an integral whole with the definition models of the concepts that are used in the knowledge models.

The example represents a knowledge model of a road. Thus the concept road as well as the related concepts are (shall be) included in the dictionary and thus the concepts are nodes in definition models. Thus, when searching on the concept road it is likely to find relations that belong to a definition model of road, which includes its supertype(s). It is also likely to find possible subtypes, and to find relations that belong to one or more knowledge models about a road and its subtypes, such as for a motorway. In that way the knowledge about the concept road and its subtypes is disclosed.

Furthermore, each concept inherits the knowledge that is available for its supertypes. So the concept motorway inherits the knowledge about roads in general, which makes that knowledge also applicable for usage by subtypes, for example for the design of motorways.

The above statements hold for any kind of object in a knowledge library. For example, it holds for pumps, vessels, trains, vehicles, buildings, sewer systems, etc., etc. and all their components. All the subtypes of such kinds of objects have their own knowledge models and inherit the more general knowledge from their supertypes.

So, due to the taxonomic nature of a dictionary in the Gellish Modeling Methodology the knowledge that is expressed for the supertype kinds of things does not need to be captured again for each of their subtype kinds of things, whereas the knowledge (and definitions and requirements) can be found when searching on subtype concepts.

This means that the specialization hierarchy of concepts with their definitions act as connectors for knowledge models, such as coat hangers for coats, whereas the concepts can be used to select the knowledge models for usage, for example during a design process.

Each application area has its own needs for knowledge. For example, people who formulate specifications of requirements have needs for different knowledge as people who create product information models or as cost estimators. This means that various persons may need different knowledge

models about the same kind of thing. This implies that every knowledge model has a specific scope and it implies that there may exist various overlapping knowledge models about the same kind of thing. The scope of a knowledge model determines the cut by which the model can be separated from the total network of relations between concepts. An example of such a scope is defined by inclusion relations that are collectively illustrated in Fout: Bron van verwijzing niet gevonden.

Thus a particular knowledge model about some kind does not need to include all knowledge that is available about that particular kind of thing. For example, there may be different models about the same thing whereas each model contains knowledge that is collected from a different perspective. Knowledge models may also include partly the same facts, so that different networks have overlapping scopes by sharing the same relations, without duplicating those relations. Formal language enabled software will be able to integrate all models and present the combined knowledge in an integrated way. As applicable knowledge can be selected from all available knowledge about something as and when required, it is usually not required to explicitly keep record and maintain the definition of the scopes of various knowledge models explicitly in a knowledge base.

The relations in a knowledge model generally include relations of the following kinds:

- relations between concepts, one of which is the concept about which the knowledge is expressed;
- relations between concepts and information objects.

If knowledge models are explicitly represented as separate objects with their own UID (next to the concepts about which they provide knowledge), then relations of the following kinds will also be included:

- a relation between the knowledge model and the concept about which the model expresses knowledge. For example:

model X	contains information about a	pump
- relations between the model and the facts that are included in the knowledge model (the facts that belong to its scope).

When knowledge about a kind is modeled, it is still optional whether information (expressions) about individual things of such a kind or values of aspects is recorded or not. The inverse is also true: information about individual things may be modeled, without the modeling of possibilities for such information. This is because of the difference between expressed

knowledge about a kind and possible available information about individual things of such a kind. For example, in a knowledge model it might be recorded that a car can have a (some) mass. This is factual *knowledge* about what can be the case for things of a particular *kind*. However, the concept ‘car’ is neither *defined* by its mass, nor by the fact that every car has *some* mass. A kind of physical object is only *defined* by a property when the *value* of the property is fixed for every object of that kind. On the other hand, the value of the mass of an individual car may be recorded even if it is not specified on beforehand that cars can have a mass.

4.4.2 The general structure of knowledge models

A knowledge model about a kind of physical object usually specifies the kind of components from which such a kind of object can be composed or typically is composed and also what kind of properties it can have or typically has and in which kind of activities or occurrences the kind of object can be involved or typically is involved. The fact that things are typically the case implies that there are exceptions. Thus such things are then not always the case. For their modeling it is sufficient that a formal language enables to express that it can be the case, whereas capturing the knowledge for those typical cases may justify the creation and definition of definition models of product types that are subtypes of an existing concept. The relations that express knowledge about what can be the case are then applicable for the products of the supertype.

The backbone of a knowledge model typically consists of the following kinds of relations:

Conceptual composition relations.

For example:

car can have as part a wheel

Conceptual possession of aspect relations.

For example:

car can have as aspect a color

Conceptual involvement in activity relations.

For example:

car can be a subject in a collision test

Conceptual relations with documents or kinds of documents, organizations, etc.

Knowledge models can contain knowledge from different perspectives. Therefore we can distinguish for example:

Knowledge models that describe many possible compositions with a large variety of options for characteristics, possible specifications and possible requirements and constraints. Such knowledge models will be of value for trainees and experts in different disciplines. For example, for cost estimation, design, etc.

Knowledge models that specify knowledge about particular subtypes. They are of value once subtypes are already selected, but a further specification of details is required. Therefore these knowledge models are about subtypes of the above mentioned concepts and contain more details. These models can therefore be called specification models.

Domain specific knowledge models, which contain knowledge for a particular application domain. For example, reference models for cost estimation. Such models may contain for example knowledge about the kinds of aspects that are relevant for the estimation of the costs involved when an object of such a type is designed, manufactured, installed, etc.

Other examples are function oriented models and activity or process models.

A knowledge model library may also distinguish between models that are specifically intended for particular departments (organization specific model), models that are discipline specific and models that are project specific.

This means that models may become very specific and its applicability may be limited to a particular context within which the model is made. On the other hand it is also possible to integrate all information and to select only the required information from the search results. Therefore it is important to pay special attention to the management of models in a model library.

4.4.3 Composition trees

Typical or possible compositions of kinds of things are often presented as a tree structure: a composition tree. Such a composition tree (also called a composition hierarchy or breakdown structure) is fundamentally different from a specialization tree (a subtype-supertype hierarchy as described above where the definition model was discussed).

A composition hierarchy describes the kinds of components from which a kind of thing can be composed and from which sub-components of those components can be composed, etc.

An example of a composition tree of knowledge about a potable water system will include only conceptual composition relations (can have as part a). The following tree structure presents such a hierarchy, implicitly assuming that it represents part-whole relations:

potable water system

- water treatment plant
 - filter
 - water storage tank
- water distribution system
 - pump station
 - pipe line

This composition hierarchy can be expressed formally as follows:

potable water system	can have as part a	water treatment plant
water treatment plant	can have as part a	filter
water treatment plant	can have as part a	water storage tank
potable water system	can have as part a	water distribution system
water distribution system	can have as part a	pump station
water distribution system	can have as part a	pipe line

A composition tree can be detailed to any level, even including parts of nuts and bolts, etc. A high or low position in a composition tree does not imply a high or low position in a specialization tree (the taxonomy). Both hierarchies are independent! Knowledge modeling is further discussed in chapter 11.

4.5 Requirements models

The fourth section of the Information Architecture deals with requirements models.

Requirements models are also models about kinds of things, just as knowledge models, because they express requirements that are applicable for things of a particular kind. Traditionally requirements about kinds of things are recorded in documents, first in paper documents, later in electronic documents.

Note that models that express designs for individual things specify imaginary individual things according to which specification one or more real individual things (sometimes prototypes and series or mass production products) should be fabricated. In some sense such models of individual things specify requirements for the real things that should be fabricated. Because of the way such models are expressed, they are not called requirements models, but product or process models of individual things.

Requirements models are models that do not specify what *can be* the case (as is done in knowledge models), but they specify what *shall be* the case for things of a specified kind. Not what shall be the case for all possible things of such a kind, but only for those of which the requirements relations appear in a context, such as a project or an organization, where the requirements are declared to be applicable. Such a context is called the validity context or the domain of discourse.

Requirements about *kinds* of things are applicable for all things of such a kind within the validity context. Such requirements should be distinguished from requirements for *individual* things that are expressed per individual thing in their respective product information models. Requirements for individual things, such as designs for new or modified products or designs for prototypes, are described in par. 4.8.

Requirements regarding individual things of particular kinds (also called ‘requirements about kinds of things’) can be expressed in a formal language as computer interpretable models around those kinds. Each requirements model consists of a collection of requirements about things of a particular kind for which such facts are required to be the case in a particular context. Requirements are expressed as relations between a kind and related things, whereas those related things typically are also kinds of things. However, basically related things may be anything, including also documents or parts of documents that contain requirements in textual and graphical form.

In other words, the requirements in a requirements model are applicable for each individual thing that is classified by the kind of thing for which the model is made.

There may be more than one collection of requirements applicable for things of the same kind. For example, for a particular kind of physical object there may be a collection of requirements that are only applicable in a particular context, whereas another collection of requirements is applicable in another context. Each collection of requirements forms a requirements model for such a kind of thing and each requirement is applicable within its ‘validity context’.

The way in which a requirements model is created is very similar to the way in which a knowledge model is created. Thus, a requirements model consists of a similar ‘cloud’ of relations as for knowledge models. The main difference between them is the use of other relation types. And requirements models require the specification of a validity context, whereas for knowledge models that is optional and rather uncommon. An example of a requirement

is the fact that in a particular validity context it holds that pipes of a particular type A shall be made of stainless steel. This is formally expressed as follows:

pipe type A shall be made of stainless steel *in context A*

Note that pipes of the same type may nevertheless be made of other substances in other contexts. Note that such a specification may be in conflict with the definition of the type that is referenced, although the formal language enables the verification of the consistency. Thus, if the (standard) product type definition model for pipe type A would specify by definition another construction material, then the software should give a warning message.

Another example of a requirements model is a collection of requirements that a government has published for motorways. To enable that such requirements models are directly used by various systems, the government might have expressed the requirements in a formal language and may have published the requirements models in a requirements model library on a website for usage via Internet by any of its supplier's application systems.

Yet another example is a requirements model of a road, which is a collection of requirements that are applicable for every object that is classified as a road (or as a subtype of road!), whereas those requirements are applicable when the standard in which the requirements are formulated is applicable. Then that standard is the 'validity context'. An example of a requirements model of a road is graphically presented in Figure 7.

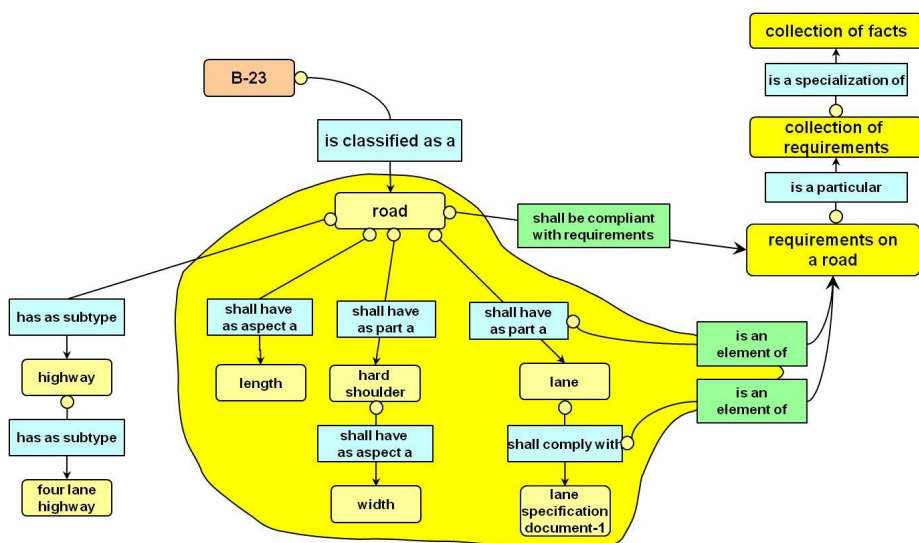


Figure 7, Example of a requirements model for a road

In the example of Figure 7, the object on the right hand side, called ‘requirements on a road’ represents a requirements model for a road. The model is defined as a collection of facts, as is indicated by its qualification relation with collection of requirements, which is a collection of facts. There can be multiple requirements models for the same concept.

It may remain implicit which expressions belong to the collection of requirements that form the scope of a requirements model. But in the example of Figure 7 the scope is indicated explicitly by two examples of <is an element of> relations between an included relation in the ‘cloud’ of relations that are included in the collection. The others should have relations of the same kind. For example, the requirement that “every road <shall have as part a> lane” is one of the facts that is an element of the collection. This means that it is one of the requirements in this requirements model on a road.

On the left hand side of the figure a few subtypes of road are defined. Those facts do not belong to the requirements model, because they are not included via a <is an element of> relation. But because they are subtypes of road they inherit the requirements.

At the top of the figure individual object B-23 is classified as a road. This means that the requirements model is applicable for that individual road. If that road was classified for example as a highway (a subtype on the left hand side of the figure), then the requirements model was still applicable for B-23 (by inheritance).

Another example of a requirements model is given in Figure 8. That requirements model illustrates that a model may consist of a data section and a documents section.

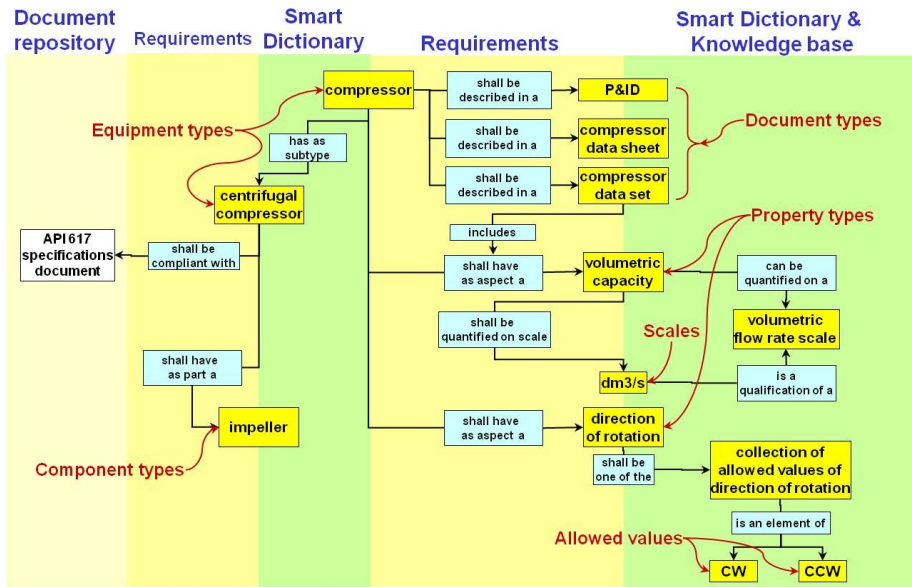


Figure 8, Requirements model of a compressor system

A *data section* consists of requirements about a concept (a kind of thing, such as a compressor or one of its subtypes) that are expressed as relations between the concept and the kinds of components or kinds of aspects that it (and its components) shall have. For example, a centrifugal compressor <shall have as part a> impeller. Requirements in the data section may also be expressed as relations between the concept and the document types on which objects of the kind shall be documented. For example, a compressor <shall be described on a> P&ID.

The *documents section* of the model consists of requirements that are expressed as relations between a concept (a kind of thing) and particular documents that express requirements in textual form, such as the API 617 standard specification document.

It may be that some or all requirements are only applicable within a particular context. That is not shown in Figure 8, but can be included in the model by specifying a 'validity context' for each fact.

The concepts that are included in a requirements model shall exist already in the formal language dictionary and shall have definitions; otherwise they shall be added according to the rules for the creation of definition models, as are described in chapter 10. Requirements modeling is further discussed in chapter 12.

4.6 Product type models

Product type models are models that reflect a standardized definition for a specialized kind of thing. This means that they are in fact definition models for such specialized kinds. For example, standardization organizations have published standards for a large number of standardized components. For example, standard nuts and bolts, standard structural steel beams, piping components, etc., etc. Also manufacturers publish their standard catalogue items, which are types of products that comply with a standard specification that is defined by the manufacturer. When such standards are applicable it means that a delivered product that is specified to comply with the standard shall have the aspects that are defined in the standard (within an agreed tolerance).

Product type models may be defined within organizations or projects to specify typical solutions and best practices or buying specification. Note that such product type models don't define all aspects of the defined subtypes. They may contain explicitly defined options or ranges and usually also leave various options completely unspecified and open.

For example, a standardized type of pipe that is made of stainless steel, has a diameter of 6 inch and has a number of other standard specifications is a subtype of the general concept pipe. That subtype inherits the general knowledge about pipe, so that a computer can point to other information that is also applicable for the specialized subtype. For example, the fact that a pipe can be represented graphically by a hollow cylinder is inherited by all subtypes of pipe.

Figure 9 illustrates how a product type models relate to the concepts in a dictionary and to knowledge models. The example is a product type model for a 'motorway type A'.

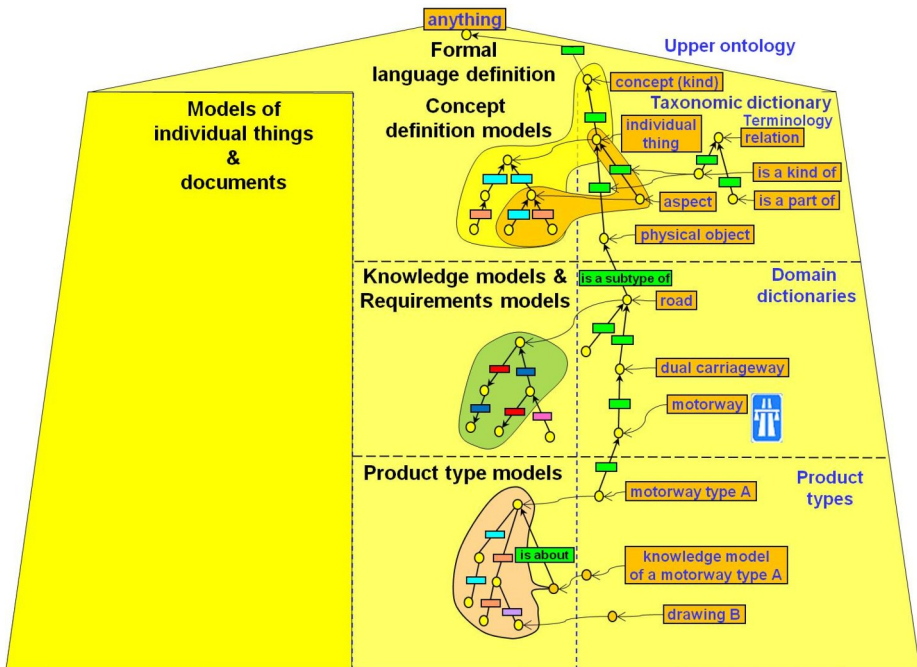


Figure 9, Architecture section 4: Product type models

Because a ‘motorway type A’ is a sub-subtype of road, it inherits the definition of a road and it also inherits the knowledge about roads.

Product type models are typically defined by having a particular composition and specific aspect values. In addition to their definition it is possible to specify required and possible compositions and aspect values for them. In other words, <is by definition> and <has by definition> relations in definition models of product types may be accompanied by <can be> and <can have> relations in knowledge models and <shall be> and <shall have> relations in requirements models for the same product type.

Usually product type models reflect typical or standard design solutions or best practices. They may express definitions product types that are only defined by a particular organization for its own use or may be defined in a particular standards organization’s standard that defined the subtype product for general use.

Product type definition models are valid in any context where a product of that type is applied. However, it may also be the case that specifications are only valid for products of a kind when the product is applied in a particular kind of role or function. An example of the latter is when viaducts are

distinguished because some rules apply when a viaduct is managed by a local government, whereas other rules apply when a viaduct of the same type is managed by the central government. In this case there are no really different types of viaducts, because for the same viaduct the management may change. This means that the difference is determined by the relation between the viaduct and the kind of managing party, whereas it remains the same type of viaduct. This means that some rules apply for a viaduct in the role of being managed by a local government, whereas other rules apply for a viaduct that is managed by a central party. In this case there are no subtypes of viaduct, but two kinds of roles: locally managed viaduct and centrally managed viaduct. Note that the rules are specified for the kinds of roles and applied to the role players.

Product type models can be highly specialized and nevertheless they may still allow for variations and options. For example, a Volvo S40 and a V70 are types of cars, which models are specified to a high degree, but nevertheless there are still variations and options available.

Product type models can be used several times to derive individual objects from the specifications. The purpose of defining such a product type and thus for specifying such a kind of model is usually for reusing the model for the design and creation of individual objects of that kind. This purpose may be for application within a particular project, for application within a company or within a whole branch of industry.

Example application domains

Product type models are valuable in many application domains such as the application domains that are described below.

During investment projects designs are made that specify imaginary individual objects and their operations, whereas during fabrication and construction real world individual objects are created. It is often considered valuable to reuse knowledge that is gained during such projects. For example, some designs may be applied several times in the same project or in other projects. This makes it attractive to convert such designs (standard specifications) into company specific standard product type models and to include them in project or company specific libraries.

Product type models may also be developed in a wider context by industries and industry sectors and published as industry standards or national or international standards. Examples are standards for nuts, bolts, pipes, cables, etc. etc.

Another kind of standard specifications consists of specifications for the types of products that companies offer on the market. Those companies thus specify a portfolio of product types according to their internal standards. The product portfolio may be collected in product catalogues, which are also a kind of library of product types.

Yet another kind of product types is formed by standard buying specification of companies that specify aspects of kinds of products and components that are bought by those companies. Companies may develop such standard buying specifications for product types in order to standardize on the use of particular types of products and thus to reduce their variety of products in stock and to reduce maintenance costs.

An example of a product type model is a model of type of viaduct V-1 that is applied several times on the M1 and other motorways in the UK. For example, viaduct 26 on the M1 is classified as a V-1 viaduct. In addition to the general model that contains specifications for a V-1 viaduct, there may be several sub-models created for a V-1 viaduct, for example one about the construction (the composition), one about functional requirements and one about cost factors.

Product type models are further discussed in chapter 13.

4.7 Document models and 3D models

This section of the Information Architecture is about document models.

Documents as well as information models are ‘information objects’ that can have various kinds of presentation or storage formats. For example a drawing can be created in AutoCAD as a dwg-file, and the same drawing can be presented in a pdf-file and on paper. And text-file can be created in Word as a doc-file whereas a copy can be stored in a pdf formatted file. This illustrates why it is important to distinguish between information (documents as information objects) and various kinds of physical carriers of such information. Therefore a dictionary of document types should contain kinds of information, such as for example: user manual, specification, drawing, schema, 3-D model, knowledge model, etc. and the dictionary should also contain kinds of information carriers, such as for example: book, report, pdf-file, doc-file, MS-Access database, XML-file or even table in xls-format.

Each document model should include expressions about a document itself, such as its title, author, publication date, etc. as well as relations with other documents including other versions as well as relations with physical files and storage locations and relations with objects about which the documents

contain information. For example, a drawing model or 3D model may contain relations between the drawing or 3D model as a whole and the things that are presented on the drawing or in the 3D model. Document models of drawings and 3D models may also contain relations between the drawings or 3D models and graphical symbols of various kinds including shape representations in 2D or 3D, possibly together with their position and orientation coordinates, as well as relations between those symbols and the objects that they represent. These relations enable the use of drawings and 3D models as user interface for searching databases and they enable keeping drawings and database content consistent. A full detailed specification in a formal language may be suitable for generating physical drawings or views on 3D models on a screen or on paper.

The consequence of the distinction between documents as information objects and documents as physical objects is that in an information model the objects should first be related to a the information (document) and then that information should be related to one or more physical information carriers (typically physical files) on which the information is physically encoded. Then the information shall be defined and the physical file shall be classified. For example a document model about specification ABC might contain the following expressions:

road type ABC	shall be compliant with specification doc ABC
specification doc ABC	is presented on ABC.pdf
specification doc ABC	is defined as a specification
ABC.pdf	is classified as a pdf file

Note that a document model includes objects and kinds that are not documents, so that it becomes integrated with other models and with the dictionary. This enables the use of the models and the taxonomy hierarchy to find documents. Document models are discussed in details in par. 5.3, and 2D and 3D models are discussed in chapter 9.

4.8 Product and process information models

The last section of the Information Architecture describes the creation of Product and Process Information Models. They are information models that consist of expressions of ideas and facts about *individual* things, such as the M1, the Eiffel tower, the Queen Elizabeth or the Pernis Refinery or about flight MH17. Thus they are about individual people, buildings, roads, process plants, ships, airplanes, systems and their components and individual activities and processes in and about them. When a product is a building, then the model can also be called a Building Information Model, when the product

is a process plant then the model can be called a Plant Information Model, etc. The term product here thus denotes any individual (physical) thing being natural objects or artifacts that result from manufacturing and construction activities that typically are intended to comply with imaginary (design) artifacts.

There are two main categories of Information Models: models about imaginary things and models about (observable) real world things. A model that contains design information is an information model about an imaginary individual object, because a designed object only exists in people's minds. A model can also be about a fabricated and constructed physical object. Then the information in the model consists of observed or measured information. Product information models may also consist of combinations of design information and real world information. For example, a designed pump may be referenced by the tag name P-1301. After construction and during operations there may be several pumps installed over time on that location, for example initially pump ABC-123 of manufacturer A was installed, which pump was later replaced by pump DEF-234 from manufacturer B. This illustrates that in this case we have information about three different objects. An Information Model that contains both kinds of information will include relations between those objects. For example:

ABC 123 is installed for P-1301

A process information model is an information model about individual occurrences, including activities and events. When activities or processes are modeled, then the models can be called business process models or physical/chemical process models or as activity models, which may include e.g. plans and schedules.

4.8.1 Modeling of information about individual things

Individual things in Information Models about individual things are related by classification relations to the concepts in the dictionary and at the same time to the knowledge models, requirements models and product type models about those concepts. This is illustrated in Figure 10.

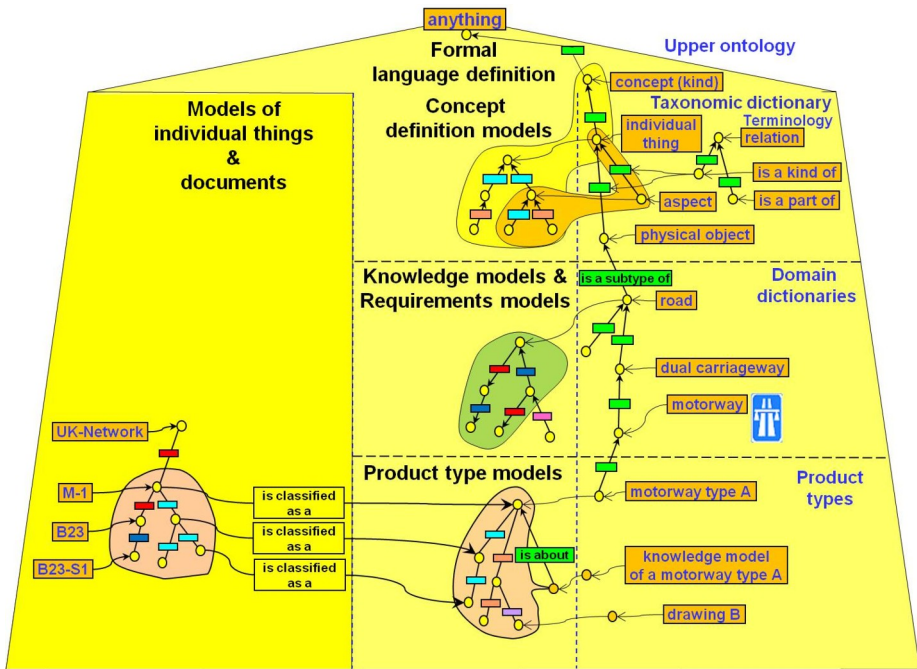


Figure 10, Architecture section 6: Models of individual things

A specification of information about individual things such as motorway M-1 or about pieces of equipment, such as pump P-1301 or the P-1301 pump system, results in Product Information Models or System Information Models. Investment projects usually specify designs of complete facilities that consist of assemblies and arrangements of several or many pieces of equipment and systems that are integrated in one big whole. By specifying composition relations in one composition hierarchy all partial information models can be integrated in one product information model for a whole facility.

An example of an information model about a facility is the following: Assume that during a capital investment project an information model is built that contains part or all of the information about an individual facility that is designed and also contains measured data about the equipment that is installed. Such a project model then contains facts about individual objects. For example, the fact that road B23 has a length of 23.4 km or the facts that road B23 is described on drawing T-12345, which drawing is presented on file F-123.dwg and as well as on F-123.pdf. In tabular form:

B23	is classified as a	road

B23	has as aspect	L23	
L23	is classified as a	length	
L23	has on scale a value of	23.4	km
B23	is described on	T-12345	
T-12345	is presented on	F-123.dwg	
T-12345	is presented on	F-123.pdf	

The above information model means that a search for B23 will result primarily in three relations: a relation that classifies B23 as a road, a relation to the length L23 of the road and a relation to the drawing T-12345. Software that reads this model should guide the further search for additional information and thus should determine the scope of the found and presented model part. For example, the software should also present the length 23.4 km and the fact that the drawing points to two files. On request, the software may launch an application to visualize the content of either one or the other file on a screen or print the drawing on paper.

An example of application software that searches in such a database is given in par. 6.

To achieve that the various parties that participate in a project use unambiguous terminology or at least implement mappings to common terminology, it is important that in the interface between their systems the same language is used. That can be achieved by specifying that systems shall be able to export information that is expressed in a formal language, which means primarily that all individual objects and all their aspects and relations shall be classified by concepts in a formal dictionary or a possible company specific or domain specific extension of that dictionary. As a result other party's systems that use the same dictionary and the same formal language can import and interpret such information. Then searches by the receiving parties for things that are classified by concepts from the formal dictionary will be successful, even if their systems internally use different names for those concepts. For example searching for things that are classified as a vessel (using its UID), will result in a list of all vessels including vessels that are classified as being a subtype of vessel. The required vessel can then be selected from the list. Even the same logic can be applied to answer queries as is illustrated by the example in par. 6.

The formal dictionary and the classification relations that relate the individual objects to the concepts in the dictionary are therefore essential parts of any product and process information model.

The usage of standardized allowed values for characteristics and scales (units of measure) is also of high importance for the unambiguous exchange of

information about individual objects and for the integration of data from different sources. Therefore they are also part of the scope of the taxonomic dictionary.

By classifying individual objects in an information model the individual objects are related to concepts about which knowledge models and product type models and requirements models may be available. Then the models about the individual objects can be derived from those models, or can be verified on their compliance with those models, or individual models can be used as a basis for generalization to become standard product type within projects or companies.

For example, if a local government uses the specification of a sewage as the basis for a standard solution of sewages, then the product model of that individual sewage can be generalized to become a product type model for a subtype of sewage, for example standardized as sewage type A. That new product type model may then be included in a model library as a standard solution for wider usage in other projects.

4.9 Relations between models

After it is defined what the natures of kinds of things are, we can build on that by adding additional knowledge. Thus a knowledge model is not a separate model next to a definition model, but it is an extension of a definition model. Similarly, a requirements model or standard specification is an extension of a definition model. Therefore, the collection of facts (network) that make up a definition model can be explicitly united in a knowledge model as well as in a requirements model or standard specification model. This is illustrated in Figure 11.

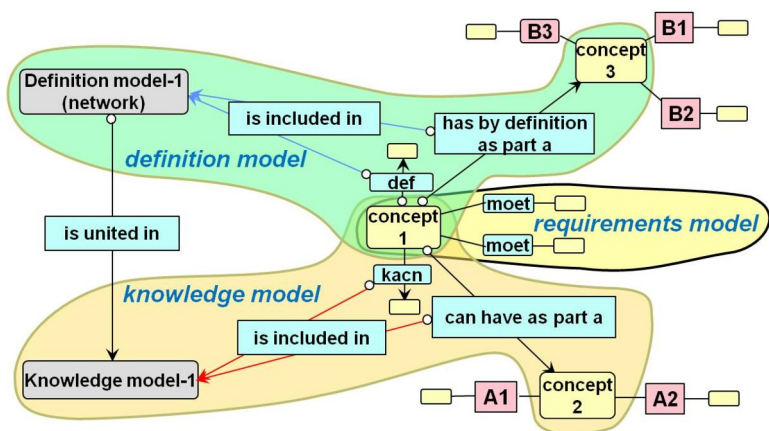


Figure 11, Relations between definition models, knowledge models and requirements models

In Figure 11 two definition relations are explicitly included in Definition model-1, which is a collection (network) of facts. Similarly there are two possibilities (can be relations) explicitly included in Knowledge model-1, whereas the requirements model remains implicit. The scopes of the models show that concept-1 is an element of all three models, whereas concept-2 and concept-3 are conceptually parts of concept-1. They are elements of respectively the definition model and the knowledge model, although their definitions and further knowledge or requirements about them (facts A1, A2, B1, etc.) are not directly elements of those models. This enables that models of parts are independent of models of assemblies.

Document models that contain information about kinds of things include relations between the document(s) and the kind (concept). In that way they also share concepts with the other kinds of models. When a document provides information about one or more individual things, then the document model is integrated in a similar way with a product or process model.

Product and process models, being models of individual things include classification relations between the individual things and concepts. Thus via those classifying concepts, the product and process models share concepts with the other kinds of models.

The above description of the relation between models from the perspective of a shared concept is an alternative view on the information architecture that emphasizes how the various models form an integrated whole information model.

The various models don't need to be centralized in order to behave as an integrated whole, but they might be distributed over various databases. The consistency can be managed via the controlled application of a common formal language. This should ensure that the objects that appear in different models are represented in each of the databases by the same unique identifier. It requires that the creation and termination of identities and naming of concepts and individual things is coordinated across systems. Figure 11 illustrates that managing the formal language definition and common language usage, with its formal dictionary and intrinsic coordination of identities, enables that the union of the various models can be treated as if they were one integrated whole information model.

4.10 Usage of modeled knowledge and requirements

4.10.1 Knowledge and requirements consultation

Knowledge models and requirements models are consulted when users, usually via an application, search for either a generic or a detailed concept or a standardized kind of product or even a manufacturer's model in order to know whether there is knowledge available about things of such a kind or whether there are requirements specified by some party for things of such a kind.

The knowledge in a knowledge library can be queried for expressions of knowledge and requirements. The results may include knowledge and requirements that are inherited from the supertypes of the consulted concept. It may also include references to subtype concepts, for which additional knowledge and requirements may be available.

Models that are expressed in a formal language can be imported directly in systems of various kinds. For example they can be imported and used in systems that are intended for knowledge aided design by generating proposals for design solutions or for the verification of designs. They can also be imported in systems with product specifications, such as deliverables of projects where they can be used for the verification whether those products and processes satisfy the requirements or for inspection or maintenance of real objects.

Furthermore, the knowledge in a knowledge library can also be used directly by users, varying from trainees to experts, for education or for manual consultation. This is illustrated by the following two examples of a designer and a cost estimator.

Usage of modeled knowledge is illustrated by a description of applications in chapter 6 Application system usage examples.

4.10.2 Example usage by a designer

A designer who is involved for example in the design of a particular (individual) steel bridge has to know which general requirements are applicable for the quality of a steel bridge. One of the aspects he has to take into account is the lifecycle cost of the bridge. Another aspect is that the material from which the bridge will be made should be compliant with specified rules for Reliability, Availability, Maintainability and Safety, for example according to the RAMS model.

Assume that he has to ensure that the costs of the design remain within an agreed range. He might save on kind of material, but questions whether some corrosion is allowed for a steel bridge. This means that he wants to search for knowledge, requirements and risks about a 'steel bridge' and especially about the process 'corrosion'.

Assume that he discovers that corrosion is not allowed at all on a steel bridge. Then as a next search he may look for material or protection material that provides sufficient protection against corrosion. In other words he may search for materials that satisfy the specified requirements.

But assume that some corrosion is allowed on a steel bridge. Then the designer should know what is meant with 'some corrosion' or a 'reasonable amount of corrosion'. He might need pictures that are collected in his company's knowledge library under the heading 'degradation of bridges'. Then a knowledge library that is designed to support such a design process will contain a knowledge models about the concept 'bridge' in which pictures of various kinds of degradation are included and thus are readily available for the designer.

4.10.3 Example usage by a cost estimator

A cost estimator who is involved in an early phase of a design of the above mentioned steel bridge will ask himself: what are the costs of such a steel bridge?

The first thing he (or she) should know is: what is the precise definition of a steel bridge. The cost estimator will first use a particular piece of software to consult the domain dictionary for that.

From that he understands that a steel bridge is by definition a bridge of which the deck is hanging on a steel arc.

The next things he should know are:

- Which design options are available for a steel bridge
- What the components are from which a steel bridge is composed.
- What the properties and specifications are that are applicable for the whole and for its parts.

Finally he has to determine what the costs per components of the assembly are.

In de knowledge library that is developed to support such a cost estimator he will find a knowledge model that is specifically created as a cost reference model. Such a model has a generic part with facts about kinds of solutions (types of steel bridges) and facts about typical compositions for those kinds of solutions (kinds of components from which those kinds of steel bridges are composed), whereas those collections of facts are specifically created for cost estimation and budgeting.

This will result in a survey of costs for various kinds of steel bridges.

Only after that he will be able to create a cost estimate for a steel bridge that is created according to some design solution.

5 Product Information Modeling

Modeled knowledge and requirements is primarily intended to be used for the creation or verification of models about individual things. Therefore knowledge models and requirements models shall reflect the structure of models about individual things. That is the reason why we first discuss in more detail the modeling process of individual things, the left hand section of the Information Model Architecture. The process of modeling definitions, knowledge and requirements is discussed in later chapters.

The scope of a Product Information Model is in principle unlimited, but in practice it will depend on the available information, the phase in the (project) lifecycle of the modeled object(s) and products and the application purpose, whereas for larger projects, multiple partial information models may be integrated into one composed information model.

Project preparation

The process to create or extend electronic Product Information Models should be preceded by ensuring the availability of an electronic definition of a formal language, including a taxonomic dictionary and possibly modeled requirements for documents and data sets that shall be produced during a project. Such information can be accessible for example by loading such data and documents in the database of application software. The model creation process shall also be preceded by the specification or adoption of a naming and coding convention (a ‘coding system’) as a standard for naming individual things.

The creation of the information model may start from scratch or may start for example with the collection of source material such as equipment lists, project plans and activity schedules as well as available documents and drawings.

5.1 The product model creation process

The process to create or extent a Product Information Model can be subdivided in a number of steps. Each step is described in further detail in the following paragraphs.

Step 1: Create or extent a product composition hierarchy.

This means: creating a number of formal expressions that specify the decomposition structure of the product(s). This is also called the

creation of a breakdown structure.

This includes the following actions:

Step 1.1: Compose the product(s)

The creation of a breakdown structure implies: decomposing the object(s) in parts and decomposing the parts into parts of parts, etcetera and specifying for each composed whole the components from which it is composed. This may include the definition of systems and their inclusion in the composition hierarchy. It may also include specifying of which assembly an object is a part, thus integrating new objects in a larger systems or assemblies. A breakdown structure is usually terminated with components that are ordered as a whole and that are often classified by selected types of objects in supplier catalogues. However the suppliers of components will usually further decompose those parts up to the level of components that are delivered by their sub-suppliers, or may specify spare parts for components of the assemblies that they sell. At the end of that supply chain the components are manufactured from raw materials, which may be solid or fluid minerals, which still may be mixtures of components in various concentrations.

Step 1.2: Adopt a coding system and naming convention.

This means making names, such as tag names, and unit names of individual things consistent and compliant with a standardized naming convention and coding system.

Step 1.3: Classify object(s) and their parts

Classification is the specification of relations between individual things and concepts (kinds) that classify them and that are selected from the formal dictionary or its extension. The classification of a whole product and of all its components means that the parts of the product, such as units, systems, etc. are classified, as well as the parts of the parts up to the required level of detail.

Step 2: Create document and file meta-data.

This means: create formal expressions that classify documents about the object and its components and relate them to physical files in directories or paper copies. Documents consist of information that can be expressed in some physical form, for example as ink on paper or as electronic file in some format, such as in pdf format or doc format or both. To define a document it should be classified and be

related to one or more physical forms. Those physical objects may be collected in collections, such as in binders and/or directories. This includes the following actions:

Step 2.1 Specify meta-data for the documents (the content of the files).

Step 2.1.1 Make document titles consistent with rules for document titles.

Step 2.1.2 Classify documents.

Classify each document by a standard document type from the formal dictionary.

Step 2.1.3 Decompose documents.

For applicable documents, such as binders, multi-volume documents and multi-page drawings: specify a decomposition.

Step 2.1.4 Specify version relations between documents.

Documents that are related, such as documents that are succeeded by new versions: relate the document to their successor.

Step 2.2 Relate documents to products, area's or processes.

Specify for each document about which products, areas or processes it contains information or specify for each product, area or process on which documents it appears. This enables that documents can be searched via product breakdown structures.

Step 2.3 Relate documents to files at locations.

Specify for each document on which a physical medium it is presented, either as an electronic data file or on paper, microfilm or any other medium.

Step 3: Specify activities and processes (functions) of/in product components.

When applicable, specify occurrences, being either processes or activities or events and the relations with the people, products or components that are involved in them, either as performer, enabler, operator, etc., or being subjected to them. Such occurrences can be decomposed and put in sequences as networks of occurrences. Physical objects or persons will be involved in the occurrences in various roles, such as involvements as performer of a process, as being subjected to an activity or event or in any other role. This implies that the occurrence or kind of occurrence shall be defined and classified and that the involvement relation shall be specified. As part of the specification of processes it is also possible to specify fluids (streams and batches) and their roles in the processes, usually as input or output.

Step 4: Specify aspects (properties and qualities) of product components and products.

Specify qualitative and quantitative aspects of the product (solids and fluids) and their components and possibly aspects of the activities, processes and products. This includes dimensions and 3D shapes.

Step 5: Specify fabricated items and relate them to designed items.

When fabricated items are produced and installed according to a design, then specify for each installed item for which design item (tagged item) it is installed. When in the course of time the installed items are replaced by other installed items, the new installed items shall be related to the same design items. The individual fabricated items as well as the imaginary design items should be classified. The classification of items can be for example by the manufacturer's model or by the type of product.

Step 6: Express other information

Finally other expressions can be added, such as information about people and their roles in occurrences, connections between items, etc.

The above steps are discussed in further detail in the following paragraphs.

5.2 Step 1, Create product composition hierarchies

The first step in creating an Information Model is creating expressions that specify how the products are composed. Such a structure is illustrated by the blue part in the left hand section of Figure 1 (page 16).

Step 1.1 - Compose the products

For each component in the products it shall be specified of which assembly (or assemblies) it is a part. When this is done in a consistent way, then a complete product breakdown structure is created. This shall result in expressions that can be added to Expression Tables. The core of such a table is illustrated by the following example:

UID of Fact	UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
208,000,080	108,000,062	Air Compressor System	1,190	is a part of	108,000,156	Unit-1300
208,000,307	108,000,156	Unit-1300	1,190	is a part of	108,000,001	LNG plant
208,000,002	108,000,001	LNG plant	1,190	is a part of	108,000,000	LNG site

The example illustrates that the same object that appears on the left hand side on a line with a role as part in a part-whole relation may appear on another line on the right hand side with a role as a whole in another part-whole relation.

Note that the lines may be included in any sequence. The sequence has no impact on the meaning. The facts may also be expressed by using inverse relations as is illustrated in the following table.

UID of Fact	UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
208,000,080	108,000,156	Unit-1300	1,190	has as part	108,000,062	Air Compressor System
208,000,307	108,000,001	LNG plant	1,190	has as part	108,000,156	Unit-1300
208,000,002	108,000,000	LNG site	1,190	has as part	108,000,001	LNG plant

When such a composition hierarchy is created it enables to navigate through modeled facilities and products for finding data and documents in various ways. For example, a particular component can be found either by searching on its name, or via its kind (which will result in a list of things of such a kind) or via one of its higher level assemblies or via one of its components.

It is recommended to identify not only pieces of equipment, but also equipment systems, being assemblies of one or more pieces of main equipment, together other system components, such as drivers, gearboxes, and other auxiliary equipment, instruments and piping. If no equipment systems would be defined, then such items cannot easily be arranged in a composition hierarchy. Note that the main equipment items are a part of their

respective systems. Those systems shall be included in the composition hierarchy. Each system shall be classified and shall be given its own UID.

Note: A recommended convention for naming of systems is to use a concatenation of the tag name of the main equipment item, a space and the term 'System'. For example: P-1265 System is a system with pump P-1265 as its main equipment item.

To facilitate searching it is recommended to consider adopting the convention of including more than one composition relation with a higher level assembly. For example, a piece of equipment may be specified as being part of an equipment system as well as being a part of a process unit, whereas the equipment system is also specified to be a part of the process unit. For example, this would mean that for P-1265 all following three facts are included in a composition hierarchy:

P-1265	is a part of	P-1265 System	
P-1265	is a part of	Boiler unit	(strictly superfluous)
P-1265 System	is a part of	Boiler unit	

The resulting collection of expressions shall be verified on its semantic consistency, for example by importing it in software that is able to execute a semantic verification.

Step 1.2 – Apply consistent naming conventions

Normally any individual thing is represented by a UID and may be denoted by several names, codes, abbreviations, translations, etc., each belonging to its own language community (as is discussed in par. Four: Bron van verwijzing niet gevonden). These names, codes, etc. extend the vocabulary of the formal language.

To ensure that a Product Information Model becomes a consistent whole it is important to ensure that coded names (tag names) and names of individual things, such as facilities, sections, units, systems, equipment, instruments and components, are consistent with a defined naming convention. This means that a coding system and naming convention shall be adopted.

For existing facilities and products with a history of more or less consistent numbering and naming conventions, it is usually not justifiable to modify existing names, such as tag names and document titles. Nevertheless, the names in a Product Information Model database might be changed without changing the information in the documents themselves whereas synonyms might be added to increase the consistency of the names and titles while also retaining access via old names.

For example, actions to make names consistent may include:

- Add or remove dashes and spaces, names in proper case and classes in lower case, correct typing errors, and change abbreviations to full terms.
- Harmonize codes for equipment types.
For example:
 - choose FL or FIL in case of filters, but not both, or add synonyms if existing conventions should be retained.
 - choose SV or PSV in case of safety valves, but not both, or add synonyms if existing conventions should be retained.
- Remove use of plural or add singles as synonyms and define collections.
- Allocate a range of UUIDs for things and for relations (see par. 3.3).

If old names are and will remain in use, then the old names need to be retained in a collection of synonym relations that form the mapping between old and new names. The following expression is an example of a mapping relation:

102 SV123 is a synonym of 102 PSV-123

Note that the UUID of the named thing at the left hand is the same as on the right hand side, because the two names denote the same thing.

Such synonym relations enable that things can be named and searched by the old as well as the new names. In addition to that the formal language enables to express in which language community the names are the preferred names, as will be discussed later.

Step 1.3 - Classify products, equipment and components

Classification is the process of specifying classification relations between individual things and concepts (kinds) that are defined in definition models and that may appear in knowledge models or requirements models. Classification relations specify the existence of the individual things and make them known things in the formal language.

Classification relations are important and required, because kinds of things (concepts) in the dictionary are the bearers of definitions, knowledge and requirements, as definitions, knowledge and requirements are expressed by means of relations between kinds of things and relations between kinds of things and documents that contain knowledge about those kinds of things.

Thus, the classification relations are the prime means to search and find definitions, knowledge and requirements about objects and they determine which knowledge and which requirements are applicable for the individual things.

For a similar reason it is important to classify individual occurrences, fluids, and aspects by concepts in the formal dictionary or in a dictionary extension.

The most important reason to classify things that way is that by such a classification all parties that are involved in the total life cycle of a product can be aware of the unambiguous definitions, requirements, etc. so that information exchange between the parties and interpretation of requirements becomes less ambiguous.

In order to be able to classify individual things it is obvious that the concepts that will be used for classification should have been defined in the first place. This gives evidence to the need to develop and maintain a common formal dictionary and to relate organization specific terminology as synonym terms for those concepts.

For example, a Motorway Information Model for motorway M1 that is expressed in a formal language would mean that a part of the model for example of a particular viaduct could contain the following facts:

V5	is classified as a	viaduct type 3
S-5	is classified as a	road surface layer
S-5	is a part of	V5

When the concepts ‘viaduct type 3’ and ‘road surface layer’ exist and are defined for example in a domain dictionary this means that the knowledge about those concepts and about the concepts that are related to them, such as their parts, are directly accessible.

This example illustrates how the objects V5 and S-5 are classified by concepts from a domain dictionary by means of a classification relation. In addition to that, the third line illustrates a formal language expression of a fact in an information model about motorway M1.

Thus, the various products and all their components at various aggregation levels, including systems, units and equipment need to be classified by kinds that are selected from the dictionary. Also all aspects, activities, functions (roles), etc. shall be classified by the concepts (kinds of things) that are defined in the formal dictionary or in a dictionary extension. A classification relation implies that all definitions, knowledge and requirements that are

specified for the kind and all its supertypes are applicable for the classified individual thing!

Each item shall be given a language wide unique identifier (UID) to represent its identity in the language in the range that is allocated to the project (see paragraph 3.3).

Each classification relation shall have a status of validity of the fact (not shown in the table below), which initially may be ‘proposed’ and finally shall be ‘accepted’ for valid statements. If an item need to be removed from a product composition hierarchy, then the status of the classification relation shall be qualified as ‘deleted’ or ‘replaced’, which indicates that the item does not exist anymore, and all relations in which the item occurs shall be treated accordingly.

For the classification relation the formal relation type ‘is classified as a’ shall be used (selected from the formal dictionary).

This shall result in a number of expressions. For example three classification relations as presented in the core of the following Expression Table:

2	101	1	60	3	15	201
UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
108,000,000	LNG site	208,000,000	1,225	is classified as a	700,000	site
108,000,062	Air Compressor System	208,000,080	1,225	is classified as a	130,073	compressor system
108,000,001	LNG plant	208,000,001	1,225	is classified as a	160,081	process plant

In the above table the UIDs of the project are in the range 108,000,000 and the UIDs of the facts are in the range 208,000,000 whereas the UIDs of the right hand side of the expressions are taken from the dictionary.

The definitions of the kinds of relations include the specification of the kinds of things that are allowed to be role players in the relations. This implies that the kinds of things should be consistent with the use of the individual things in other relations. For example, because the objects in the above classification relations are classified, it is implied that they are individual things. This means that they may not be used in other relations that are relations between kinds of things. The semantic consistency of the expressions can thus be verified by software. For example the Gellish Search Engine is suitable to execute such a semantic verification when the information is expressed in a Gellish Expression Format [Ref. 4].

A semantic verification might report missing data, duplicate lines and inconsistent data (which shall be fed back to the project team and shall be corrected in the database).

5.3 Step 2, Create document models

In information management we make a distinction between two different kinds of objects:

- physical information carriers;
- documents, being the content of such carriers.

Multiple information carriers can contain the same information. This is the reason why we distinguish between documents and information carriers and therefore a document (a ‘common content’ or qualitative information) is a separate object that can have a relation with multiple physical files, as is described in paragraph 4.7. For example, the same (qualitative) information can be present on a physical file in MS-Word format and on a file in pdf format and on a paper copy. All those three physical information carriers present the same document.

A document is thus defined as information (which may consist of a collection of pieces of information) that forms the common content of one or more physical information carriers. An information carrier is a physical object on which the information is presented. Examples of information carriers are: a sheet of paper, a bundle of sheets of paper, an electronic file, such as a doc file or a pdf file, or it can be a collection of electronic files (possibly collected in a directory).

So a document is defined here as either:

- A drawing in a particular version. It may be a part of a number of sheets. The identifier of a drawing thus may include a sheet number and typically includes its revision id.
For example: Drawing ‘T-123 sheet 1 of 4 - rev. A’.
- A data set in a particular version.
For example: Spreadsheet ‘U-1200 Equipment list – rev. B’
- A multi-page document in a particular version.
For example: Report ‘R-123 version 2.1’.
- An information content of any other medium in a particular version.
- A collection of documents.
For example: the collection of drawings on different sheets that make

up T-123 or the content of binder ‘Volume 2 Gas turbine Equipment Operation’.

When we use the term ‘document’, then we mean any of such common content and not its physical manifestation. A document is typically materialized in textual form in natural language. However, content can also be a drawing, images or sound, a number of expressions in a formal language or information in another form or a combination of forms. It can even be a computer program (software). However, for readability reasons we keep using the term document, although the term can denote information content in any form on any medium.

When an information carrier contains content in tabular form, whereas the table columns and rows have defined meanings, then we usually call such a document a ‘data set’. A data set can be the content of an electronic data file or a database or collection of files. Examples of data sets are: the content of a spreadsheet or the content of an MS-Access or Oracle database. The physical information carriers are then the xls files, mdb files, etc.

This paragraph describes how to create formal language expressions with data about a physical file and data about the document(s) that is (are) contained in those file(s). Such data about documents and files is also called meta-data about the documents, which thus also includes information about drawings, data sets, etc. and the files that contain them.

The chapter consists of three parts:

1. Meta-data about the documents (the content of the files).
2. Meta-data that relate the documents to the product and its components, area’s and processes.
3. Meta-data about the physical files and their relations to documents.

5.3.1 Step 2.1 - Specify meta-data about the information

Make document titles consistent

Document titles and data set titles are recommended to be made compliant with a standard, such as ‘Requirements for Information about Documents’ [ref. 2]. If existing documents or data sets need to be included in an information model that does not comply with those requirements, then they are recommended to be improved according the guidelines in that document. Even then, it may be decided to improve only the titles in the database and not to change the titles on the physical documents (files). The mapping to the

old titles shall then be retained in a separate mapping table in preparation of automated conversion in case information is delivered using the old document titles again.

Qualify information

Each piece of information or document (thus including also data sets) may have a name and shall be defined (according to its content) being of a kind of information. Typically that kind is a standard document type or type of data set that is selected from the formal dictionary. Such a definition relation (1726) specifies the existence of the qualitative information. This enables including the full text as description of the qualitative information. For example a piece of information that is a requirement for bridges can be called requirement 35, and such a qualitative requirement may be specified in full text, whereas the information on a drawing is only denoted by the drawing number and possibly a title. These two examples result in the following formal language expressions:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	Description	Status
101	requirement 35	201	1,726	is defined as a	970,007	requirement	A bridge shall be able to carry a load that is ...	accepted
102	T-123 sheet 1 of 4 - rev. A	202	1,726	is defined as a	490,183	P&ID	P&ID of Unit 3100	accepted

Unique Identifiers (UIDs) shall be allocated to all (pieces of) information and to all pieces of aggregated information.

Each qualification relation shall have a status of validity of the fact. The status may be ‘proposed’, ‘accepted’, ‘deleted’ or ‘replaced’. If the status is ‘deleted’ or ‘replaced’, then the information is archived or it’s physical (master) copy(s) may be destroyed.

The relation between the information and the information carriers is discussed in step 2.3.

Aggregating or decomposing information

For some document types (such as binder contents or multi-page drawings) it may be valuable to define also one or more higher levels of aggregated information. For example, it may be decided to arrange information in sections or in packages (also physically, such as in binders), whereas the sections may be aggregated in volumes. For example, it is recommended to

include the pages of multi-page documents in an aggregated document, especially because some information will be valid for the whole content and references may be made to the whole, instead of to the separate sheets or versions. Typically the structure of aggregated information will reflect a table of content, as it will describe which pieces of information are included in which section or whole of the aggregated information, such as a package.

This should be done as follows:

1. Define the aggregated information and define it. For example as a package.
2. Define pieces (as required), being parts of the information that are included in the larger piece of information and qualify those pieces. For example as section, or as subsection or package.
3. Define for each document (or sheet) that it is part of a particular subset collection.
4. Define for each piece that it is included in the larger aggregated information.
5. If an electronic file is created for aggregated information, then it shall be specified that the information <is presented on> a particular electronic file, in the same way as is done for documents. This is further described (and illustrated) in step 2.3.

This will result in the following expressions:

UID of fact	UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
211	111	LNG Plant Documentation	1,726	is defined as a	970,492	aggregated information
212	112	Section 1, Heat exchangers	1,726	is defined as a	970,493	information in section
213	112	Section 1, Heat exchangers	5630	is information that is included in	111	LNG Plant Documentation
214	113	E-6004 construction drawing	5630	is information that is included in	112	Section 1, Heat exchangers

Note that the above table does not describe a physical drawing, neither on paper, nor as an electronic file, but only the (common) content of such paper copies and files.

Specify version relations between documents

Documents that are succeeded by new versions shall be related to their successor. This is done by specifying a version succession relation between the succeeded document and the successor document.

For example:

UID of fact	UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
221	122	LD1-Rev. A	1,393	is a next version of	120	LD1-Rev. O
222	123	PP11-1 sheet 1 of 2 Rev. B	1,393	is a next version of	121	PP11-1 sheet 1 of 2 Rev. A

If documents are withdrawn, then the status of the relation by which it is defined (1726) shall be modified into ‘deleted’ or ‘replaced’. When a document is replaced by another document, then an auxiliary relation shall be created between the replaced and the replacing document that expresses that the replaced document is replaced by the replacing document.

This is similar to expressions that state that a fact is replaced by another fact. The existence of anything in the formal language is determined by the presence of a defining relation, being a classification relation, a specialization relation or one of their subtypes (such as 1726). Therefore, a generic way to express replacement relations is by relating a replaced fact to a replacing fact. The Gellish Expression Format implemented this in a generic way, by including an extra table column, called ‘UID of successor of fact’, which provides a field for a UID of a replacing fact. Note that an indicated replacement is only valid if the status of the replaced fact is ‘replaced’. Thus the format enables to express such a replacement on the same line as where the replaced document is defined. Thus, the UID of the fact that contains a defining relation for the new document shall be recorded as a ‘UID of successor of fact’ on the line where the replaced document is defined by a defining relation. The status of the fact about the defining relation for the replaced document shall become ‘replaced’.

This is illustrated in the following example:

UID of fact	UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	Status	UID of success or of fact
223	120	LD1-Rev. O	1,726	is defined as a	490,183	P&ID	replaced	224
224	122	LD1-Rev. A	1,726	is defined as a	490,183	P&ID	accepted	

Arrange documents in a table of content

A large number of physical documents of the archive may be filed in binders. There is usually an overall table of content of the binders and per binder there may then be a detailed table of content (also called a document index). A table of content arranges documents, usually in a logical order. Therefore, that order is reflected in the database, to facilitate searching of documents.

For example, the formal language representation of a table of content of a 'Mechanical Catalogue' part of an archive may be built-up as follows:

- The documents are collected in binders (volumes).
- A total archive is composed of a collection of sections and subsections, which are a logical grouping of documents of similar nature. For example, all documents about heat exchangers may be collected in Section 8 Heat Exchangers and Subsection 15 G contains documents about Salt Water Screens.
- Each Section or Subsection may cover either a part of a binder, a full binder, or it covers several binders. This means that Sections or Subsections may be segmented over different binders.
- Individual documents are also collected in packages. Each package is collected in a Section or a Subsection. For example, purchase orders are collected in P.O. packages, whereas each package is denoted by a package-number (e.g. Pack-36721 Electric Motors).

These various groupings need to be represented in an information model for the archive. For example, Subsection 15 G is split over Volume 39 and 40 and all documents about package Pack-36501 Nitrogen Generation Units are collected in Section 16, which section consists of three subsections, which are distributed over Volume 41, 42 and 43.

- For relevant individual documents it may be indicated about which piece(s) of equipment (tag nr) the document contains information, resulting in relations between documents and tagged items.

The distribution of documents over Volumes (binders) is not relevant for an electronic archive. This implies that the Volumes might be eliminated from the formal language representation of the table of content. However, it might be relevant to maintain the relation with an existing paper based situation. Therefore, the distribution of documents over the Volumes can be reflected also in the database.

As a result, the above information can be recorded in formal language expressions as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
656	365-PP12-1-V28-59-1	879	1,726	is defined as a	490196	drawing
704	NG-601	980	5,046	is described in	656	365-PP12-1-V28-59-1
656	365-PP12-1-V28-59-1	946	5630	is information that is included in	955	Volume 42 Section 16 Part of Nitrogen generation units
955	Volume 42 Section 16 Part of Nitrogen generation units	947	1,726	is defined as a	970,495	archive subsection
955	Volume 42 Section 16 Part of Nitrogen generation units	948	5630	is information that is included in	337	Section 16 Nitrogen Generation Units
337	Section 16 Nitrogen Generation Units	456	1,726	is defined as a	970,493	archive section
337	Section 16 Nitrogen Generation Units	457	5630	is information that is included in	357	Mechanical Catalogue of Liquefaction Plant
357	Mechanical Catalogue of Liquefaction Plant	458	1,726	is defined as a	970,002	information

Note that relation types that are used to describe collections differ from relation types that are used to describe assemblies, because assemblies are about single objects, whereas collections are about plural objects. For example, to indicate that one collection is a subset of another collection we use the ‘is a subset of’ relation whereas for an assembly we use the ‘is a part of’ relation.

5.3.2 Step 2.2 - Relate documents to product components

Specify for each component of a product on which drawing it appears or which document contains information about it. An alternative specification of the same information is to specify the inverse relations that describe for each document about which component or about which process or activity it contains information.

For example:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
100	D-326	892	5,046	is described in	146	LD1-4-C1-0100 Rev. 0

When a document is succeeded by a next version (which strictly speaking is another document), then it is probable that an object that is described on the first document is also described on next versions of the document. However this is not a logical consequence, because there is no constraint on the removal of information about objects from a next version. This means that application software might support searching by implementing this “fact propagation” rule, although it is recommended to indicate that it is a probability by deduction.

5.3.3 Step 2.3 - Relate documents to files at addresses

For each document it shall be specified on which a physical medium it is presented, either as an electronic data file or on paper, microfilm or any other medium. A file name shall include a file extension.

In addition to that each physical file shall be classified as an electronic data file and each physical file shall have its own UID allocated. Finally a directory shall be defined. The directory shall be classified and for each file it shall be specified of which directory it is an element.

This will result for example in the following expressions:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
146	LD1-4-C1-0100 Rev. 0	223	4,996	is presented on	108,300,308	BRPFD09.dwg
308	BRPFD09.dwg	224	1,225	is classified as a	490,533	electronic data file
308	BRPFD09.dwg	223	1,227	is an element of	108,300,567	C:\LNG\Files
567	C:\LNG\Files	225	1,225	is classified as a	492,017	directory

Multi-file documents

Sometimes a document is physically presented on multiple files, which may or may not be collected in a zip file. Typically a document and its appendices are often presented in different files, whereas the file types of those appendices may also differ from the document and from each other. For example, a main document may be stored as a doc file, whereas one of the appendices may be stored as an xls file.

In such multi-file cases the collection of files is a meaningful object. Therefore, an object shall be created that represents the collection of files. For each physical file that forms part of the multi-file document it shall be specified that it is an element of that collection. If the files are physically collected, e.g. condensed in a zip file, then that collection is real and will have zip as its file type. If the files are not physically collected in a zip file, then the collection is not represented by a separate file, but it is a logical collection that should be created as a recognized imaginary object.

Either the directory of the separate files or the directory of the zip file shall be specified by a relation between the file(s) and a directory (which is also a collection of files).

For example, a collection of expressions that is not arranged in a zip file, the table will become as follows may be specified in the following expressions:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
900	MCC_C single line diagram	500	4,996	is presented on	956	MCC_C collection of files
956	MCC_C collection of files	501	1,225	is classified as a	492,018	collection of electronic data files
957	MCC_C1.dwg	502	1,225	is classified as a	490,533	electronic data file
957	MCC_C1.dwg	503	1,227	is an element of	956	MCC_C collection of files
957	MCC_C1.dwg	504	1,227	is an element of	567	C:\LNG\Files
958	MCC_C2.dwg	505	1,225	is classified as a	490,533	electronic data file
958	MCC_C2.dwg	506	1,227	is an element of	956	MCC_C collection of files
958	MCC_C2.dwg	507	1,227	is an element of	567	C:\LNG\Files

5.3.4 Step 2.4 - Compose documents

Documents usually consist of monolithic pieces of text. This implies that information about such documents (meta-data) can only be related to those documents as a whole. Such documents may contain so called hyperlinks that consist of pointers that are included in the document and that refer to other documents or parts of the same document. Conventionally pointers operate only in one direction and their operation is dependent on the location and name of the referred documents. This is a draw-back that often leads to broken links. Such hyperlinks are internal document content and therefore they are not managed explicitly in a Product Information Model.

However, another possibility is that documents are decomposed in separate pieces of text, on in a level of granularity that may vary. For example, a document may be decomposed e.g. in sentences or paragraphs and chapters, whereas the sequence in which the pieces of text in the documents appear is recorded in a document model. This enables that the separate pieces of text can be related individually to elements of the Product Information Model. For example, a paragraph may express a requirement for a particular component or for a kind of component. An explicit relation between that component and that requirement is a very powerful means to manage requirements and to verify whether they are satisfied. The specification of such a decomposed document in a computer interpretable form is described below.

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	Full description
102	paragraph-2	201	1,726	is defined as a	970,007	requirement	The lubrication system shall have a capacity that ...
102	paragraph-2	202	5,398	is a specification for a	130,166	lubrication system for a compressor	
102	paragraph-2	203	5,519	is by definition a part of	103	API 617	
102	paragraph-2	204	5,332	is the next element after	101	paragraph-1	

5.4 Step 3, Specify activities and processes (functions)

Processes are the occurrences that take place in or with the facilities and its components, such as the burning of fuel in a diesel engine, the production of clean water in a water purification system, or the transport of people and goods via a railway, but also the control of those processes, the measurement of the properties and the performance of the facilities, etc. These processes are often called the functions of the facilities that perform the processes.

Activities are occurrences that are performed by people, such as fabrication, inspection, operation and maintenance of the facilities and their components.

Both kind of occurrences, processes and activities, can be decomposed in smaller occurrences, such as tasks, events, and the like. Their decomposition is described in the same way as the decomposition of a product in components.

For example, the decomposition of a liquefaction process in a compression and a cooling stage are described as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
141	Compression in K-1301	240	1,190	is a part of	140	Liquefaction of S1
142	Cooling in E-1304	241	1,190	is a part of	140	Liquefaction of S1

Similarly the decomposition of a commissioning activity in sub-activities can be expressed as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
151	Cleaning of C-1201 System	251	1,190	is a part of	150	Commissioning of C-1201 System
152	Inspection of C-1201 System	252	1,190	is a part of	150	Commissioning of C-1201 System

The further decomposition into tasks is done in the same way.

Any occurrence can be described by its relations with the physical objects and/or people that are involved in their particular role in the occurrence.

For the description of physical and chemical processes there are additional objects involved, being the inputs and outputs of process steps (unit operations) such as streams, raw materials, intermediate materials or finished products. Furthermore, the same thing or stream that has a role as output of one process step has a role as input in the next process step. This is described by the input and output roles that the things or streams play in relations with the process steps. For example, the liquefaction process that was decomposed above into a compression and a cooling process step has the following flow diagram:

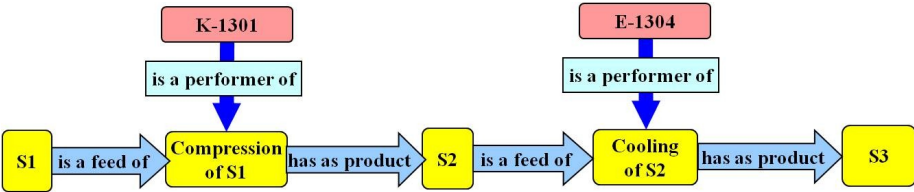


Figure 12, Process Flow Schema of gas liquefaction

The compressor K-1301 is performer of the Compression of S1 process step, which has S1 as input and S2 as output stream. That same S2 is input in the process step ‘Cooling of S2’, which has E-1304 as performer and has S3 as its output stream. In other words: Compression of S1 is a function that is performed by compressor K-1301.

These process steps, streams and performers can be added to a Product Information Model for the liquefaction facility as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
161	K-1301	261	4,761	is performer of	162	Compression of S1
162	Compression of S1	262	1,225	is classified as a	191,936	compression
163	S1	263	4,785	is an input in	162	Compression of S1
164	S2	264	4,786	is an output of	162	Compression of S1
163	S1	265	1,225	is classified as a	432,099	gas stream
164	S2	266	1,225	is classified as a	432,099	gas stream
165	E-1304	261	4,761	is performer of	166	Cooling of S2
166	Cooling of S2	262	1,225	is classified as a	191,804	cooling
164	S2	263	4,785	is an input in	166	Cooling of S2
167	S3	264	4,786	is an output of	166	Cooling of S2
167	S3	265	1,225	is classified as a	432,099	gas stream

An occurrence can be specified also by aspects of the occurrence, such as the fact that it begins at a particular moment in time at a particular location and that it has a certain duration (which may be negligibly short). The addition of such aspects is described in the next paragraph.

When activities are defined it might also be needed to specify which objects are subjected to the activities and in which sequence the activities need to be or are performed. This can be done by specifying relations an object and an activity and relations between the activities. For example, a decomposition of activities in tasks, or a sequence of activities, can be described by relations that specify that an activity is a predecessor of another activity, such that an activity can or may only start when the previous activity is completed. Such relations between activities and involved objects are specified as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
150	Commissioning of C-1201 System	251	4,760	has as subject	140	C-1201 System
151	Cleaning of C-1201 System	251	1,388	is a predecessor in time of	152	Inspection of C-1201 System

Note that the first line describes another kind of involvement in an activity than the input and output or performer kind of involvements. It is the inverse of the relation type <is a subject in>, so that the inverse expression would be:

C-1201 System is a subject in Commissioning of C-1201 System

Documents about the way in which this kind of activities should be executed as well as documents that describe the results of a specific activity can be added in the same way as described above for the relation between physical objects and documents.

The next chapter elaborates further on the modeling of activities, processes and events.

5.5 Step 4, Specify aspects of products and components

5.5.1 Properties and qualities

Aspects include characteristics, such as properties and qualities of assemblies and components or streams, activities and processes. The term ‘property’ can cause confusion, because it is used by different people to denote different concepts (e.g. in RDF the term property is used to denote a concept that is here denoting a binary relation). Here the concept ‘property’ is defined as a subtype of aspect that has an extent or intensity that can be quantified by one or more relations to numbers or ranges, typically on a scale. Qualities are aspects which quality cannot be quantified by a number, but they can be qualified by relating them to defined qualitative aspects (textual values).

Any individual physical object in a product may have various aspects of different kinds. Aspects can vary from single value aspects to complicated aspects such as aspects that describe behaviors or shapes and properties that have values that vary over time and sometimes have values that vary by measured location in space. Because of that we should make a distinction between individual aspects and their value(s) and therefore individual aspects need their own individual identity (UID).

Below we discuss the modeling of allocation of single value aspects. The modeling of behavior (activities and processes) is described in chapter 7; the modeling of time and shapes is described in chapters 8 and 9.

Aspects can have qualitative values and often also quantitative values.

Qualitative aspects are aspects that have values that are typically denoted by texts or by symbols. They are usually selected from a list of allowed values, such as provided by a pick-list in a system. For example, a color is an aspect which values may be selected from a list of allowed values, such as red, green and blue. Such (allowed) qualitative aspects are defined concepts and

therefore they should be included in and selected from the formal dictionary or should be added to it if they do not exist yet.

Quantitative aspects (properties) are usually quantified by a number on a scale. For example, an operating temperature is an aspect which values can be quantified on a Celsius scale or on a Fahrenheit scale. The numbers and the units of measure should also be selected from the dictionary (and maybe they need to be added to it). Note that quantification of an aspect by a number is a special case of its qualification by a value in textual form. Therefore, quantification is a subtype of qualification.

The specification of aspects requires a pattern of at least three main expressions: the specification that an individual object has (possesses) an individual aspect the classification of the aspect and the qualification or quantification of the aspect. The aspect pattern requires the following kinds of relations:

has as aspect
is classified as a
is qualified as

Whereas the latter can be replaced by one of several quantification relations, such as:

has on scale a value greater than

For example, a vessel, called V-6060, has an inside diameter of 2500 mm and is painted in a green color. This is specified as follows:

UID of left hand object	Name of left hand object	UID of Fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	UID of UoM	Name of UoM
108,400,601	V-6060	601	1,313	has as aspect	108,400,701	diameter of V-6060		
108,400,701	diameter of V-6060	602	1,225	is classified as a	550,134	inside diameter		
108,400,701	diameter of V-6060	603	5,025	has on scale a value equal to	922,305	2500	570,423	mm
108,400,601	V-6060	604	1,313	has as aspect	108,400,702	color of V-6060		
108,400,702	color of V-6060	605	5,020	is qualified as	551,666	green		

For the specification of a material of construction there is a special relation type that enables to specify the material on one line, instead of two as follows:

UID of left hand object	Name of left hand object	UID of Fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	UID of UoM	Name of UoM
108,400,601	V-6060	606	5,423	is made of	280,192	stainless steel		

5.5.2 States and phases

States and phases are described in a similar way as qualities. There is a concept of state or phase and there is a qualification of those concepts by qualitative values or by quantitative values (also called qualitative aspects and numbers on scales).

Material can be in various states, which states may be qualified for example as vapor, liquid or solid, but they can also be in other states, such as being in a damaged state, or rusted, painted, etc.. The first kinds of states are also called phases. Such states (phases) are dependent on the temperature and pressure of the material. A batch of fluid can even be in a mixed phase, for example it can be in vapor/liquid state, which means that it is partly liquid and partly vapor. Some states have a special name, such as ‘steam’, which is a type of water, which by definition is in a vapor phase. Thus, a phase is a specialization of a state, whereas liquid, vapor, etc. are qualitative subtypes of the concept ‘phase’.

Note that the term ‘gas’ is used in two ways: one as a kind of state (phase) as synonym of ‘vapor’ (especially in American English), the other one as a kind of atomic structure, which indicates that a batch of material will be in the vapor phase when being at normal atmospheric conditions.

Solids can be in various crystalline forms. Well known examples are: snow and ice, but also ferrite and carbonite are states of solid material (of steel).

The state (or phase) of physical objects can be modeled as follows:

fluid-1	is in (has as aspect)	state-1
state-1	is qualified as	vapor
fluid-2	is in	state-2
state-2	is qualified as	vapor/liquid
batch-1	is in	state-3
state-3	is qualified as	granular

Other subtypes of state (qualitative states) describe the mode or situation in which a physical object is. For example:

valve-1	is in	state-1
state-1	is classified as	open

item-1	is in	state-2
state-2	is classified as	corroded

Note, that all these states can be classified according to their concept as being (a subtype of) either a phase or a state.

The fact that states are made explicit and have their own individual identity (UID) enables that each state can have various values over time. Thus state changes do not imply multiple states, but they imply multiple values that should be related to the same individual state, each with its own time stamp (modeling of time will be discussed later).

A direct classification of a physical object by its state is a short-cut relation that does not recognize the state as an individual aspect that can vary over time. This would result in:

fluid-1	is classified as a	vapor

Strictly speaking this classification of fluid-1 is not really a classification of its nature, because it is only incidentally true, as long as fluid-1 is in the vapor phase.

A formal language taxonomic dictionary should contain standard states, also called qualitative states, as the Formal English dictionary does.

5.5.3 Structure - Process material & radiation

Physical objects usually have as one of their characterizing aspects that they are composed of some kinds of components and by the number of components and the way in which they are arranged. When this is applied to the atomic and sub-atomic level, then we call that aspect their substance or stuff. In other words, physical objects have an aspect that is called its substance, which is a kind of arrangement or structure and bonding of elementary particles, which usually implies also kinds of arrangements of types of atoms and molecules.

For fluids and other materials which are subject to a process the kind of stuff or substance is usually denoted by their role as kind of *process material*. For solid items the stuff is usually referred to as their *material of construction* (the latter is further discussed in the next paragraph).

Examples of kinds of substances that can be used as process material are:

- water, oil, polymer, gas, air, acid, rock, steel, etc.

Individual physical objects thus have individual substances that are classified by kinds of substances. However, often a short-cut relation is used to directly

relate the individual physical object and the kind of substance, without making the individual substance explicit. This short-cut kind of relation is therefore called a classification of a physical object by a kind of substance. For example, if a batch of fluid has an aspect that is classified as water, then it is possible to classify the batch directly as being water. Thus the expressions:

F1 has as aspect S1
S1 is qualified as water

Can be replaced by an expression that uses a short-cut relation as follows:

F1 is classified by substance as water

The basic concepts for such classifications are kinds of structures of elementary particles that are called kinds of chemical *elements*, according to the periodic system of elements, together with isotopes, molecules and ions, etc. In industrial practice all physical objects are composed of mixtures of various types of elements, molecules and particles, the concepts of pure components are used primarily in theoretical models and simulations. For that reason it makes sense for the dictionary to distinguish between chemically pure components and industrial substances that are typically mixtures.

For example, the term ‘hydrogen’ can be used to denote different concepts, such as a kind of element (H), a kind of molecule (H₂), or a kind of gas (a collection of H₂ molecules, typically being in a gas phase). In practice, hydrogen gas is never 100% pure hydrogen. Thus hydrogen (gas) is defined as a gas that mainly consists of hydrogen atoms, typically bond in molecules. Thus when a material is classified by a kind of substance, it means that the classified material is normally a mixture that mainly consists of a specific kind of components in some arrangement. This still allows for substances that are assumed to be pure. Thus, if we classify a physical object (by substance) as being hydrogen, then it means that it mainly consists of hydrogen atoms that are arranged in any form. Purities and concentrations of materials can be expressed via specifications of composition relations (for the modeling of concentrations of components in mixtures see chapter 5.5.6).

The terminology that is usually used for specifying a kind of substance for solid items is different from specifying a kind of substance for fluids.

For example, for solid items we say: ‘item-1 is made of steel’, whereas for fluids we say: ‘batch-1 consists of water’ (or shorter ‘batch-1 is water’). For example, the different phrases in the following four expressions nevertheless denote the same kind of relation.

fluid-1	consists of	water
fluid-2	consists of	oil
batch-1	consists of	PVC
item-1	is made of	PVC

Therefore, these two phrases are synonymous.

Waves and radiation

Waves and radiation are physical objects (phenomena), although they have no mass. Waves have structural aspects, such as frequency and amplitude. Waves are usually classified directly by kinds of wave that imply their structure to some extent.

Examples of kinds of waves are:

- x-ray, radio wave, electricity, heat radiation, light, noise, etc.

The following expression classifies a wave:

signal-1 is classified as a radio wave

5.5.4 Material of construction

A specification of a ‘material of construction’ specifies the kind of substance aspect of a material that is used for constructing a solid item. It implies that the stuff is in solid state at atmospheric conditions and at operating temperature.

Thus, classification of an item according to its material of construction is a short-cut relation for expressing that the item has a substance aspect and classifying that aspect by a kind of (solid) substance.

For example, ‘steel’ is a kind of substance, which consists mainly of iron and carbon atoms in a specific structure’. It can be further subtyped as ‘ferritic steel’, ‘austenitic steel’, etc.

That kind of sort-cut relation is thus not a pure classification relation and is denoted by the phrase <is made of>.

For example a stainless steel bearing B1 is specified as follows:

B1	is classified as a	bearing
B1	is made of	stainless steel

The standard kinds of physical objects in formal dictionaries usually do not contain subtypes according to material of construction, because this would in principle create a combinatorial explosion of the number of concepts. Therefore standardized concepts such as ‘stainless steel bearing’ or ‘PVC pipe’, etc., will be added only in exceptional cases, especially when the

number of recognized materials of construction for a type of object is very limited.

5.5.5 Batches & streams and their states and properties

A quantity of material that does not have a particular shape is a physical object that is typically not classified by a kind of physical object, but by its kind of substance. Such quantities are typically batches, streams or heaps. This holds for fluids, such as liquids and vapors as well as for granular solids. For example, when we say: “this is water” or “this is (liquid) steel”, then we classify the fluid not by its form or function, but by its substance. This differs from the classification of solid items, where we usually don’t say: “this is steel”, but we say: “this is a pipe, that is made of steel”. A consistent method for classifying physical objects therefore is: classifying any individual physical object primarily according to a kind of physical object. Then, if required, state that the physical object possesses a substance (an aspect) and classify that aspect by a kind of substance. For example as follows:

S1	is classified as a	batch of liquid
S1	has aspect	Substance of S1
Substance of S1	is qualified as	water

It is also possible to classify a batch by its substance, by using a short cut relation type instead of the last two lines in the above detailed model. This is done as follows:

S1	is classified by substance as	water

The use of such a short cut relation type requires that consistency is managed, because other people may apply the extensive way of modeling. That consistency can be verified by software, because the definition of the short-cut kind of relations includes references to the two kinds of relations in the extensive model pattern.

A distinction between batches or streams and their substance enables decomposing batches and streams into components. For example a two phase flow of a mixture consists of two partial streams, e.g. a vapor flow and a liquid flow, each with its own substance. Such a mixture may have as aspect a vapor/liquid ratio. That ratio may have a single value or its value may change over time, whereas also the quantities (concentrations) of the components in the vapor phase and the liquid phase may change over time. Then each component can be classified separately and each component can have its own substance that is qualified differently.

Batches can be static, such as crude oil in an oil field or water stored in a tank. Batches can also be in motion, such as a liquid which is stirred in a mixer. When a batch is moving from one place to another through a channel or pipeline, then it is called (classified) as *stream*. For example, a liquid or gas flowing through a pipeline, or water in a river is a stream.

Note: the term ‘stream’ can in principle be used for any queue of physical objects flowing in a row, such as a stream of cars. In particular contexts, such as in the context of process plants, it only refers to moving fluids, typically through a pipeline. A stream of fluid material is similar to an ‘electric current’ which is a moving electromagnetic wave (moving electrons), typically through a conductor.

Fluids can possess properties such as pressure, temperature, density and viscosity. At a particular location and moment in time a fluid is in a specific state or condition. Such a state means that all its properties at that location and at that moment in time have values that are valid together. This is sometimes called a ‘property point’. For example, a fluid can be at its critical point (state), which means that particular ‘critical properties’ (values) are valid at that condition.

A state may change, which means that the properties are variable. Therefore, the state properties are also called state variables.

For example, the properties of a fluid at the suction nozzle of a pump can be described as follows (note that suction nozzle-1 indicates a physical object which is part of the pump):

Fluid-1	has aspect	State-1
State-1	is classified as a	state
State-1	is valid at location	Suction nozzle-1
Fluid-1	has aspect	T-1
T-1	is classified as a	temperature
T-1	has on scale a value equal to	30 C
Fluid-1	has aspect	P-1
P-1	is classified as a	pressure
P-1	has on scale a value equal to	50 bar
Fluid-1	has a property of	Dens-1
Dens-1	is classified as a	kinematic viscosity
Dens-1	has on scale a value equal to	30 kg/m ³
T-1	is a state variable of	State -1
P-1	is a state variable of	State -1
Dens-1	is a state variable of	State -1
etc.		

Typically there is a correlation between the state properties, so that they are mutually dependent. An example of such a correlation is an equation of state, which is a law of physics that describes how the pressure, temperature, density and compressibility of an isolated fluid in equilibrium influence each other when they change over time. The properties are parameters in the correlation.

5.5.6 Concentration of elements in mixtures

The taxonomic dictionary includes the definitions of concepts that represent chemical elements from the periodic system of elements. These are considered to be ‘pure’ chemicals. For example, it includes H, which is the name of an element, also called ‘hydrogen’, which is a physical object of a particular sub-atomic structure.

In addition to that, the dictionary contains kinds of substances, which includes kinds of molecules and mixtures of various types of molecules. For example, H₂ is a concept that could be used to classify a batch of gas by its substance as being hydrogen or to classify a molecule as being of substance hydrogen.

In practice materials are never 100% pure, although in theory and in models and as an approximation they can be treated as if they are pure. So, in practice, H₂ can be used to classify a gas which consists *mainly* of H₂ molecules, and partly of other kinds of molecules. The purity, and its opposite impurity, can be defined as a property that describes the extent in which the classification by substance holds. For example:

Fluid-1	is classified by substance as	hydrogen	
Fluid-1	has aspect	p1	
p1	is classified as a	purity	
p1	has on scale a value equal to	98	%

Note that in this way of modeling there is no direct relation between the purity p1 and the classification relation. Therefore it is preferred to make explicit that the purity p1 is in fact an aspect of the classification relation. This is the reason why in a Gellish Expression Format that relation about a relation can be expressed in two additional columns in the table, one column for a UID and the other for a name, to express the purity as an extent in which is classification is the case. This is done as follows:

Name of left hand object	Name of relation type	Name of right hand object	Extent	UoM
Fluid-1	is classified by substance as	hydrogen	98	%

The detailed composition of the material can further be defined by specifying the concentration of other kinds of molecules in the material.

In practice, a *concentration* is often used as if it was a property of a mixture, for example: a dictionary might include the kind of property ‘sulfur content’ or ‘sulfur concentration’. If we would define concentrations as properties of mixtures then the dictionary should be extended with a separate concentration property for each kind of possible component. That would lead to an enormous number of kinds of properties. However, a concentration is not a property of a mixture or a property of a component, but it is by definition a ratio between two quantities (properties) of different substances (the component and the mixture). Furthermore, the above way of modeling purity and concentrations have as disadvantage that the individual components have no explicit identity (UID) and thus their quantity and properties cannot be specified. Therefore, it is better to describe concentrations in a formal language as the extent (fraction or percentage) in which a component is a part of a whole. In other words the concentration is an aspect (an extent) of a part-whole relation. For example, an H2S concentration in a fuel of 5 % can be modeled as follows:

Name of left hand object	Name of relation type	Name of right hand object	Extent	UoM
F1	is classified by substance as	fuel		
C1	is classified by substance as	H2S		
C1	is a part of	F1	5	wt%

Note that in both models F1 denotes a mixture, whereas in the latter model C1 represents the component. This enables that other aspects of C1 could be added to the second model. For example, the flowrate of C1 could be added.

The specification of concentrations in kinds of mixtures follows a similar pattern, but uses different kinds of relations. The following definition of water with 5% salt illustrates the application of a pattern that can be used to specify such a qualitative substance (being a substance with quantified concentrations).

mix-type-M	is a qualitative subtype of	salt water		
mix-type-M	has by definition a part a	comp type-S	5	wt%

5.6 Step 5, Relate fabricated items to designed items

Installed physical items are real fabricated items, each of which typically has a name that is a combination of its manufacturer and its serial number. Fabricated items are distinguished from the tagged items, which are their imaginary counterparts. A fabricated item that is installed at a particular place may be replaced by another fabricated item. Thus, in the course of time, there may be more than one item installed at the place of a tagged item.

Each fabricated item shall be defined by classifying it by the ‘manufacturer’s model’ (or ‘model and size’) of which it is an exemplar. The manufacturer’s model itself is a type of thing that shall be added to the dictionary by specifying that it is a subtype of a more general equipment type, using the relation type <is a model of>. The manufacturer’s model shall have a name that is a combination of the manufacturer’s name and the model name and possibly its size. Fabricated items are added and integrated in the Product Information Model via a specification of a relation with the tagged item at which place the real item is installed.

For example, a particular compressor with serial number Elliott E8368 is installed at the place of tagged item C-6002 A. That C-6002 A is classified elsewhere as a centrifugal compressor. The Elliott machine is an exemplar of a type that is a model of a centrifugal compressor. The addition of this installed item to the model will result in a table with a number of lines that express the above kinds of facts as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
131	ELLIOTT E8368	230	1,313	is installed at the place of	133	C-6002 A
131	ELLIOTT E8368	231	1,225	is classified as a	132	ELLIOTT model 46MB-4
132	ELLIOTT model 46MB-4	231	5,396	is a model of	130,057	centrifugal compressor

Measured data will always relate to the fabricated items, because the designed counterparts (the tagged items) are imaginary things that cannot be measured.

5.7 Modeling of physical objects

Making high quality information models highly depends on the proper classification of individual things as well on the selection of the right kinds of relations between the constituting components of the model. Therefore it is important to have a clear understanding of the nature of various kinds of physical objects. Therefore, below we discuss the nature of various kinds of physical objects and various aspects of their modeling.

5.7.1 Facilities and systems and their roles

Facilities and systems are high level assemblies of various types of equipment, piping, electrical items, instrumentation and control items, etc. They are physical objects just as the lower level assemblies. However, it is a bit too strong to say that they *are* assemblies, because not all components need to be assembled together. But it is too weak to say that a facility or system is a collection of items. Therefore a facility or system is a whole (physical object) that consists of components (physical objects) that are connected or arranged in a specific way in relation to each other, and may be more or less loosely assembled. For artificial physical objects it holds that they usually have an intended role or function.

The criterion for selection of the items that are included in a model may just be on the basis of the fact that the components play a role in a specific process. For example, a cooling system may consist of a collection of more or less independent items, all of which however contribute to some cooling process.

This means that in first instance some physical object may be classified only according to the role that it is intended to play (in the formal dictionary the concept ‘facility’ is defined as an intended role of facilitating.) In a later stage, when the technical solution is defined, it may be classified in further detail and then it may be decomposed after which its components will be classified.

For example, initially:

System-1	is classified by role as a	cooling facility
System-1	has as aspect	C1
C1	is classified as a	capacity
C1	has on scale a value equal to	300 kW

Note: when a specific physical object *is classified by role during a design*, then that classification implies that the object has an *intended role of the specified kind*, because such a role is apparently the intention of the designer.

After fabrication and construction the real installed system still has an intended role, independent of the fact whether it actually fulfils such a role. Therefore, the three phrases are all synonyms. In other words the following three expressions are three ways of expressing the same fact.

System-1	has an intended role as a	cooling facility
System-1	has a role as a	cooling facility
System-1	is classified by role as a	cooling facility

This example does not say anything about the technical solution or the type of components of the facility. The only thing that is known up to that stage of the design is that is an (intended) physical object, because a ‘facility’ is by definition a role of a physical object. At a later stage of the design the components may be defined, for example by specifying an air cooler as follows:

E-101	is classified as a	air cooler
E-101	is a part of	System-1

When something is classified by a particular kind of role, it is said to have such a role, although it still might be classified as something else. For example, a computer system might have a role as flow controller in a process control system, but this does not imply that the computer is classified as a flow controller.

Thus the statement:

system-1	has a role as a	flow controller (role)

does not imply that it *is* a dedicated flow controller (physical object). And the inverse is also true: If some physical object is classified as a screw driver, then it has an intended role, but its actual role may be something else (e.g. as tin opener).

Note: This holds for all classifications of a physical objects, especially in a design phase: classification as a pump implies that a physical object has an intended role of doing what a pump is normally designed to do, classification as a piping elbow implies that a physical object has an intended role of bending a flow, etc. Actually in some cases an otherwise classified physical object may execute the role. For example, a blinded tee might be installed which actually executes (= has) the role of (the) elbow.

The following example illustrates an example of a composition of a high level assembly. A process unit is composed, amongst others, of equipment systems, piping systems and control systems. These systems can be further (de-)composed into components. The top of such a composition hierarchy is illustrated in the following table.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
101	91,001	Pernis site	is classified as a	70,000	site
102	91,002	unit 6000	is classified as a	160,104	process unit
104	91,003	SGH plant	is classified as a	160,093	process plant
105	91,004	piping system-6000	is classified as a	160,092	piping system
106	91,005	control system 6000	is classified as a	70,068	control system
107	91,001	Pernis site	is a part of		
108	91,002	unit 6000	is a part of	91,003	SGH plant
109	91,003	SGH plant	is a part of	91,001	Pernis site
110	91,004	piping system-6000	is a part of	91,002	unit 6000
111	91,005	control system 6000	is a part of	91,002	unit 6000

Note that all assemblies and parts are classified by kinds of physical objects from the taxonomic dictionary and they are also defined as parts of higher level assemblies. Some systems cross boundaries of other units or systems, such as steam distributions systems. Furthermore, components may be a component in more than one assembly. For example, a control valve is typically a part of a piping system and also part of a control loop.

5.7.2 Equipment, machines, components and spare parts

Equipment and machines are physical objects that are assembled from various parts. Sometimes they are made according to a dedicated design, but in most cases they are fabricated in mass or in series from a generalized design. Dedicated designs are specifications of a *specific individual imaginary* physical object which is the basis for the fabrication of an actual physical object. Mass or series production is fabrication of actual individual physical objects according to the specification (design) of a *typical individual imaginary* physical object that is representative for a type of physical object or manufacturer's model.

Equipment *components* are physical objects that are intended to be part of a larger composed object, such as a piece of equipment. The term 'component' (or 'part') indicates that the physical object has a role as 'part' in a 'part-whole' relationship with a higher level aggregated object. Therefore component and part are subtypes of role.

As a consequence, a composition relation is defined as a relation between two objects, one in the role of part and the other in the role of whole. For example as follows:

Bearing-1 is a part of P-6050

Properties of the parts should preferably not be associated directly with the whole. For example it is semantically imprecise to say that ‘impeller diameter’ 300 mm is a property of centrifugal pump P-6050, because 300 mm is a diameter which is a property of an impeller. This means that for allocating properties to items, it is best practice to create a product model in which the assembly should be decomposed to the appropriate level and then to allocate the properties to the correct part. Thus for example:

D1	is an aspect of	Impeller of P-6050
D1	is classified as a	diameter
Impeller of P-6050	is a part of	P-6050
Impeller of P-6050	is classified as a	impeller

Nevertheless, for cases that the part is not defined, it is possible to use a short-cut relation to relate an aspect of a part to an assembly as follows:

D2	is an aspect of a part of	P-6050
D2	is classified as a	impeller diameter

Whereas the concept impeller diameter is defined as follows:

impeller diameter	is a specialization of	intrinsic aspect
impeller diameter	is by definition an intrinsic	diameter
impeller diameter	is by definition an intrinsic aspect of a	impeller

This definition enables software to verify the consistency between the two expressions and to conclude that both specifications express the same fact, so that $D1 = D2$.

In many cases people are used to apply the short-cut relation and intrinsic aspects, because they are used to consider properties of a part to be properties of the whole. Often it is expressed that way with as rationale that they are 'not interested' in the part. For example on equipment datasheets often intrinsic aspects appear such as ‘inlet size’ or ‘impeller diameter’ or material of construction for important parts, e.g. material of construction of the plunger of a plunger pump or of its mechanical seal. This means that if we don’t want to use short-cut relations we first have to define those items as part of the whole, and then we can specify the properties or materials of construction of the parts, for example as follows:

Part-1	is classified as a	plunger
Part-1	is a part of	P-6050
Part-1	is made of	stainless steel

5.7.2.1 Spare parts.

Spare parts, such as a spare plunger, are items that will potentially replace existing items. They are not primarily associated with the whole in which they are potentially installed, but, they are spare for a similar part. Therefore, a spare should be associated with the item for which it is spare as follows:

Part-2	is classified as a	plunger
Part-2	is spare for	Part-1

5.7.2.2 Specific designs, typical designs and standardized parts

The design of an item can be either a typical or a specific design, which means that the design process results in either a typical or a specific physical object, dependent on the question whether a dedicated specific design was made or a generic typical design.

For example, Ford makes typical designs for its cars, and Shell makes a specific design for nearly every facility, but both make or use typical designs for various (standard) components of those cars and facilities

Equipment is often to a large extent built up from standardized parts, such as standard piping components, cables, bearings, etc. Those standard parts do not differ from non-standardized parts, apart from the fact that their designs have non-specific names. For example, tag numbers are specific because they are the names of individual items. However, typical designs which are representative for several specific items cannot have tag numbers. Thus they will have more general names. But for the rest the definition of typical physical objects does not differ from the definition of specific physical objects.

So, the above explanation of how to define physical objects holds for specific (individual) physical objects as well as for typical physical objects. The difference can be seen from the role which they play relative to each other in a derivation relation:

P-101	is classified as a	centrifugal pump
pump-A	is classified as a	centrifugal pump

Often, a typical design is made and then the specific 'design' is said to be according to the typical design. This means that the specific physical object is 'derived from' the typical one. This is specified as follows:

P-101	is derived from	pump-A

A derivation relation implies that the properties of the typical object shall be copied to the specific one or they are used instead.

5.7.2.3 Actual items and design items

An item that is fabricated (similar to when a batch of fluid is produced) is called an ‘actual physical object’. The description of actual physical objects is not different from the description of the design objects, which are imaginary physical objects as they ‘exist’ only in the mind of the designers. In fact the only difference is that for designs, the values of the properties are derived from peoples mind (and may have a specified tolerance), whereas the values of the properties of actual physical objects are derived from observations, especially measurements (and may have measurement errors). But these few differences have minimal effect on the way the objects are described.

In many cases the properties of the actual object are only compared with the design during fabrication and inspection, but often not separately recorded for the longer term. Instead, in practice the design is often used as a reference for the shape and other properties of the actual equipment, assuming that the actual item conforms to the design of that item.

An actual physical object is another physical object than the (typical or specific) design object according to which the actual object is made. Their relationship is defined as follows:

X-12345 is a realization of V-6050

In this expression V-6050 is a design, or (specific or typical) intended physical object.

Note, actual physical objects are always specific, only intended physical objects (designs) can be either specific or typical.

Data which comes with an offer for delivery or with the actual delivery of an actual piece of equipment may be about (a further detail of) the specific design, or about a typical design of the item, but can also be really measured data about the actual item:

1. The supplier / manufacturer may have added *further detail to the specific design* as described on the requisition which he received from his customer (e.g. the facility owner or engineering contractor), in which case his design data is additional data for the same physical object as described on the requisition.

More than one supplier / manufacturer may make an offer and for that purpose they will make different sets of additional detail. In that case their sets of data should be documented as ‘alternative designs’, which are based on the same basic set (as in the requisition) and which is shared between the alternatives.

2. The manufacturer could also have made an *independent design for a typical individual physical object* which he offers among others for this case. In that case the design data are about a different physical object than the object on the requisition. The relationship between that typical design and the design as described on the requisition is then:

#1 is derived from #2

Note: This is a relation between 'required object' and a 'planned object', which is specified via a relation between two designs, one in the role of required, the other in the role of offered.

3. The manufacturer might also supply measured data from inspection reports, which are data about the actual physical object.

5.7.3 Networks and piping systems

Physical networks consist of nodes and branches. Nodes are themselves usually assemblies or even networks. What seems to be a simple node on a high aggregation level may appear to be a detailed network on a lower aggregation level. Thus assemblies and networks can have a role as nodes in a larger network.

Piping systems are typical examples of physical networks. Therefore the modeling of networks is discussed by explaining the modeling of a practical example of a piping system.

Figure 13 is a schematic graphical representation of a piece of a process plant that includes a piping system.

<i>Be aware that in this chapter we are dealing with the physical <u>piping objects</u> and <u>not</u> with the physical <u>graphical objects</u> that are part of a drawing. The graphical presentation of piping systems is discussed separately.</i>

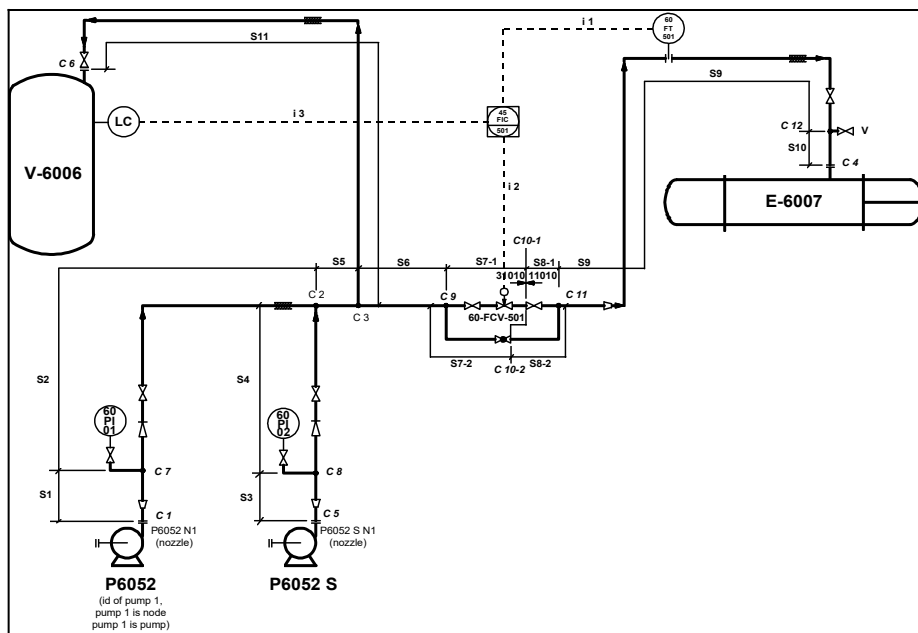


Figure 13, Example of a piping and instrumentation diagram (P&ID)

The scope of the piping system is indicated by the annotations with names S1, S2, etc. A piping system is intended to conduct a fluid when flowing from one or more locations to one or more others. The piping system typically comprises of piping materials, mechanical connections and some instruments. Types of mechanical connections are e.g. welded connections and flanged connections.

A piping system is a network of *piping branches* and devices and junctions that have a role as *nodes* in the network. The way they are connected specifies the topological aspect of the network. A piping branch is defined as a piping assembly between two junctions or nodes in a piping system. The nodes are the connection assemblies, which are assemblies of connection materials that are involved in the connections of branches in networks of items. At a high aggregation level a piece of equipment can serve as a node in a network. At a more detailed level the nodes are assemblies of connecting materials, such as bolts, nuts, flanges, welds, etc.

The topology of the network is modeled by defining the branches. The table below illustrates how a part of the above piping system can be modeled.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object

Equipment

201	101	P6052	is classified as a	130,206	pump

Nodes

207	104	P6052 N1 (C1)	is classified as a	10298	nozzle
208	105	junction C2	is classified as a	553637	junction

Piping branch

224	117	piping branch B1	is classified as a	10,086	piping branch
225	118	piping branch B2	is classified as a	10,086	piping branch
226	119	piping branch B3	is classified as a	10,086	piping branch
227	120	piping branch B4	is classified as a	10,086	piping branch

Routes via piping branches

230	117	piping branch B1	has as source	104	P6052 N1 (C1)
231	117	piping branch B1	has as destination	105	junction C2
232	118	piping branch B2	has as source	105	junction C2
233	118	piping branch B2	has as destination	106	junction C3
234	119	piping branch B3	has as source	107	junction C5
235	119	piping branch B3	has as destination	105	junction C2
236	120	piping branch B4	has as source	106	junction C3
237	120	piping branch B4	has as destination	108	junction C6

Note that the indication of nodes as sources and destinations of branches expresses a direction of the branch, typically representing an intended flow direction.

5.7.3.1 Pipelines

For the concept *pipeline* we find in the engineering practice various rather wide definitions, because they range from small to very large and can encompass a large variety of components. The definition of a pipeline does not necessarily imply that the end of a pipeline coincides with the end of a

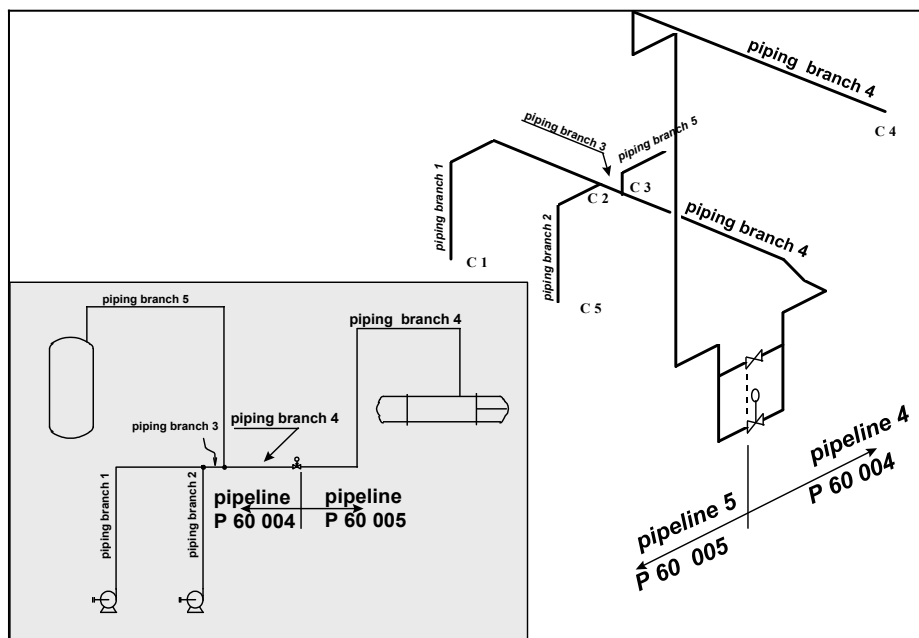


Figure 14, Pipelines versus branches and systems
piping branch.

For example, Pipeline 4 and Pipeline 5 in Figure 14 cover one piping branch (piping branch 4) of the piping network. Therefore we define a pipeline as an assembly of one or more succeeding *piping segments* (for a definition of piping segment see next paragraph), which segments may be parts of *one or more* piping branches. Nevertheless *pipelines* often can be defined on the piping branch assembly level. But the pipeline is not a prime object to describe the topology of a piping system. That is caused by the fact that occasionally the piping *segments* must be used to define the pipeline.

A pipeline is typically identified by a service id, a process unit id and a line id in one concatenated string as e.g. 'P60.004' and 'P60.005' etc. .

The following table illustrates how a pipeline is defined, using piping segments (and not using branches). For the specification of those segments see 5.7.3.2.

Pipeline 4

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
306	107	P60.004	is classified as a	10,082	pipeline
309	107	P60.004	is a part of	203	piping system-6000

Composition of pipeline 4, defined by segments

320	241	piping segment S5	is a part of	107	P60.004
321	242	piping segment S6	is a part of	107	P60.004
322	243	piping segment S7-1	is a part of	107	P60.004
323	244	piping segment S7-2	is a part of	107	P60.004

The topology of the piping system is defined by branches that form the routes between the nodes.

Routes via pipeline 4

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
330	107	P60.004	has as source	208	node C2
331	107	P60.004	has as destination	216	node C10-1
333	107	P60.004	has as destination	332	node C10-2

Note that Pipeline 4 has one source and two destinations. This implies that it provides two routes.

In case two branches constitute in fact one physical object, as is shown in the adjacent figure, the two pieces belong to a branch at one level higher. Like zooming out on a road map where certain details disappear.

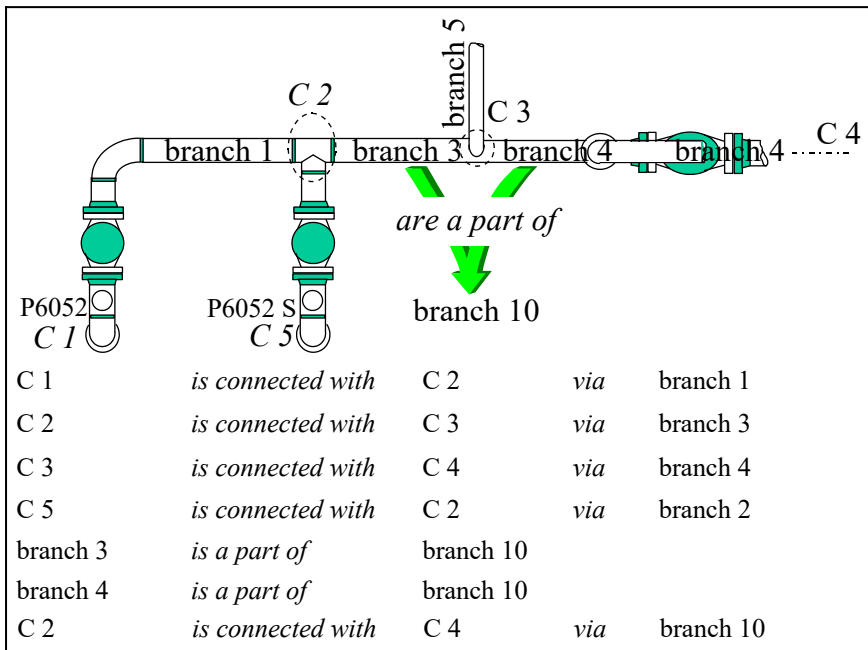


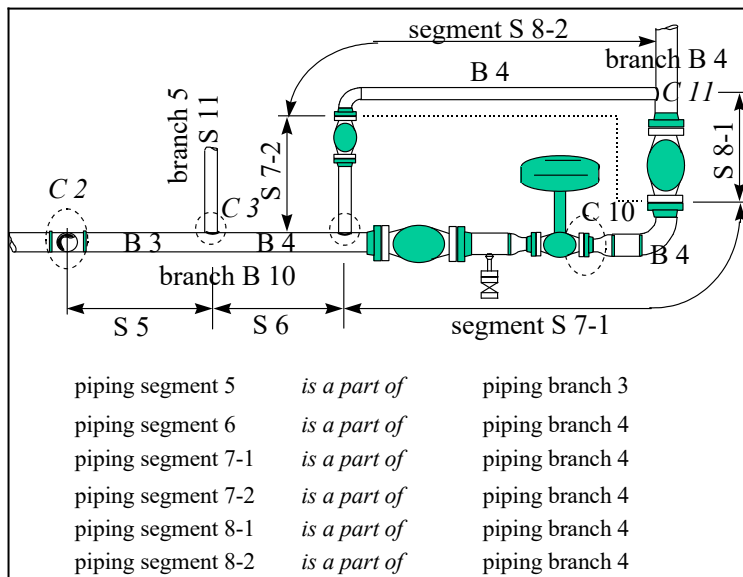
Figure 15, Addition of branches

As a consequence of a continuous evaluation of a design, new mechanical connections may be introduced in addition to the mechanical connections defined earlier. This may require that new piping branches have to be introduced as a part of existing piping branches. This is illustrated by the example in Figure 15, where branch 10 from node C2 to C4 is split into branch 3 and 4, because of the introduction of branch 5 during the progress of the design.

5.7.3.2 Piping segments

In engineering practice *piping segments* are defined in order to get manageable chunks of piping data.

The piping segments are assemblies of piping materials which scope is defined by certain constant attributes along the segment as shown as part of the line label. For example a considerable change in pressure, volume or temperature may impose changes to the design base for the piping items. This may or may not coincide with a *change in piping specification* *). A change in piping specification usually imposes a change of the pipeline tag label. This is illustrated in Figure 16.



*) Note, the term 'piping specification' (also called piping class) indicates a collection or kind of typical piping components, suitable for operating under specific process conditions.

The following table illustrates the specification of piping segments.

Piping segment

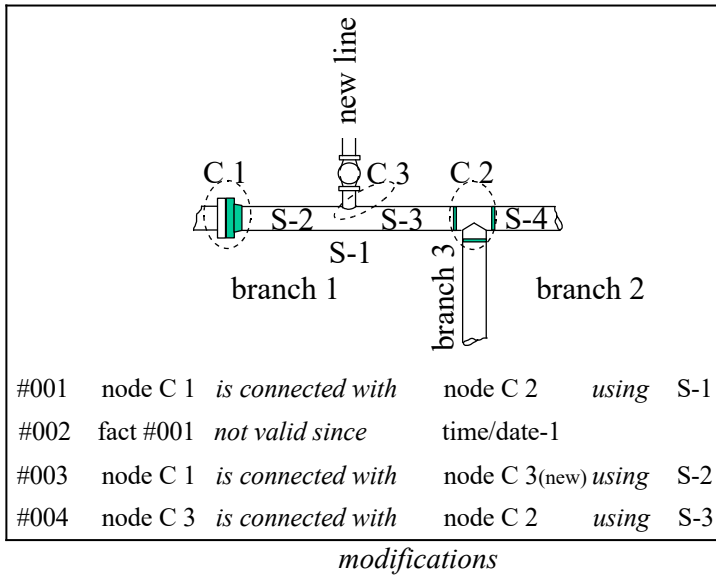
UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
237	150	piping segment S1	is classified as a	10,093	piping segment
238	151	piping segment S2	is classified as a	10,093	piping segment
250	150	piping segment S1	is a part of	117	piping branch B1
251	151	piping segment S2	is a part of	117	piping branch B1

Routes via piping segments

268	150	piping segment S1	has as source	104	P6052 N1 (C1)
269	150	piping segment S1	has as destination	213	node C7
270	151	piping segment S2	has as source	213	node C7
271	151	piping segment S2	has as destination	208	node C2

Figure 16, Piping segments in branches

Figure
17,
Change
control
of



In many cases piping segments begin and end at the *mechanical connections* such as branch connections. As a consequence of a continuous evaluation of a design, new mechanical connections may be introduced in addition to the mechanical connections defined earlier. New piping segments have to be introduced as a part of existing piping branches and/or piping segments while a segment defined earlier may lose its validity. Newly added lines become segments and/or branches as a consequence of the requirement that piping segments shall be a part of a piping branch. This requires proper change control.

5.7.3.3 Piping components

When we decompose the piping segments we arrive at a level where the *piping components* are still assemblies but those assemblies are known by the piping experts as items that can be purchased as a whole. Thus their type can be selected from existing catalogues or can be marked as components that need to be fabricated.

Piping components that are involved in connections are also called connection materials. They are assembled into objects that are usually classified as *connection assemblies*. Thus a connection assembly is an assembly of all items that are involved in a connection, each with its own role. For example a gasket is involved for sealing, bolts and nuts are involved to join the components together.

Examples of piping components are e.g. pipe flanges, valves, elbows, tees,

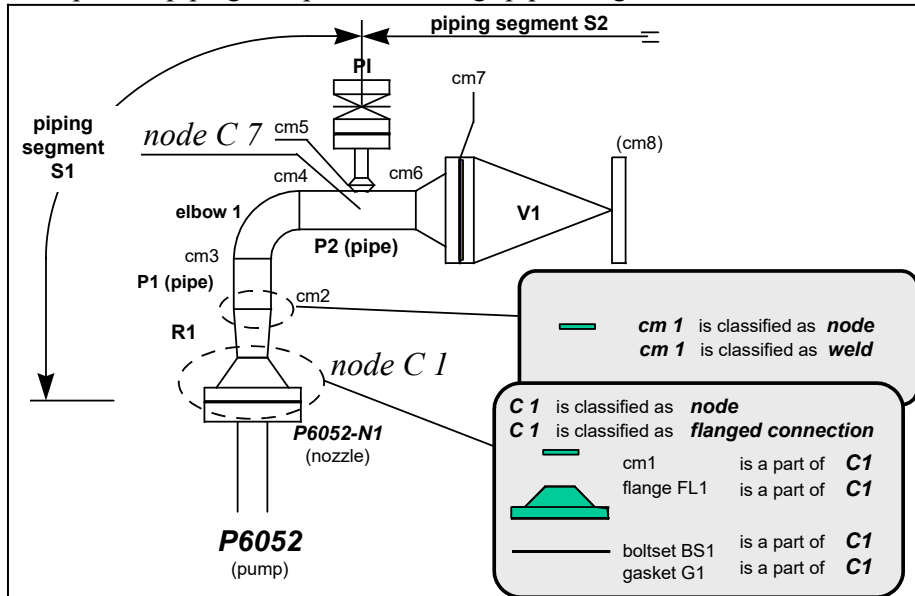


Figure 18, Piping components as parts of junctions and segments

reducers, spool pieces, (bare) pipes, etc. They are connected or coupled via *connection material*, such as gaskets, bolts, pipe flanges and welds.

The table below illustrates the classification of the physical objects (piping components and connection materials) and the routes as present in Figure 18.

Piping component specification

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
341	100	reducer R1	is classified as a	10,106	reducer
342	101	pipe-P1	is classified as a	10,077	pipe
343	102	elbow E1	is classified as a	10,028	elbow
344	103	pipe-P2	is classified as a	10,077	pipe
345	104	valve V1	is classified as a	820,034	check valve
346	105	PI	is classified as a	10,310	piping standard assembly
347	106	hole PI	is classified as a	730,010	hole
348	207	node C1	is classified as a	340,037	flanged connection
349	107	node CM1	is classified as a	340,091	weld
350	108	node CM2	is classified as a	340,091	weld
351	109	node CM3	is classified as a	340,091	weld
352	110	node CM4	is classified as a	340,091	weld
353	111	node CM5	is classified as a	340,091	weld
354	112	node CM6	is classified as a	340,091	weld
355	113	node CM7	is classified as a	340,037	flanged connection

Composition of piping components

363	341	reducer R1	is a part of	237	piping segment S1
364	342	pipe-P1	is a part of	237	piping segment S1
365	343	elbow E1	is a part of	237	piping segment S1
366	344	pipe-P2	is a part of	224	piping branch B1
367	345	valve V1	is a part of	237	piping segment S1
368	347	hole PI	is a part of	364	pipe-P2

Connections of piping components

370	341	reducer R1	has as source	207	node C1
371	341	reducer R1	has as destination	350	node CM2
372	342	pipe-P1	has as source	350	node CM2
373	342	pipe-P1	has as destination	351	node CM3
374	343	elbow E1	has as source	351	node CM3
375	343	elbow E1	has as destination	352	node CM4
376	344	pipe-P2	has as source	352	node CM4
377	344	pipe-P2	has as destination	213	node C7
378	344	pipe-P2	has as source	360	node C7
379	344	pipe-P2	has as destination	354	node CM6
380	344	pipe-P2	has as source	360	node C7
381	344	pipe-P2	has as destination	353	node CM5

As demonstrated with piping branches and piping segments, the connection materials that are used in a connection relation (e.g. nuts, bolts, gaskets or welds) can appear at several composition levels. In the next example a piping flange is used to connect pipes and, at a lower level, the weld is used to connect the piping flange to the pipe.

Further decomposition of connection assemblies

383	201	flange FL1	is classified as a	10,039	pipe flange
384	202	bolt set BS1	is classified as a	340,019	bolt set
385	203	gasket G1	is classified as a	340,039	gasket
388	383	flange FL1	is a part of	207	node C1
389	384	bolt set BS1	is a part of	207	node C1
390	385	gasket G1	is a part of	207	node C1

5.7.3.4 Features in piping connections

A (physical) feature is a physical object that is a part of an item and that cannot be isolated from the item nor exist on its own. Typically the boundary between a feature and the item is not well definable. For example: a surface, which can have a surface tension and a roughness, but has an undefined thickness or depth. Other examples are a ribbon, a groove, an integrated lifting lug, etc.

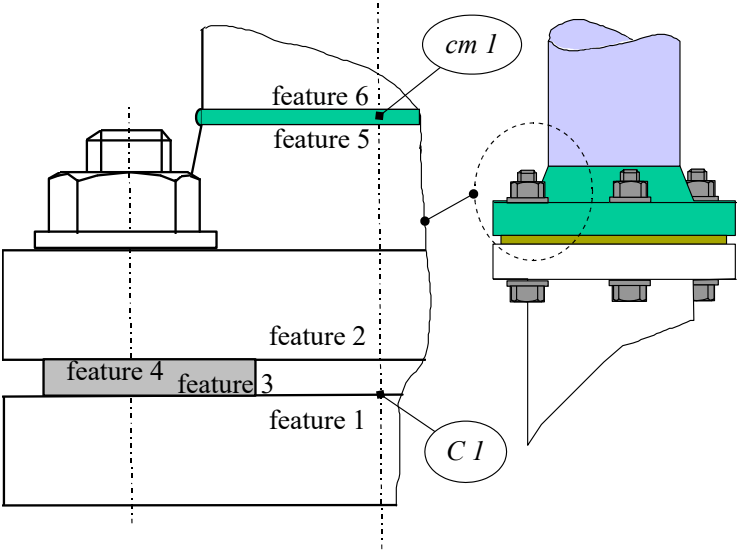


Figure 19, Surfaces as features

Features can be, but need not be connected to each other. They may also touch each other. Features can have properties such as dimensions, roughness and shape. This is illustrated in Figure 19.

So, optionally the use of *features* in the connection, that are parts of the piping component, can be made explicit via a connection relation. At least two are connected.

Physical features

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
392	204	feature-1	is classified as a	10,040	flanged end
393	205	feature-2	is classified as a	10,040	flanged end
394	206	feature-3	is classified as a	10,279	gasket contact face
395	207	feature-4	is classified as a	10,279	gasket contact face
396	208	feature-5	is classified as a	10,349	welding end
397	209	feature-6	is classified as a	10,099	plain end

398	210	node Cc-1	is classified as a	251,722	node

Features as part of components

400	392	feature-1	is a part of	202	P6052-N1
401	393	feature-2	is a part of	383	flange FL1
402	394	feature-3	is a part of	390	gasket G1
403	395	feature-4	is a part of	390	gasket G1
404	396	feature-5	is a part of	383	flange FL1
405	397	feature-6	is a part of	341	reducer R1

Use of features in a connection

410	204	feature-1	is touching	206	feature-3
413	207	feature-4	is touching	205	feature-2

5.7.4 Electrical

Electrical engineering deals with physical objects which process electrical currents in some way or another. As a result this subject area deals with physical object that perform or enable the processes, other physical objects that are subject to those processes, such as electro-magnetic fields and electric currents (or waves), as well as the electrical processes, such as transformation, generation and conversion of electricity. It also deals with the algorithms which describe those processes (occurrences). The Formal English dictionary contains concepts in all those areas.

The relationship between electrical equipment and the electrical current can be expressed in formal languages in analogy the relationship between mechanical equipment and fluids or other process material processed by them. So, a flow direction in an electrical network is described via the electric currents and their roles as inputs to and outputs from electrical processes. The topology and decomposition of an (physical) electrical network is described in analogy to a piping system as described above.

For the definition of electrical equipment and their components the dictionary used among others standards such as IEC-60050. However, note that definitions in those standards need to be adapted in order to comply with a strict hierarchy of concepts as is required for a taxonomic dictionary.

5.7.5 Instrumentation & Control systems

Process control systems, which include also safeguarding systems, are the hardware systems that perform control activities according to the control

logic (for the modeling of processes see chapter 7). The control systems can be decomposed into measuring devices, controlling devices and final control devices, together with various types of wires to connect them and thus making the topology of a control network. This is analogous to electrical systems (or networks) and piping systems.

The ‘process materials’ that are processed in the systems are the signals, which are physical objects modulated such that they carry information (see par. 5.7.8).

For the definition of the instrumentation and control hardware components in the formal dictionary various standards are used, such as IEC-50, IEC-70, ISA-S20, ISA-S51.1 and IEC 770.

5.7.6 Infrastructure and Buildings

The modeling of infrastructural objects, such as roads, viaducts, etc. and buildings is basically the same as for other kinds of physical objects as is described in the previous paragraphs. A special case is that the subdivision of spaces. Spaces are defined as physical objects, because they have a shape and exist in space and time. They may be filled with air, they may be subdivided and object may be placed in them. See also par. 5.7.9 Holes and spaces.

Kinds of items in the civil area are for example road, structural items such as profile, support structure, etc. and kinds of architectural items are for example control room, wall, door, building, etc.

The construction materials, such as various types of steel, concrete and plastic are defined as specializations of substance as is discussed in par. 5.5.4.

An example of the definition of a T profile beam is as follows:

T1	is classified as a	T profile
T1	is made of	carbon steel
etc.		

5.7.7 Waves and energy

Waves, such as electromagnetic waves, radiation, light and sound behave according to the laws of physics and therefore are (subtypes of) physical object. They behave like fluids, but can also flow through vacuum. Rules that apply for the modeling of physical objects in general also apply for waves.

Energy appears in several forms, such as electrical energy, thermal energy (heat), mechanical energy (kinetic and potential energy), chemical energy, etc. Energy can be described as a property of a physical object.

For example, heat (radiation) can flow through the wall of pipes in a heat exchanger. Such heat is a physical object that is transferred from one place to another. It behaves as if it is a fluid which flows through the pipe wall. In those cases the heat waves carry the energy. This can be described as follows:

Q1	is classified as a	heat	
Q1	has aspect	h1	
h1	is classified as a	heat flow rate	
h1	has on scale a value equal to	1000	J/s

Note, that the size of the batch of heat, being the total quantity of heat, might not be known nor specified, because it may depend on an unspecified duration of the heat transfer process.

Energy can be carried by waves, but is usually carried by a material physical object. For example, thermal energy is a form of energy that may be carried by a fluid. In those cases the energy is described as a property of the physical object. For example, the enthalpy of a fluid (in J/kg) describes an energy content of the fluid. This can be described as follows:

B2	is composed of	water	
B2	has aspect	h2	
h2	is classified as a	enthalpy	
h2	has on scale a value equal to	1000	J/kg

When energy is converted from one type or energy level to another in the same carrier, then the physical object that carries the energy has two properties, one before the conversion process and one after. This means that it has either two properties at two different locations or it has one property with two values at different moments in time.

For example, when heat is extracted from a fluid A and transported through the pipe walls in a heat exchanger to another fluid B, then the energy is transformed from enthalpy h1 of the fluid A into heat flowing through the 'heat transfer area' to increase the enthalpy h2 of fluid B.

Continuous steady state processes are typically modeled using two fluids A and B each with two enthalpies at different locations, one enthalpy at the inlet before the heat transfer (h11 and h21) and other enthalpies at the outlet of the heat exchanger after the heat transfer has taken place (h12 and h22). In such a steady state process the time difference between the two enthalpies is usually neglected (unless the dynamic behavior is modeled for process control). Batch processes are typically modeled using two fluids A and B which each has one enthalpy with two different values at different moments in time. Typically the modeling neglects a possible difference in location of the fluids before and after the heat transfer.

The processes may also be modeled with four fluids, as if fluid A and B each were two different fluids, two (A1 and B1) being the inputs and two others (A2 and B2) being the outputs of the heat exchanger.

An example of a description of fluid A in a steady state process described with two enthalpies (h1 and h2) at two different locations (N1 and N2) is as follows:

A	is subjected to	Heating of A
Heating of A	is classified as a	heat transfer
E-101	is performer of	Heating in E-101
E-101	is classified as a	heat exchanger
A	has aspect	h1
A	has aspect	h2
h1	is classified as a	enthalpy
h2	is classified as a	enthalpy
h1	is located at	N1
h2	is located at	N2

The nozzles that indicate locations are specified as follows:

N1	is classified as a	inlet nozzle
N1	is a part of	E-101
N2	is classified as a	outlet nozzle
N2	is a part of	E-101

The conversion process to extract heat from the fluid and to generate a heat flow is a physical process. In case of simulation of such a process, the execution of the simulation calculates the heat flow rate and heat flux from the enthalpy difference and the flow rate of the fluid. The relationship is therefore documented in an algorithm and possibly in the definition of the input and output data for that algorithm. Such an algorithm is a correlation that is a higher order relation in which the various properties are the parameters.

For a further explanation of the description of physical processes and algorithms, see chapter 7.

5.7.8 Signals

Control logic is usually described by control logic diagrams. The boxes on those diagrams represent control processes, which are subtypes of ‘occurrences’ in the dictionary. The lines between the boxes are ‘signals’ that have a role as output from one control activity and the same signal has a role as input to one or more subsequent control activities. These signals are physical objects (usually modulated waves) which represent information,

such a state (property) of an equipment (e.g. on, off, open, close, running, stand-still, etc.), or a property of a fluid (e.g. pressure, temperature, volume flow rate, level, etc.). The fact that the arrows in a logic diagram are directed and point from one control activity to another means that the signal that represents the (measured) equipment state or fluid property is output from one and input to another control activity. These roles of the signals define the topology of the control network.

For example, a flow regulation to control a level is described by two properties: a flow rate of a fluid and a level of another fluid. The level of fluid S1 is output of a measurement activity and input to a control activity (e.g. described by a PID control algorithm), the required flow rate of fluid S2 (a set point, being a required property) is output of that control activity and is input to the actuation activity. This means that an arrow on such a logic diagram represents a property of a fluid. For example, signals that represent properties are described as follows:

S1	is composed of	water
S1	has aspect	L1
L1	is classified as a	level
Signal-1	is classified as a	signal
Signal-1	represents.	L1
S2	is composed of	water
S2	has aspect	F1
F1	is classified as a	required flow rate

The arrow on the diagram that represents the level of S1 is modeled as follows:

Signal-1	is output of	measure L1
measure L1	is classified as a	level measurement
Signal-1	is input in	Control-1
etc.		

Similarly for signals that represent the state of a piece of equipment (being running or not-running) at a particular moment in time:

M-101	is classified as a	electric motor
M-101	has aspect	state-x
state-x	is classified as a	running state
Signal-1	is classified as a	signal
Signal-1	represents	state-x

Signals can be seen from the perspective of their content, and then they are 'messages'.

However, those messages can only exist in a physical form of an information carrier. So, in practice *signals are physical objects* and more precise, they are usually modulated waves, being a physical object in such as shape or movement, that that shape or movement represents a message. And the message is an expression of the fact that a property has a particular value on a particular scale.

Thus, although it is modeled that a signal represents a property, it actually presents the fact that is the quantification of the property, which is expressed by a relation. That expression forms the information.

For example, sound is a modulated wave moving through the air in such a frequency and amplitude (strength) or sound pressure level that it can be heard and interpreted.

sound-1	is classified as a	voice
sound-1	represents	‘this is a message from ...’

Example 2: a control signal can represent information for example that a motor is running or not running. At an early stage of a design it can be neglected how this is interpreted from the physical signal as follows:

motor-1	has aspect	running state-x
running state-x	is classified as a	running state
signal-1	is classified as a	control signal
signal-1	represents	running state-x

A signal usually represents a property value of a physical object. At a later stage of detailed design, the signal is specified in detail. Then it is specified that the represented property is transformed by the measurement process into a property (a modulation) of the signal.

For example, the running state of a motor, being ‘not running’ or ‘running’, is measured. This results in an electrical signal that is modulated in the measurement process such that it has an electrical tension of 0 or 6 mV dependent on the state.

The relationship between the state and the electrical tension is defined by the measurement process. The resulting voltage that represents the value of the state of the motor can be defined directly (using a short-cut relation type).

For example, assume that the state of a motor can have only two values, not-running and running, then this can be described as:

running state-x	is defined by list	not-running, running
signal-1	is classified as a	electrical signal
signal-1	has aspect	voltage-x

voltage-x	is classified as a	electrical tension	
voltage-x	is defined by list	0, 6	mV
voltage-x(1)	represents	running state-x(1)	
voltage-x(2)	represents	running state-x(2)	

Note 1: A parameter is a subtype of a property and a list is a structured collection of values.

Note 2: It is more accurate and more flexible to describe the relationship between the running state and the voltage via the measurement process, which has the motor shaft (property) as input and the electrical signal (voltage) as output. This transformation activity defines the relationship between the two properties. If both methods are used simultaneously there is a danger of inconsistency!

5.7.9 Holes and spaces

Holes and other spaces are physical objects which are enclosed by boundaries which are either theoretical surfaces or real other physical objects. Those enclosed spaces may be empty but need not necessarily be empty.

For example, a void for pulling a tube bundle out of a heat exchanger is a 3D space which is not bounded by another real physical object (apart from the floor, which is a part of the earth). Nevertheless the theoretical surfaces which define those boundaries are surfaces whose actual location can be pointed at in 3D space. Those boundaries are therefore also physical objects (for further explanation of those surfaces see par. 5.7.10 Features). A void is intended to be an empty space, although it might be allowed to place movable objects in the void.

Example 2: a room, e.g. in a control building, is a space which is bounded by real physical walls, a floor and a ceiling. It is not intended to be completely empty; on the contrary it is intended to provide space for placement of material and location of people. However, a room can also be defined as an assembly of a space and the surrounding walls, floor and ceiling:

R1	is classified as a	room	
R1	has aspect	V1	
V1	is classified as a	volume	
V1	has on scale a value equal to	160	m3
F1	is a part of	R1	
F1	is classified as a	floor	
F1	has aspect	A1	

A1	is classified as a	area	
A1	has on scale a value equal to	60	m2

Example 3: a window is bounded by the surface of the hole in a wall and if the ‘thickness’ of the window is needed to be defined, then it is also bounded by two theoretical surfaces. The window can be seen as part of the wall and the position of the window is defined by a placement relative to the co-ordinate system of the wall, as follows:

W1	is classified as a	window	
W2	is classified as a	wall	
W1	is a part of	W2	
W1	is placed relative to	W2	
W1	has aspect	w1	
w1	is classified as a	width	
w1	has on scale a value equal to	1.80	m
W1	has aspect	h1	
h1	is classified as a	height	
h1	has on scale a value equal to	1.20	m

For details about relative placement of physical objects, see par. 9.5.4, Placement of Physical Objects.

Thus holes and other spaces are described as normal physical objects, which have a shape and volume and possibly other properties. They may possess physical features, which are surfaces that describe the boundaries of the space. Those surfaces have their own shape and possibly other properties.

5.7.10 Features

The term feature is often used in various ways, it can indicate a function or role which a physical object can play or it can be a physical object itself. Here we deal with the latter. For clarity they are called physical features.

Physical features are subtypes of physical objects which are typically non-separable parts from larger physical objects. For example, a raised face of a flange, a lifting lug on a motor body, a surface, a physical point, an edge, a centre line or a tangent line of a vessel (which actually is a surface).

Because they are physical objects themselves they are described just as normal physical objects. The only difference is that they are usually not fabricated as separate items and thus they are not connected to a larger physical object, but they are in integral part of it. Therefore, the relation between physical features and their larger whole physical objects is classified by a separate kind of relation, denoted by the phrase <is a feature of>.

For example, a nozzle at the shell of a vessel is not a physical feature, because it is fabricated as a separate component and is welded to the vessel. Thus it is specified as a separate part of the vessel. However, an inlet nozzle of a pump is typically casted as integral part of the body of the pump. The latter nozzle is a physical feature, which is a feature of the body. The two cases are illustrated by the following expressions.

N1	is classified as a	nozzle
N1	is a part of	V-6060
N1	is connected to	shell-1
N2	is classified as a	nozzle
N2	is a feature of	body-1
body-1	is a part of	P-6052

5.8 Verification rules for correctness of product models

Correctness of product and process models, thus correctness of collections of formal language expressions *about individual products* requires compliancy with at least the following rules.

Rule 1: Explicit classification

Each individual thing shall be classified by at least one explicit classification relation with a kind of thing (concept or qualitative aspect), whereas such a kind of thing shall be:

- Either selected from the formal dictionary, or
- Properly defined as a subtype of a concept in the formal dictionary (possibly as an element of a domain dictionary), whereas each individual relation is classified by a kind of relation (relation type) that is also selected from the formal dictionary.

How to verify rule 1:

- Verify for each used individual thing whether it is classified by a classification relation (or a subtype of it) with a concept (UID) that is a subtype of the concept ‘anything’. A concept is a proper subtype if it is related through a sequence (chain) of specialization relations (or through subtypes of specialization relations) with more general concepts, which sequence, when following the hierarchy, terminates at the concept “anything”. If the top concept is not ‘anything’, then an error shall be

generated including a report about the concepts that actually terminates the sequence.

This can be verified using software with built-in verification rules in combination with the formal dictionary. For example this can be verified by importing the expressions in the Gellish Search Engine.

Rule 2: Grammatical correctness

Each relation between things is classified by a kind of relation (relation type). This can be specified in columns 60 and 3 of an Expression Table or equivalent format. Such a kind of relation defines the kinds of roles that are played by the related things in a relation of such a kind. Each role player shall be able to play the role that it is required to play according to the kind of relation. The (required or specified) nature of a role player shall not be in conflict with the nature required by other relation types in which the role player plays a role.

The formal language definition in the upper ontology section of the dictionary (TOPini) includes “facts about conceptual relations” that define the relation types of the formal language. These definitions also specify the required kinds of roles and the kinds of things that can play those kinds of roles (which implies that also their subtypes can play those roles).

Example 1: the formal language expression ‘C is a specialization of D’ is only correct if C is a concept, because a <is a specialization of> relation requires as first role a subtype, whereas a subtype role can only be played by a concept (being ‘concept’ or a subtype of ‘concept’). The relation type requires as second role a supertype, which can also only be played by a concept. So, D must also be a concept.

Example 2: the formal language expression ‘A is by definition a performer of a B’ is only correct if A is a subtype of physical object, because the facts in the formal language definition specify that an <is by definition a performer of a> relation requires as first role a performer and it also specifies that a performer role can only be played by (a subtype of) physical object. The relation type requires as second role a “performed”, which is a role that can only be played by (a subtype of) occurrence. So B must be (a subtype of) occurrence. These conclusions about the nature of A and B shall not be in conflict with the nature required by other relation types in which A or B plays a role.

How to verify rule 2:

Verify for each used relation type whether the relation type exists in the formal language definition in the upper ontology section of the dictionary. Proprietary extensions of relation types shall be verified separately.

This can be verified using software with built-in verification rules in combination with the formal language definition.

Verify the consistency of the object type of the left hand and of the right hand object according to various relations in which those objects are involved as follows: When something is related to another thing by a relation of a particular kind, then it can be derived from the definition of the relation type in the formal language definition what kind of thing it must be. This can be determined as follows: according to the formal language upper ontology the relation type requires a first kind of role and a second kind of role. This is either directly specified for the relation type or it is inherited from its supertype relation type. The formal language upper ontology also specifies which kind of thing can play such a kind of role. This can also be specified either directly or by inheritance from its supertype kind of role. When the object is a left hand object and the formal language phrase for the relation type is a normal phrase (or both the opposite), then the thing shall be of the kind of thing that can play the first kind of role. When it is a right hand object or an inverse phrase then it shall be of the kind of thing that can play the second kind of role, etc.

Conclusions about the kind of thing that are derived from various relations for something and its explicit classification relations or specialization relations (or their subtypes) shall not be in conflict with each other. There is no conflict if the conclusion about the kind of thing drawn from the classification or specialization relation(s) is the same or a subtype of the kind that is required by a relation type.

This should be verified by rules that are the built-in the software.

Note: If the specialization hierarchy of concepts is incomplete (the rule that each concept shall be defined as a subtype of one or more supertype concepts is not satisfied; see chapter 10 about definition models and the dictionary), then this verification is impossible.

Rule 3: Semantic consistency

The relation types that classify the relations in formal language expressions determine the nature of the things that play the roles in the relation. Those

natures shall be consistent with the natures that are required by other relations in which the same thing plays a role.

Example 1: from the formal language expression ‘C is a specialization of D’ it can be concluded that C is a concept. If there is another expression that states that ‘C is a part of E’, then from that relation it can be concluded that C is an individual thing. These two expressions are thus inconsistent as C cannot be a concept and at the same time be an individual thing.

Example 2: from the formal language expression ‘A is a performer of B’ it can be concluded that B is an occurrence. If another expression exists that states that ‘B is a performer of F’ then the two expressions are inconsistent, because B cannot be an occurrence as well as a physical object.

How to verify rule 3:

There is no conflict between a conclusion about the kind of thing drawn from one relation type and a conclusion drawing from another relation type for the same thing, when the required kind of thing is the same or is a subtype or supertype of the kind that is required by the other relation type. In other words they are in conflict when the conclusions about the kinds of things belong to different branches of the specialization hierarchy.

This should be verified by rules that are built-in software.

Rule 4: Completeness of contextual facts

A formal language expression, for example in an Expression Table, shall contain contextual facts that comply with the definitions of contextual facts according to one of the subsets that are defined in the “Gellish Syntax” definition document [Ref. 4]. Each contextual fact (each field on each row in an Expression Table) shall have values that comply with the rules defined in that document.

How to verify rule 4:

The software shall verify whether a value in a field is present when that is obligatory and whether the value satisfies the format constraints for the field.

5.9 Summary of relation types for product models

The following relation types form the core for information models about individual products. Additional relation types for the modeling of other kinds of information, for example for modeling the content of documents or for

modeling of processes in which the products are involved can be found in the formal language taxonomic dictionary section with relation types.

Note that the relation types that are denoted below can also be denoted by inverse phrases. When an inverse phrase is used then the left hand object and right hand object in an expression shall be swapped.

UID of relation type	Relation type name	Description
1,225	is classified as a	is a relation that relates an individual thing to a kind of thing. Typically the left hand object is an individual physical object in the product or an individual document and the right hand object is a kind of thing (a class) selected from the formal dictionary.
1,190	is a part of	is a relation that relates two individual objects, that specifies that the left hand object is a part of the right hand object.
4,856	is a subset of	is a relation that relates a collection to a bigger collection, that specifies that all the elements of the left hand collection are also elements of the right hand collection.
1,227	is an element of	is a relation that relates an individual thing to a collection that specifies that the left hand individual thing is an element in the right hand collection.
1,393	is a next version of	is a relation between two individual objects, that specifies that the left hand object is a next version of the right hand object. Typically the right hand object is replaced by the left hand object.
5,046	is described in	is a relation between an individual object and a document (information) that specifies that the left hand individual object is described in the right hand document.
4,996	is presented on	is a relation between a document (information) and a physical information carrier, such as an electronic file, that specifies that the left hand document is presented on the right hand information carrier.
1,313	is installed at the place of	is a relation between a real installed item and an imaginary item that specifies that the left hand real item is installed at the location of the right hand imaginary item. Typically the real item is indicated by its asset registration number (serial number) and the imaginary item is indicated by a tag name.
5,396	is a model of	is a relation between a kind of thing that is specified in detail and a kind of thing that is less specified that indicates that the left hand (manufacturer's) model is a specified model that is a specialization of the more general right hand kind of thing.
4,761	is performer of	is a relation between an individual physical object and an occurrence that specifies that the physical object makes the occurrence to happen.
4,785	is an input in	is a relation between an individual physical object and an occurrence that specifies that the physical object is consumed or used by the occurrence. Typically a material, energy or signal.
4,786	is an output of	is a relation between an individual physical object and an occurrence that specifies that the physical object is produced by or result of the occurrence. Typically a material, energy or signal.
4,760	is subject in	is a relation between an individual physical object and an occurrence that specifies that the physical object as subjected to the occurrence. (inverse: <has as subject>)
1,388	is a predecessor in time of	is a relation between two occurrences that specifies that the left hand occurrence has been finished before the right hand occurrence may start. (inverse: <is a successor in time of>)
1,313	has aspect	is a relation between an individual thing and an individual aspect that specifies that the individual thing possesses the aspect.
5,025	has on scale a value equal to	is a relation between an individual aspect and a number that specifies that the individual aspect has a magnitude that is quantified by the number on a scale, whereas the scale is separately specified in an additional column.
5,020	is qualified as	is a relation between an individual aspect and a qualitative aspect that specifies that the individual aspect is qualified by the qualitative aspect.
5,423	is made of	is a relation between a kind of thing and a kind of substance that indicates that the kind of thing is made of material which substance aspect is of that kind of

	substance. For example, SS bolt <is made of> stainless steel.

Table 4, Basic relation types for product information models

6 Application system usage examples

This chapter shows how software can be used to create semantic information models as collections of formal expressions that are stored in a universal semantic database and imported and exported as system independent formal language files, expressed in a universal expression format. It also demonstrates intelligent searching, retrieval and inspection of documents and data in a semantic information models.

6.1 The Gellish Communicator

The Gellish Communicator is a generic software system with a universal database. The system is built to demonstrate the power and benefits of expressing information in any of the formal languages from the Gellish family of formalized natural languages, among which Formal English and Formal Dutch (Formeel Nederlands) and according to the Gellish Information Modeling Methodology. Applying a common taxonomic dictionary and expressing information in a computer interpretable, universal and system independent formal language has various benefits. Examples of benefits that are illustrated by the Communicator are:

- Exchanging information between various systems of different parties without a need for data conversion;
- Combining and integrating information from multiple sources, including consistency management;
- Intelligent querying and applying reasoning with the rules of logic that are included in the definition of the formal language;
- Expressing and use of knowledge and requirements for generating design proposals in computer enhanced design as well as verification of products and processes on compliance with expressed requirements;
- Harmonization of data in systems.

The Communicator system is also meant to demonstrate the possibilities of simultaneous use of terminology of different companies, branch organizations or national and international terminology standards in various languages when they are combined with the Gellish language definition.

A formal language definition that is available as a computer interpretable basic language defining ontology and a number of domain specific taxonomic

dictionaries [Ref. 4] is imported in the Communicator and is the basis for all models.

6.2 Functionality

The Communicator can be applied in numerous areas in the whole lifecycle of products and processes. For example, it can be applied in applications for recording and/or searching and application of knowledge and information about whatever subject, or of requirements and the generation of proposals for the design, manufacturing, use and maintenance of products and processes, or for automated verification of product and process models guided by knowledge and requirements that are expressed in a formal language.

Communication between parties and systems is facilitated by the Communicator because product and process models that are expressed in a formal language can be exported and imported to and from files in a standard format. The information in the files can be exported in any of the languages of the Gellish family of formal languages, independent of the formal language in which the models were originally created. The files can be sent to third parties that apply one of the formal languages of the family and then be interpreted and used by them. The Communicator can verify imported files on correctness of the use of the formal language and on the compliance with imported knowledge and requirements. Furthermore, the information can be presented as if it is integrated with other models in the Communicator database. All these functionalities facilitate interoperability between systems and projects.

The Communicator can communicate with its users via a graphical user interface that can present information in various languages. In first instance English and Dutch (Nederlands) are available. Figure 20 illustrates that English is selected as GUI language. The GUI language is independent of the formal language that is used for the models.



Figure 20, Start screen of the Communicator

The Communicator can only function in a formal language when it starts with reading a formal language defining ontology. The core of the language definition is therefore read at the start of the system. The start-up file or database that is read is reported in the ‘Messages and warnings’ section of the user interface, as is also illustrated in Figure 20. Apparently the read language defining ontology file does not contain ‘intentions’ for the expressions. Therefore it is reported that the default value ‘statement’ is used, which means that the expressions are interpreted as statements (as is intended).

6.2.1 Import, combination and verification of information

After reading the basic language definition (the basic language defining ontology) the Communicator can extend the used language definition by importing one or more public or proprietary domain taxonomic dictionaries and terminologies, possibly of multiple languages. This is done by selecting the option ‘Import’, followed by selecting one or more Gellish Expression Format files. The richer the imported language definition and the more domain dictionaries are part of it, the semantically richer and more powerful the system becomes, without the need to modify the software!

The imported language definition, with the dictionaries forms the formal language in which other knowledge, requirements and product and process models can be expressed. In other words, they provide the expression capabilities for producing and interpreting other expressions in the formal

language and they contain the criteria for verifying of expressions on correctness of language usage and contain the rules for their interpretation.

Depending on the intended application, it is valuable to import one or more knowledge libraries or requirements models and definitions of product types or catalogues of various origins before product and process models are created or imported. During the import the expressions are verified on correctness and on consistency of the new information with earlier imported information in the various files.

Finally, object models, product models and process models of individual objects, products and processes can be imported. If the unique identifiers (UIDs) in those various models are well managed and the models consist of expressions that correctly apply the formal language definition, then their combination and integration does not need any data conversion. The whole language, knowledge, requirements and models of individual things forms an integrated information model of which the consistency and coherence can be queried and used.

6.2.2 Formulating queries with conditions

As all information is integrated, the Communicator can not only be used for querying information models about individual things, products and processes, but also for querying the language definition and its dictionaries as well as for querying knowledge and requirements models about those products and processes.

The functionality for querying is illustrated below, using an imported model of a road, as is described in [Ref. 4], together with imported definitions and knowledge about roads.

A query on a database can be accompanied by one or more conditions, which the selected information should meet. Conventional systems typically require that such conditional queries are expressed in a dedicated query language, such as SQL. Therefore, usually such queries are formulated by specialists. In Gellish formal languages, and therefore also in the Communicator, such queries and conditions are specified in a similar same way as how models are expressed. The only difference is that in queries the searched things are represented by one or more ‘unknowns’. For searching such unknowns it is required to specify information that is known about the unknown(s). For example, roads that are located in New York can be found by specifying that an unknown called ‘what’ is classified as a road and adding the condition that the unknown(s) is/are located in New York. This query can be expressed as a search model in Formal English as follows:

what	is classified as a	road
what	is located in	New York

In this query the term ‘what’ denotes one or more unknowns. Note that this model is precisely the same as a specification that a particular road is located in New York, apart from the fact that the name of the road is replaced by a term that denotes one or more unknowns. Such an unknown may be denoted by any unknown term and there may be more than one unknown. For example, ?, ?-1, ?-2, what-1, what-2 and road-x are all valid names for unknowns. The term is a free text, for which holds that reuse of the same term in succeeding expressions in a specification of conditions indicates that the condition(s) are applicable for the same object or objects. When entering an unknown, the Communicator allocates unique identifiers (UIDs) to the unknowns. Such UIDs for unknowns are in the range 1 through 99, conform the reserved range for unknowns according to the formal language definition. Therefore, users can interpret from an allocated UID in that range that the Communicator interpreted something as an unknown.

Figure 21 presents a window of the Communicator in which the above query with condition is entered on the first line with data entry fields.

Query

Left hand term

Relation type phrase

Right hand term

Unit of measure

?

is classified as a

road

1

1225

40153

0

Definition of lh object:

☒ Case Sensitive

☒ First Char Match

Conditions:

?

is located in

Province Groningen

Select one of each of the following options:

Left UID	Name	Kind	Community	Language
1	?	unknown kind	Model	Dutch

Confirm

Verify model

Relation UID	Name	Kind	Community	Language
1225	is classified as a	categorization relation	Formal English	English
1225	is classified as	categorization relation	Formal English	English

Right UID	Name	Kind	Community	Language
40153	road	strip	civil technology	English
44133	road with separated lanes	road	building technology	English

Messages and warnings:

Query : ? (1) is classified as a (1225) road (40153)

Candidate answers:

statement 1 about [road]: <N52> <is classified as a> <dual carriageway>.

statement 2 about [road]: <N51> <is classified as a> <road with three separated carriageways>.

statement 3 about [road]: <N5> <is classified as a> <dual carriageway>.

Condition 1: ? (1) is located in (5138) Province Groningen (203)

Answer:

Candidate 1: <N52> <is classified as a> <dual carriageway>

No, the condition is not satisfied that: N52 is located in Province Groningen

Candidate 2: <N51> <is classified as a> <road with three separated carriageways>

Indeed: N51 is located in Province Groningen because...

N51 <is located in> Groningen

Groningen <is located in> Province Groningen

Satisfied 1: <N51> <is located in> <Province Groningen>

Candidate 3: <N5> <is classified as a> <dual carriageway>

No, the condition is not satisfied that: N5 is located in Province Groningen

Warning: Aspect N51-12 (146) has no value.

Figure 21, Query with condition

The second line of Figure 21 provides UIDs that are generated by the Communicator. These UIDs make clear that the Communicator has recognized the phrase of the relation type as well as the term ‘road’ because it presented their UIDs according to the dictionary. Possible homonyms or terms in which the same characters ‘road’ appear are mentioned as options, under the heading ‘Select one of the following options’. Together with those options the Communicator presents their UIDs, their supertype, their languages and language communities. This enables the users to disambiguate homonyms and to select the intended concepts from the list of presented options. The first field contains a character string, in this case consisting of a question mark (‘?’), that is unknown to the dictionary. This causes that the Communicator gives it a UID between 1 and 99.

In addition to the first line of the query, the user can specify one or more additional ‘conditions’ that the results should satisfy. This means that a model is specified in which one or more unknowns appear. The query implies that the model acts as a template that should be satisfied by possible solutions. In the example the condition is that the unknown object, denoted by the same character string ‘?’, <is located in> Province Groningen.

The selected options and possible conditions are confirmed by pressing the <Confirm> button, after which the Communicator reflects the question as is shown in the ‘Messages and errors’ area and it starts executing the query.

6.2.3 Logic conclusions and answers

Several rules of logic are applied when Communicator generates responses on queries. First it applies the rule that states that ‘what is applicable for a concept is also applicable for all its subtypes’. That rule is applied as follows.

In the example model in the database there are no objects included that are classified as a road. Furthermore, there are no specifications that one or more roads are located in the Province Groningen. Thus, we would expect that no roads will be found, and if roads would be found, then they would not satisfy the condition. Nevertheless, as is shown in the area ‘Messages and errors’ in Figure 21, the Communicator found three candidate answers: the N51, the N52 and the N5. The fact that those candidates could be found became possible through using the logic that is enabled by the formal language definition. Namely, in the taxonomic dictionary of the formal language it is expressed that a ‘single carriageway’ as well as a ‘dual carriageway’ is defined as a kind of (a subtype of) road and that a ‘single carriageway with two lanes’ is a kind of ‘single carriageway’. In other words, the taxonomic dictionary comprises the following expressions:

single carriageway	is a kind of	road
dual carriageway	is a kind of	road
single carriageway with two lanes	is a kind of	single carriageway

By making use of the knowledge in these definitions and the application of rules from logic it was possible to find three objects that are classified by these subtypes of road. Thus the Communicator found three candidate roads. This illustrates the importance of a good taxonomy (subtype-supertype hierarchy) of concepts in the formal dictionary.

Another rule of logic is applied after that, when the Communicator verified for each of the found roads whether it satisfies the specified condition of being located in Gelderland. That verification uses a rule for ‘transitive

relations'. That rule states: if A has a transitive relation with B and B has a transitive relation of the same kind with C, then it follows that A has also such a transitive relation with C. This logic can be applied in this example, because the formal language defining ontology specifies that a location relation is a subtype of transitive relation as follows:

is located in is a kind of transitive relation

Thus, if A is located in B and B is located in C, then it follows that A is also located in C. This rule from logic could be implemented in the Communicator due to the fact that the relation types are standardized in the formal language and that they are arranged in a hierarchy (a taxonomy) of kinds of relations. From that hierarchy it follows, according to the first rule, that the relation type <is located in> is defined as being a subtype of 'transitive relation'. Applying that rule results in the conclusion that only N51 is located in the Province Groningen, because the model specifies that the N51 is located in Groningen. Whereas it also specifies that Groningen is located in the Province Groningen. That reasoning is included in the answer of the Communicator, as is shown in Figure 21.

6.2.4 Queries for product and process models

The logic that uses the hierarchy of relation types is also applied in other occasions. Especially when in a query some relation type is used, whereas the search should also find relations of kinds that are subtypes of the specified more general relation type.

This is illustrated by the response to another query about the above mentioned roads. That other query is depicted in the upper section of Figure 22. In that query an object with the name N51 is assumed to be known, whereas the query is about all available relations of any kind with that object, without conditions. That query is formulated in Formal English using a very generic relation type as follows:

N51 is related to (a) ?

Query

Left hand term	Relation type phrase	Right hand term	Unit of measure
N51	is related to (a)	?	
112	6068	1	0

Definition of lh object:

☒ Case Sensitive
☒ First Char Match
Conditions:

Select one of each of the following options:

Left UID	Name	Kind	Community	Language
112	N51	road with three separated carriagew	The Dutch state	English
141	N51-b1	width	The Dutch state	English
142	N51-b2	width	The Dutch state	English

Relation UID	Name	Kind	Community	Language
5935	is or can be related to (a)	relation	Formal English	English
6068	is related to (a)	binary relation	Formal English	English

Right UID	Name	Kind	Community	Language
1	?	unknown kind	Model	Dutch

Confirm
Verify model

Messages and warnings:

Query : N51 (112) is related to (a) (6068) ? (1)
Candidate answers:
statement 1 about [N51]: <N51> <is classified as a> <road with three separated carriageways>.
statement 2 about [N51]: <N51> <is a part of> <Road net_1>.
statement 3 about [N51]: <ES1_1> <is a part of> <N51>.
statement 4 about [N51]: <ES1_2> <is a part of> <N51>.
statement 5 about [N51]: <Substructure-N51> <is a part of> <N51>.
statement 6 about [N51]: <N51> <has as part> <Lane N51-1>.
statement 7 about [N51]: <N51> <has as part> <Lane N51-2>.
statement 8 about [N51]: <N51> <has as aspect> <cop N51>.
statement 9 about [N51]: <N51> <has as aspect> <quality N51-2>.
statement 10 about [N51]: <N51> <is owned by> <Province Groningen>.
statement 11 about [N51]: <N51> <is located in> <Groningen>.
statement 12 about [N51]: <R2> <is the start point of> <N51>.
statement 13 about [N51]: <Maintenance N51> <has as subject> <N51>.
statement 14 about [N51]: <Transport on N51> <is facilitated by> <N51>.
Warning: Aspect N51-12 (146) has no value.

Figure 22, A response to a query that uses a generic relation type

The definitions of relation types in the formal language are all arranged in a subtype-supertype hierarchy, and for each relation type it is specified what the kinds of the left hand and right hand related things by definition are. In this example the specified relation type, called <is related to (a)> has as UID 6068 and is defined in the language definition as a binary relation between an individual thing, in this case called N51, and either another individual thing or a kind of thing, which is specified as an unknown ('?'). Furthermore, all relation types for relations between individual things as well as for relations between an individual thing and a kind are defined as subtypes (or subtypes of subtypes, etc.) of this relation type. Because all relation types in the language definition are arranged in a taxonomic hierarchy, the Communicator can read the hierarchy and find all relations of any subtype of the specified relation type. This implies that this single line expression is sufficient for the

Communicator to retrieve a complete product model with all relations of any kind with an object, such as N51 in this example. This illustrates the importance of the hierarchy of relation types.

The Communicator uses the specified character string 'N51' to search for things that have this string in their name. If a character string does not occur in any known term, or if it is explicitly denoted as being an unknown (by being called e.g. 'what', 'which', 'who' or '?'), then the 'object' (anything) is allocated a UID between 1 and 99 and it is registered as 'unknown' by the Communicator. In this example the Communicator discovers that the string 'N51' appears more than once in names of things. The formal definition of relation types that defines the kinds of the related things enables the Communicator to read the definition of the used relation type (6068), and to conclude that it specifies that the left hand object shall be an individual thing. Accordingly the Communicator verifies during its search for options whether candidate objects are indeed individual things. This verification is possible because a rule of the formal language states that things are individual things when they are classified by an explicit classification relation with a kind of thing⁶. Similarly, something can be recognized as being a kind of thing, because by using the taxonomic hierarchy of kinds it can be verified whether something is a subtype of the kind that is called 'individual thing'. This again demonstrates the importance of a taxonomy of concepts, including a hierarchy of kinds of things and kinds of relations. After applying these rules, the Communicator presents the found individual things as options, together with among others their classification etc. as shown. This presentation of options and the verification of a selection by the users are in particular intended to distinguish homonyms.

The most probable option is preselected. For the preselected or selected option the Communicator presents its UID as well as a textual definition. This makes unambiguously clear which option is selected. From the list of options the intended option can be selected. Road N51 appears to be one of those options and is therefore selected as the object in focus.

When a user of the Communicator concludes that the system has interpreted the query correctly, and the correct things are selected, then he can confirm that by pressing the 'Confirm' button. This starts the search for expressions that comply with the query. In this example, the search results initially in

⁶ When an explicit classification relation is missing, and the object does not appear in a subtype-supertype relation either, then it might be that the object appears in relations of other kinds, from which it may be concluded that the object is an individual thing. In such cases it shall be verified whether such a conclusion is in conflict with its use in other expressions that use other kinds of relation.

finding 14 expressions, as is specified in the section ‘Messages and errors’ in Figure 22. Those 14 expressions represent only relations in which the ‘object in focus’, the N51, directly plays a role.

The following paragraph shows that product models also include expressions of relations that are indirectly related with the object in focus. This is illustrated by the figures with various views and details of the found product model.

6.2.5 Views on product models

The scope of a product model of a physical object, such as the N51, typically includes at least the aspects (properties) of the object as well as its composing parts. When the scope of a model is not limited to an explicitly specified subset of all possible relations about the object in focus, then the scope of a product model is usually determined by rules. For example, the rule that a product model includes not only the direct parts of the object in focus, but also the parts of the parts, up to any available detail. Furthermore, the rule that it does include the aspects, their classification as well as the values of those aspects, for the object in focus, as well as the aspects of its parts and parts of parts, connections with other parts, etc. Finally it may include the occurrences in which the object in focus is involved as well as occurrences in which any of its parts are involved.

The above rules are applied by the Communicator for searching for additional expressions about parts, aspects of parts and occurrences in which those things are involved. The results for the N51 are presented in the various views below. Figure 23 presents an example of a view of the product model of the N51.

Model views									
Expressions Composition Product List of kind Activities Documents									
Tree structure									
Line#	Object			Kind	Aspect	Kind of aspect	Value	UoM	Status
1	N51			road with three sep					
3					Aspect	Aspect type	Value	UoM	Status
4					cap N51	capacity	350	1/h	proposed
5					quality N51-2	quality	moderate		proposed
6	Part hierarchy	Part of part	Further part	Kind					
7	E51_1			road begin					proposed
8	E51_2			road end					proposed
9	Substructure-N51			substructure					proposed
10	Lane N51-1			lane	N51-b1	width	3,2	m	proposed
12		N51-1top		top layer	N51-1topD	thickness	20	mm	proposed
11					N51-I1	length	4,6	km	proposed
13	Lane N51-2			lane	N51-b2	width	3,2	m	proposed
14					N51-I2	unknown kind	unknown value		proposed
15	Occurrences	Role	Involve	Kind					
16	Transport on N51			transport					proposed
17		concerned	S_N51	traffic stream	L_N51	traffic intensity	10000	1/h	proposed
18	Maintenance N51			maintenance	T-1	duration	3	dag	proposed
19		input	Start signa	start signal					proposed
20		output	Stop signa	signal					proposed
21		Part occur		Kind of part					
22		Prep-N51		preparation					voorstel
23		Rep-N51		repair					voorstel
24		Inspection-†		inspection					voorstel
25		Usage-N51		usage					voorstel

Figure 23, A product model view of the N51

Conform the definition of the scope of a product model, the Communicator doesn't search only for relations with the object in focus (N51) itself, but it searches also for relations of the appropriate kinds with related objects. As a result it is apparent from Figure 23 that the product model includes aspects of the N51, its parts and parts of its parts, with their aspects and also the occurrences (activities) in which the N51 is involved, parts of those occurrences, and inputs and outputs of those occurrences. The model includes for example properties of lanes and the duration of a maintenance activity in which the N51 is involved as well as the start signal, the stop signal and its partial activities.

The 'view' of Figure 23 has a generic structure that can be used to present a product model for any individual physical object. The generic structure allows that the presented nature and number of details is flexible and depends on the information that is found about the 'object in focus'.

Figure 24 presents an example of another presentation of the same product model. The form is also generated automatically by the Communicator.

Model views

Expressions Composition **Product** List of kind Activities Documents

1	Product form for:			road with three separated carriageways			N51		
2				Description:					
3				Aspect	Aspect type	Value	UoM	Status	
4				cap N51	capacity	350	1/h	proposed	
5				quality N51-2	quality	moderate		proposed	
6	Part hierarchy	Part of part	Further part	Kind					
7	E51_1			road begin				proposed	
8	E51_2			road end				proposed	
9	Substructure-N51			substructure				proposed	
10	Lane N51-1			lane	N51-b1	width	3,2	m	
11					N51-l1	length	4,6	km	
12		N51-1top		top layer	N51-1topD	thickness	20	mm	
13	Lane N51-2			lane	N51-b2	width	3,2	m	
14					N51-l2	unknown kind	unknown valu	proposed	
15	Occurrences	Role	Involvements	Kind					
16	Transport on N51			transport				proposed	
17		concerned	S_N51	traffic stream	I_N51	traffic intensity	10000	1/h	
18	Maintenance N51			maintenance	T-1	duration	3	dag	
19		input	Start signal Usage-N51	start signal				proposed	
20		output	Stop signal-N51	signal				proposed	
21		Part occurrence		Kind of part					
22		Prep-N51		preparation				voorstel	
23		Rep-N51		repair				voorstel	
24		Inspection-N51		inspection				voorstel	
25		Usage-N51		usage				voorstel	

Figure 24, Data sheet (product form) for physical objects, generated for road N51

The form or data sheet in Figure 24 is also dynamically composed.

Note that the Communicator presented value fields in yellow for aspects which value may be changed in the form. This indicates that the form may also be used as an input form.

Furthermore, the Communicator performed some verification on completeness. This is apparent from the two values that are marked in red. This indicates that property N51-I2 is not classified and in addition to that is has no value, although classifications and values are compulsory when an object is specified to have a particular property.

The last two figures present views in which occurrences and activities are in focus. The model of the N51 comprises not only information about physical objects and their properties, but also information about occurrences, such as activities and processes in which the N51 is involved. Thus the model of the N51 includes not only a product model, but also an activity or process model. The Communicator has dedicated dynamic views for the presentation of occurrences. An example of a combination of a graphical and a textual view is presented in Figure 25 in which the composition and sequence of the activity ‘Maintain-N51’ is presented.

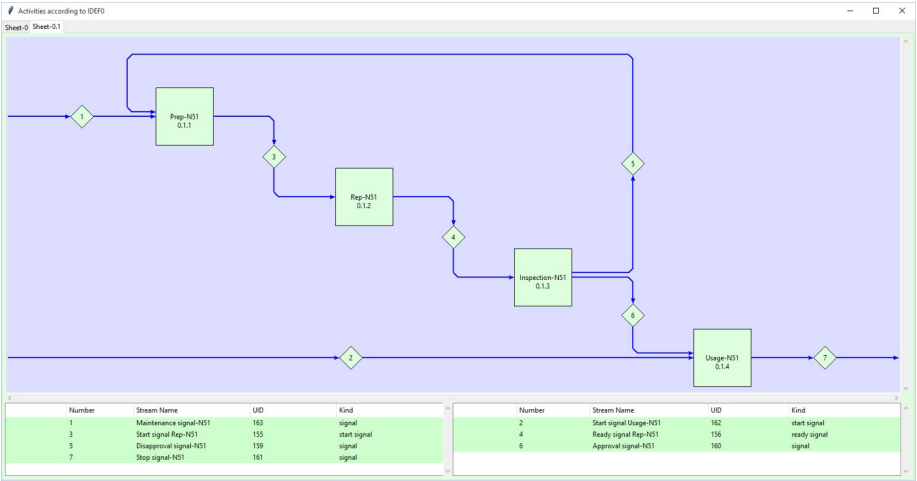


Figure 25, Automatically generated activity diagram (IDEF0 diagram)

The method of presenting as in Figure 25 is derived from the IDEF0 standard for ‘functional modeling’ (see <http://en.wikipedia.org/wiki/IDEF0>). The first figure (which is not depicted) presents the whole activity ‘Maintain N51’. Figure 25 displays the second figure, which shows the component activities and presents them in their sequence in time. The lines between the activities represent material and information flows between those activities. The numbers in those lines refer to the table under the figure, which presents additional details about those flows.

Another view on the model from an activity or process perspective is presented in Figure 26.

Model views

Expressions

Composition

Product

List of kind

Activities

Documents

Occurrence	Part occurrence	Involved	UID	Kind	Role	UID of role
Transport on N51			138	transport		
		S_N51	127	traffic stream	concerned	640028
		N51	112	road with three sepi	facilitator	640107
Maintenance N51			150	maintenance		
		N51	112	road with three sepi	concerned	640028
		Start signal Usage-N51	162	start signal	input	640016
		Stop signal-N51	161	signal	output	640019
Prep-N51			152	preparation		
		Start signal Rep-N51	155	start signal	output	640019
		Disapproval signal-N51	159	signal	input	640016
		Maintenance signal-N5	163	signal	input	640016
Rep-N51			153	repair		
		Start signal Rep-N51	155	start signal	input	640016
		Ready signal Rep-N51	156	ready signal	output	640019
Inspection-N51			158	inspection		
		Ready signal Rep-N51	156	ready signal	input	640016
		Disapproval signal-N51	159	signal	output	640019
		Approval signal-N51	160	signal	output	640019
Usage-N51			157	usage		
		Approval signal-N51	160	signal	input	640016
		Stop signal-N51	161	signal	output	640019
		Start signal Usage-N51	162	start signal	input	640016

Figure 26, Activity hierarchy with involved objects and material or information streams

The view Figure 26 presents a ‘topology’ of the maintenance activity and its component activities as well as their classification and the information flows and flows of matter that are involved with input and output roles in those activities.

A complete information model comprises a collection of expressions that can be presented and stored for example in the Gellish Expression Format. An example of a table in that format for the information model of the N51, excluding the knowledge and definition models, is presented in Figure 27.

Language	Community	Product	List of kind	Activities	Documents	rhUID	rhName	Description	uomUID	UoM	Remarks	Status	Context
English	The Dutch state	113	N52	536	1225	is classified as a	45382	dual carriageway	0			proposed	
English	The Dutch state	120	E52_1	512	1260	is a part of	113	N52	0			proposed	
English	The Dutch state	120	E52_1	563	1588	has a role as a	45449	road begin	0			proposed	
English	The Dutch state	121	E52_2	513	1260	is a part of	113	N52	0			proposed	
English	The Dutch state	121	E52_2	564	1588	has a role as a	45415	road end	0			proposed	
English	The Dutch state	112	N51	537	1225	is classified as a	45412	road with three seg	0			proposed	
English	The Dutch state	112	N51	592	1727	has as aspect	143	cap N51	0			proposed	
English	The Dutch state	143	cap N51	594	1225	is classified as a	790728	capacity	0			proposed	
English	The Dutch state	143	cap N51	596	5025	has on scale a value equal to	2000000350	350	570387	1/h		proposed	
English	The Dutch state	112	N51	593	1727	has as aspect	144	quality N51-2	0			proposed	
English	The Dutch state	144	quality N51-2	595	1225	is classified as a	551008	quality	0			proposed	
English	The Dutch state	144	quality N51-2	597	5020	is qualified as	587380	moderate	0			proposed	
English	The Dutch state	118	E51_1	510	1260	is a part of	112	N51	0			proposed	
English	The Dutch state	118	E51_1	561	1588	has a role as a	45449	road begin	0			proposed	
English	The Dutch state	119	E51_2	511	1260	is a part of	112	N51	0			proposed	
English	The Dutch state	119	E51_2	562	1588	has a role as a	45415	road end	0			proposed	
English	The Dutch state	136	Substructure-N51	632	1260	is a part of	112	N51	0			proposed	
English	The Dutch state	136	Substructure-N51	651	1225	is classified as a	45417	substructure	0			proposed	
English	The Dutch state	112	N51	581	1260	has as part	139	Lane N51-1	0			proposed	
English	The Dutch state	139	Lane N51-1	583	1225	is classified as a	43427	lane	0			proposed	
English	The Dutch state	139	Lane N51-1	585	4798	has as property	141	N51-b1	0			proposed	
English	The Dutch state	141	N51-b1	587	1225	is classified as a	550464	width	0			proposed	
English	The Dutch state	141	N51-b1	589	5025	has on scale a value equal to	926318	3.2	570036	m		proposed	
English	The Dutch state	139	Lane N51-1	598	4798	has as property	145	N51-11	0			proposed	
English	The Dutch state	145	N51-11	600	1225	is classified as a	551353	length	0			proposed	
English	The Dutch state	145	N51-11	601	5025	has on scale a value equal to	920577	4.6	570415	km		proposed	
English	The Dutch state	139	Lane N51-1	602	1260	has as part	147	N51-1top	0			proposed	

Figure 27, Information model export file of the N51 in the Gellish Expression Format

Figure 27 presents only a part of the (scrollable) table, not only vertically, but also horizontally, because the table includes also contextual facts (meta data). Those contextual facts include information about who created the expression, when that occurred, from which date-time up to which date-time the statement is valid, etc. By selecting a line and pressing the <Context> button, those contextual facts are presented after which they can be modified.

6.2.6 Exchange of information

A collection of expressions that is the result of a query can be stored in a file and exchanged with other parties. This is done as described below.

By selecting the button <Save on file> at the bottom right corner in the screen of Figure 27 (and after selection a file format), a file is created and the screen that is shown in Figure 28 is presented.

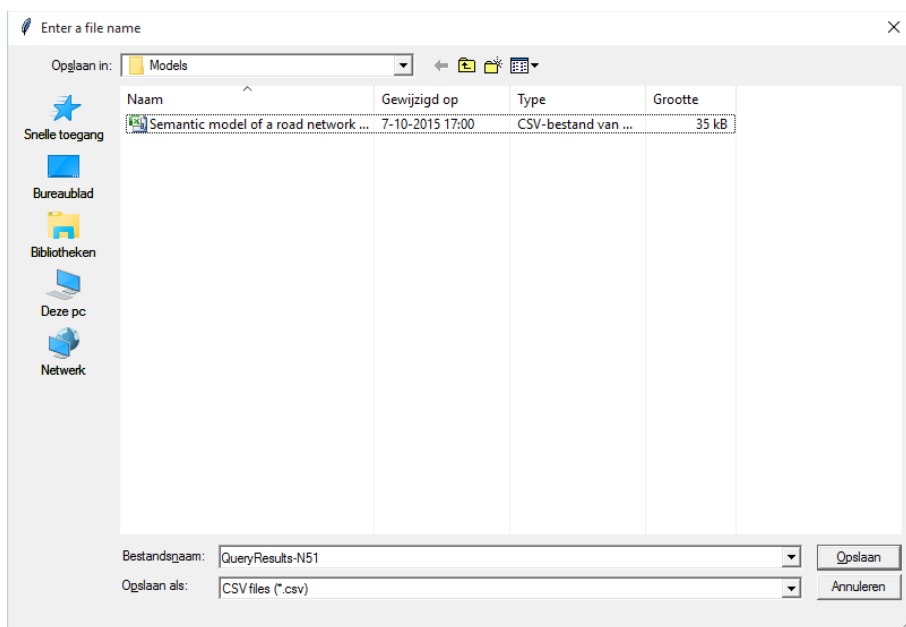


Figure 28, Storage of search results

The screen of Figure 28 enables specifying a name and location where the file can be stored. After selecting the <Store> ('Opslaan') button, the files can be sent to or be made available for somebody else, for example as an appendix of an email or as attachment to a specification.

The receiving party can then import the file in another application that is suitable for reading files with expression in the formal language and in the appropriate format. The Gellish Communicator is also suitable as a receiver. In the receiving application, the information can possibly be combined with other information. For a correct interpretation, that other application should use the same language definition as the sending party, possibly extended with a local dictionary and terminology of the receiving party. During import, the application will typically execute similar verifications as the sending party has performed. That will usually imply that the content of the file is verified on correctness of usage of the formal language, on consistency with the new context, as well as on compliance with the knowledge and requirements that are available in the new environment. This process is the same as is described above for import, etc. in par. 6.2.1. If required, the receiving party can return a modified or extended file or he may return another file.

The Communicator might also create a file with a query with a search specification, with or without conditions that is expressed in the formal

language. Such a file can also be sent to another party. Then the receiving party can import the query in his system and execute it on his own database. The resulting answers can then be returned to the requesting party.

6.2.7 Editing of information models

The Communicator supports editing of information models per file. Extensions and modifications can therefore be performed on files that are imported through selecting the option <Edit> in the main menu.

For example, assume that we want to add an individual thing, called N58, which is classified as a road to a model in a file with information about multiple roads. Then we need to add the following expression:

N58 is classified as a road

And assume that we want to add as partial textual definition:

“that is located in Groningen.”

Then we first import the file in <Edit> mode after which the ‘Editor’ screen appears. Figure 29 gives an example of the Editor screen in which the above information is already entered.

Editor

Enter a new fact. Then: <Verify>, <Context> and <Add>.
 <Save as> creates a copy file.
 <Proposals> generates proposals for expressions, based on the classification of an individual thing

File name: C:/D-disk/Gellish/Gellish Python/New Model CSV.csv [Verify]

Language of lh term: English [Context]

Community: Formal English Fact UID: 692 [Add]

Left hand term	Relation type phrase	Right hand term	Unit of measure	
N58	is classified as a	road		Proposals
Lh UID	Rel-type UID	Rh UID	UoM UID	Save as
255	1225	40153	0	
Lh role type:		Rh role type:		close
classified individ		classifier for an individ		
Partial definition: that is located in Groningen.				

Proposals for expressions: [Accept]

				☐
				☐
				☐
				☐
				☐
				☐
				☐
				☐
				☐
				☐
				☐

Errors and messages:

Verified expression: [255, 'N58', 1225, 'is classified as a', 40153, 'road', 'Is een road that is located in Groningen.\n']

Figure 29, Entering an expression that defines N58

Before the entered expression may be added to the information model, it should first be verified by the Communicator whether the entered data forms a correct expression and whether it is compliant with the information that is already available in the model and the dictionary. Therefore, before selecting <Add>, the user should select <Verify>. Figure 29 presents the results of the verification step, which shows that the verification process recognized that N58 is a new thing. Therefore, the Communicator allocates to it a new UID (255 in the example), being the next number that is available in the range that is mentioned in the header of the file. Furthermore it recognizes the phrase that denotes the relation type with UID 1225 in the dictionary, and it recognizes the name ‘road’ with UID 40153 in the dictionary. Furthermore, the kinds of roles as presented.

If verification does not result in error messages or in unwanted results (such as incorrect UIDs), and if also the <Context> with contextual facts satisfies the user, then the <Add> button can be used, after which the Communicator adds the expression to the file that was imported in <Edit> mode.

The use of the lower ‘proposals’ part of the Editor window is discussed in the next paragraph.

6.2.8 Knowledge and requirements based proposals and verification

If knowledge and requirements about kinds are expressed in the formal language and if they are imported in the Communicator, then they are available for the generation of proposals for new expressions and for the execution of verifications.

Thus, if an individual thing, such as N58, is added and is classified by a kind, such as road, and the user selects the button ‘Proposals’, then the Communicator uses the knowledge and requirements about that kind and generates proposals for new expressions. Figure 30 presents the results of the above steps with an example of proposals for possible expressions about the N58.

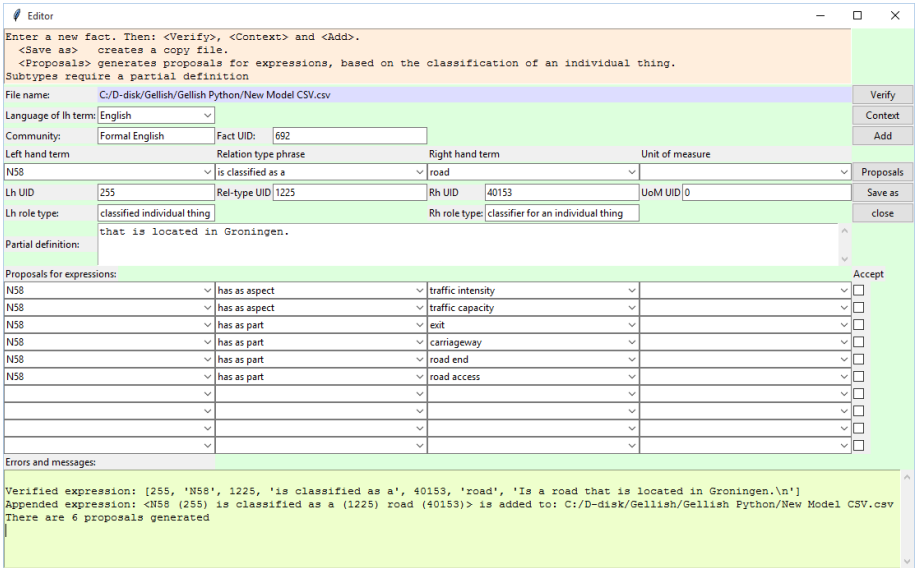


Figure 30, Added expression with generated proposals

The Communicator was able to derive the proposals that are presented in Figure 30 because knowledge and requirements about the concept road and about its supertypes were expressed in the formal language and were imported. For example, the imported knowledge model about a road contains the following expressions:

- road can have as part a exit
- road can have as aspect a traffic capacity
- etc.

Because the N58 is classified as a road, the Communicator can conclude that these possibilities are options that are applicable for the N58. Furthermore, the formal language definition includes the specification that a kind of relation that expresses a possibility, such as is expressed by the phrase <can have as part a>, can be realized by a relation of the kind <has as part>. Because of that knowledge the proposal can be N58 <has as part> (an) exit.

When a user of the Communicator accepts one or more of the proposals he should specify individual objects, which he should classify by the specified kinds or their subtypes and possibly he should also qualify or quantify them. If that is not done, then the Communicator produces an error message and generates a proposal for a correct way of modeling by proposing expressions.

For example, assume that a user accepts the proposal and enters the expression that N58 <has as aspect> lane capacity, exactly as proposed as is illustrated in Figure 31.

Editor

Enter a new fact. Then: <Verify>, <Context> and <Add>.
<Save as> creates a copy file.
<Proposals> generates proposals for expressions, based on the classification of an individual thing.
Subtypes require a partial definition

File name: C:/D-disk/Gellish/Gellish Python/New Model CSV.csv

Verify

Language of lh term: English
Community: Model
Fact UID: 694

Left hand term

Relation type phrase

Right hand term

Unit of measure

N58

has as aspect

traffic capacity

Proposals

Lh UID

256

Rel-type UID

1727

Rh UID

110600245

UoM UID

0

Save as

Lh role type:

possessor of an

Rh role type:

intrinsic aspect

close

Partial definition:

that is located in Groningen.

Proposals for expressions:

N58	has as aspect	N58-traffic capacity-1		☐
N58-traffic capacity-1	is classified as a	traffic capacity		☐
N58-traffic capacity-1	has on scale as value	number	UoM	☐
				☐
				☐
				☐
				☐
				☐
				☐
				☐

Accept

Errors and messages:

Verified expression: [256, 'N58', 1225, 'is classified as a', 40153, 'road', 'Is a(n) road that is located in Groningen.\n']
Appended expression: <N58 (256) is classified as a (1225) road (40153)> is added to: C:/D-disk/Gellish/Gellish Python/New Model CSV.csv
There are 5 proposals generated
Verified expression: [256, 'N58', 1727, 'has as aspect', 110600245, 'traffic capacity', '']
** Right hand object <traffic capacity> is a kind; it should be a classified individual. See the proposal.

Figure 31, Incorrect input with a proposal for improvement

Then the Communicator presents as the last sentence in the ‘Errors and messages’ area an error message, because the object ‘traffic capacity’ is

recognized as being a kind, whereas according to the definition of the relation type <has as aspect>, the left hand as well as the right hand object shall be an individual thing. Thus the right hand term should denote an ‘individual traffic capacity’.

The formal language definition prescribes a pattern for specifying the addition of an individual aspect. Based on that pattern, the Communicator can produce a proposal for a correct expression of an addition of an individual aspect. In the example the proposal consist of three lines in the ‘Proposals’ area. By concatenating the name of the possessing individual object, the kind of aspect and a sequence number, the Communicator has also generated a proposal for a name of the individual aspect: ‘N58-traffic capacity-1’.

The user is free to modify the proposals by changing the names and by entering the appropriate value and unit of measure. After that the proposals can be accepted by marking the fields under ‘Accept’. This is illustrated in Figure 32.

Editor

Enter a new fact. Then: <Verify>, <Context> and <Add>.
 <Save as> creates a copy file.
 <Proposals> generates proposals for expressions, based on the classification of an individual thing.
 Subtypes require a partial definition

File name: C:/D-disk/Gellish/Gellish Python/New Model CSV.csv

Language of lh term: English

Community: Model Fact UID: 694

Left hand term: N58 Relation type phrase: has as aspect Right hand term: traffic capacity Unit of measure:

Lh UID: 256 Rel-type UID: 1727 Rh UID: 110600245 UoM UID: 0

Lh role type: possessor of an. Rh role type: intrinsic aspect

Partial definition: that is located in Groningen.

Proposals for expressions:

Left hand term	Relation type phrase	Right hand term	Unit of measure	Accept
N58	has as aspect	N58-traffic capacity-1		☒
N58-traffic capacity-1	is classified as a	traffic capacity		☒
N58-traffic capacity-1	has on scale as value	300	1/h	☒
				☐
				☐
				☐
				☐
				☐
				☐
				☐

Errors and messages:

Verified expression: [256, 'N58', 1225, 'is classified as a', 40153, 'road', 'Is a(n) road that is located in Groningen.\n']
 Appended expression: <N58 (256) is classified as a (1225) road (40153)> is added to: C:/D-disk/Gellish/Gellish Python/New Model CSV.csv
 There are 5 proposals generated
 Verified expression: [256, 'N58', 1727, 'has as aspect', 110600245, 'traffic capacity', '']
 ** Right hand object <traffic capacity> is a kind: it should be a classified individual. See the proposal.

Figure 32, Addition of an individual traffic capacity

In the example in Figure 32 the user left the proposed name ‘N58-traffic capacity-1’ unchanged and he added the value 300 and the unit ‘1/h’. The system then generated a UID for the lane capacity, whereas a next UID is chosen from a series that is declared applicable for this model (as specified in the first header line). By pressing the ‘Save as’ button, the Communicator saved the formal language expressions as an extended file with a new name at a specified location.

This illustrates how users can be supported in modeling and creating new expressions by using knowledge and requirements that were also expressed in the formal language and that were imported by the Communicator.

6.3 The Gellish Search Engine

The Gellish Search Engine enables searching for anything via a powerful text string search or via its UID. Such a search can use any part of combination of parts of the name or one of the aliases of an individual thing or of a kind, using the generic search window that is shown below.

For example: a search for documents about a particular compressor.

There are three ways to find a document:

- a) Search for the *name* of the individual compressor and navigate through the product model to find documents about the object.
- b) Search for a text string(s) in the *document title* or *document number* (drawing number) which can also be a binder name, section or sub-section.
- c) Search for a *kind* of object or *kind* of document and determine the required tag name or document that is classified by the kind.

For all three ways of searching the same single search field in the generic search window can be used. There is no need to specify the way of searching.

We choose option c) and start with searching for the kind ‘compressor’ by entering a part of the name as search string in the generic search window of Figure 33.

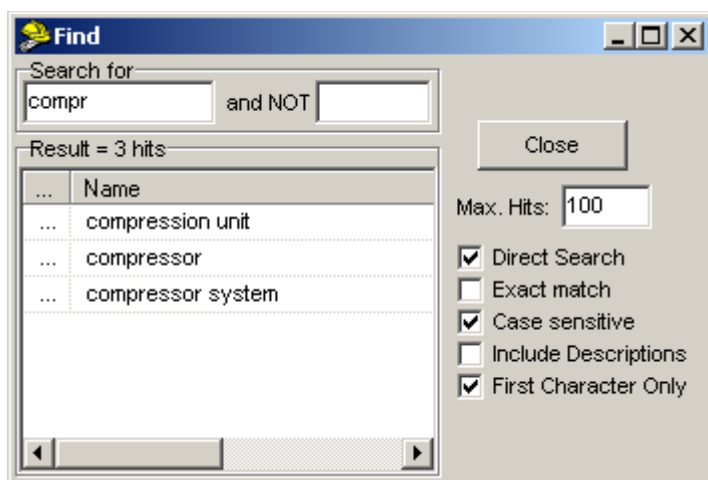


Figure 33, Generic search window

After double click on the preferred option ‘compressor’ the search engine delivers a list of things that are classified as compressor or as one of its subtypes as presented in Figure 34. Such a search will use the hierarchy of kinds in the taxonomic dictionary.

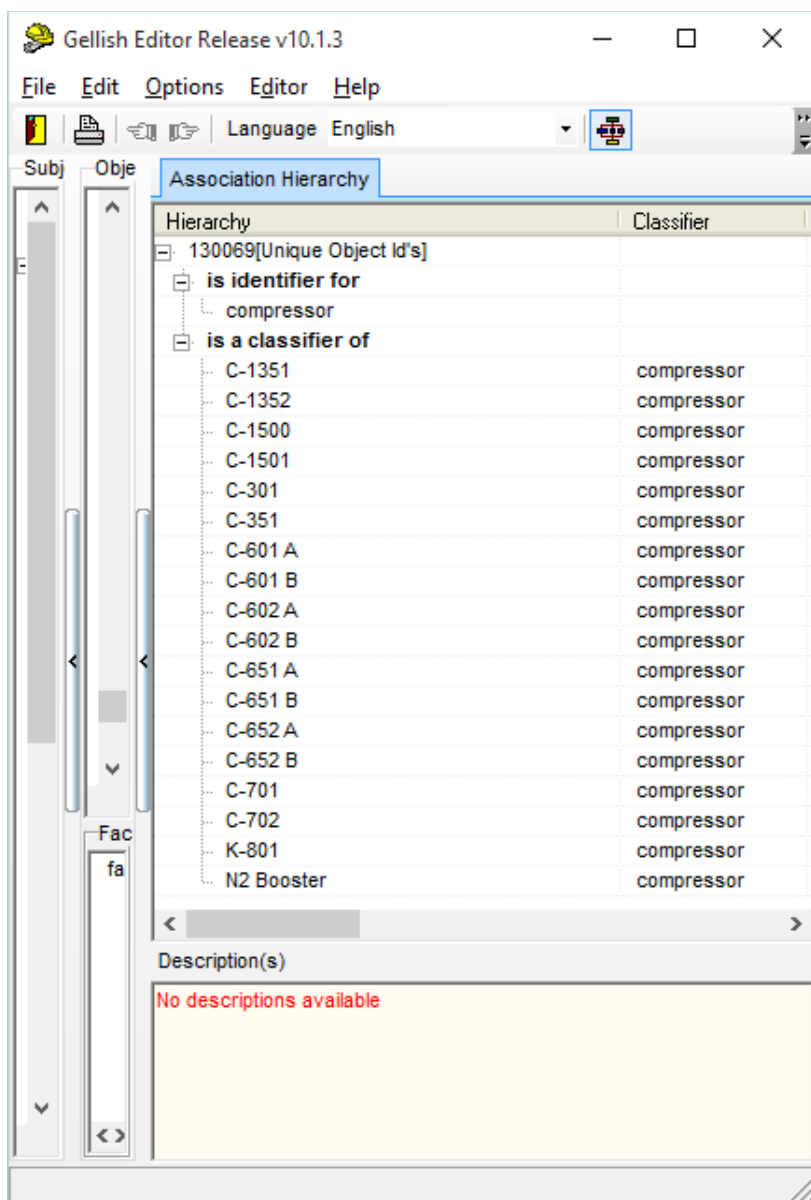


Figure 34, Resulting list of compressors

Then we can navigate through the network of objects that are related to the concept 'compressor' that is in focus, by double clicking on any object of interest. For example, a double click on C-351 will put compressor C-351 in focus, while presenting information about it as presented in Figure 35:

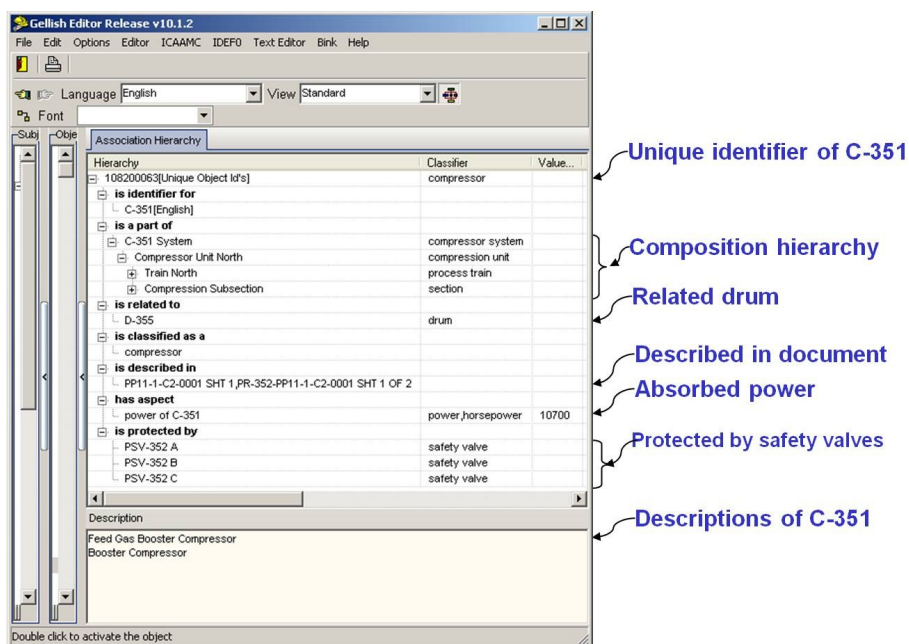


Figure 35, Information about compressor C-351

The above information includes the fact that C-351 is described in document PP11-...

Such a document can be selected by double clicking on its name PP11-..., which will result in information about the drawing as presented in Figure 36.

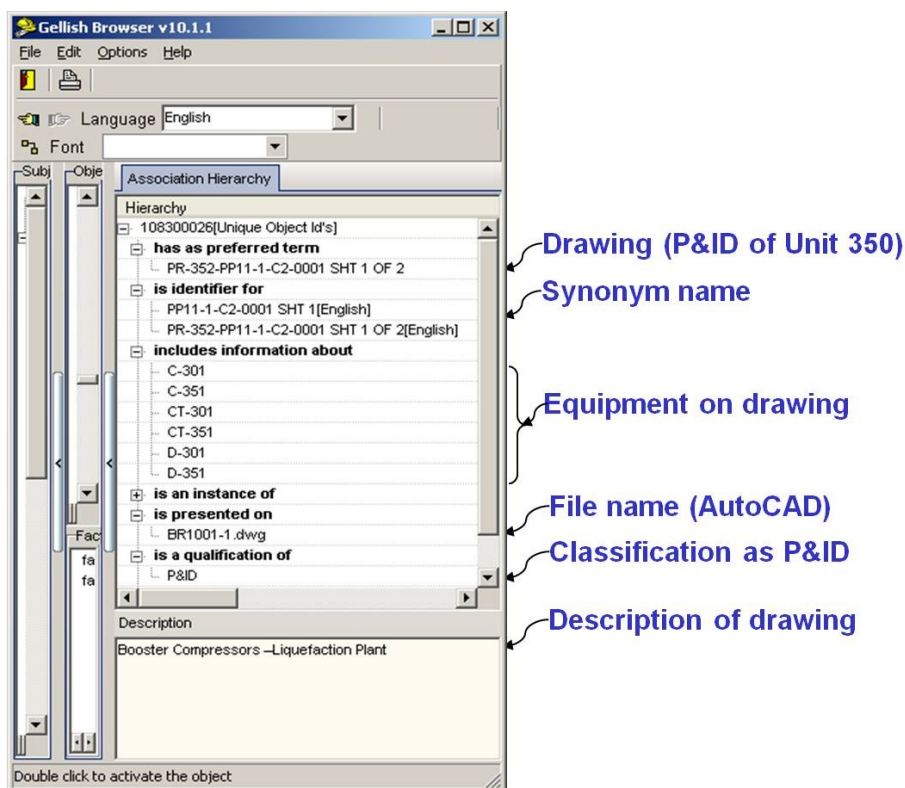


Figure 36, Information about drawing PP11-352...

This display illustrates that this document contains information about six pieces of main equipment. The document could have been found via any of them.

The document appears to be related to one or more files by an <is presented on> relation. In the example the document PP11-... is presented on file BR1001-1.dwg.

That file can be selected by double clicking on the file name.

As a result of that the file becomes the object in focus, whereas in the window "Objects" the file name is highlighted as is shown in Figure 37.

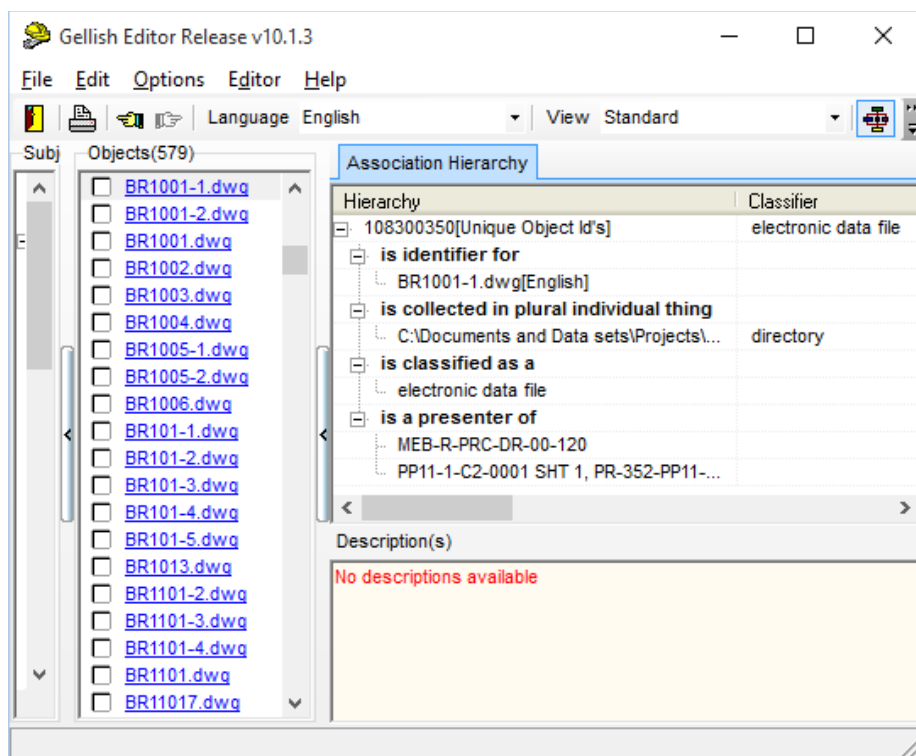


Figure 37, Information about the file BR1001-1.dwg

Double clicking on the highlighted file name will launch a Viewer/Browser to view the AutoCAD drawing as is illustrated in Figure 38.

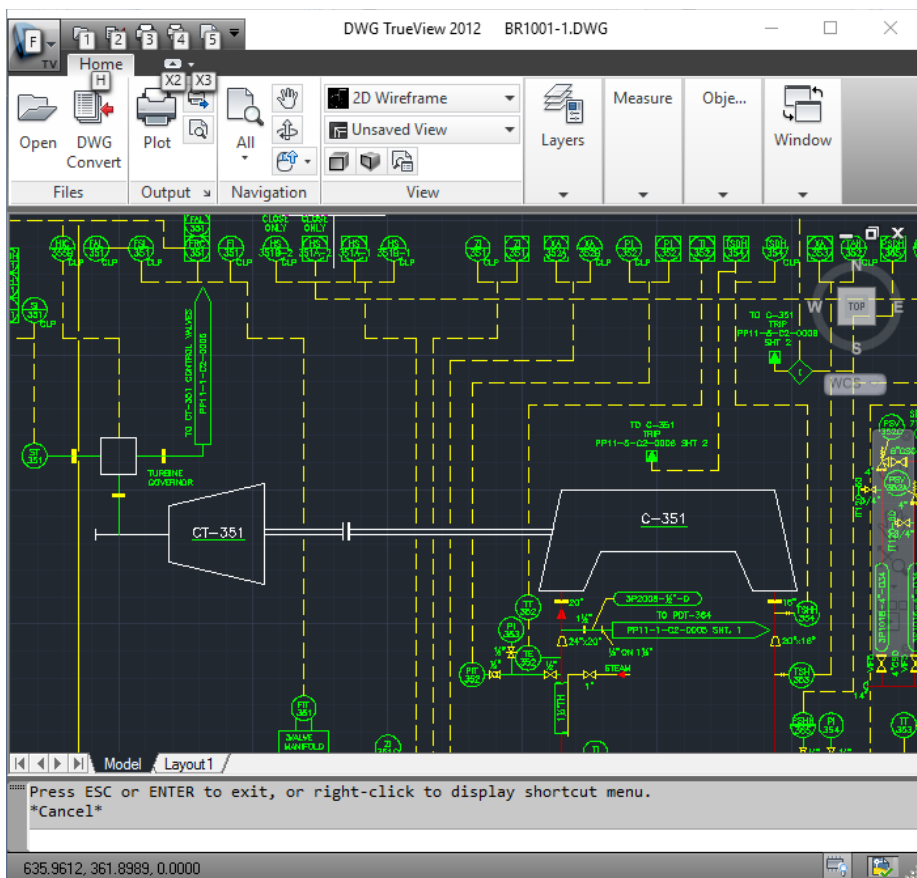


Figure 38, Display of AutoCAD drawing

Note that the Search Engine used only formal expressions to navigate through the model.

7 Process information modeling

An occurrence in its widest sense is anything that happens. This includes human activities, but also physical and (bio)chemical processes, control processes, events and physical phenomena. Processes include both discrete processes and continuous processes. A discrete process consists of a sequence of discrete process steps or actions. Examples are business processes as a sequence of activities, mechanical processes in the mechanical industry and batch processes in the chemical industry. A continuous process is a process that is an operation on a continuous stream of raw material to produce one or more continuous streams of product materials. For example, the flowing of water in a river is a continuous process. An occurrence in fact always happens over time, but especially for events the duration often can be neglected.

Occurrences can be classified by concepts that can be denoted by nouns as well as by verbs, such as the classification of a human activity as a walk or as walking. Both are included in the Gellish dictionary. Most kinds of occurrences have an active form, a passive form and a noun form and sometimes other forms. For example: act, acting and action, but also activity. All these forms are treated as synonyms.

In general, generic kinds of activities do not include the subject of the activity, nor the performer or the enabler. For example, just ‘design’ or ‘operate’. However, more specialized ‘qualified’ activities often do include the subject of the activity or process. For example, ‘design a pump’ or ‘operate a facility’. In such a case the definition of such a kind of occurrence should include an explicit relation that states that it has by definition as subject a particular kind of thing.

A similar story holds for specific activities that include a specific subject. For example, ‘design P-101’.

An occurrence, activity, process or event describes the behavior of the objects involved in that occurrence as well as their mutual interaction. This means that an occurrence is an interaction between two or more objects. Therefore, an occurrence can be expressed as a higher order relation, being a relation with usually more than two related objects. The details of an occurrence are expressed by the specification of the objects that are involved in the interaction, especially in the various roles that they play.

Activities, processes and events with sequences of occurrences and their involved physical objects together can be modeled in semantic information

models as higher order relations that themselves are related with the involved things, each playing its own roles. The kinds of involvement relations between an occurrence and its involved things are specialized according to the kinds of roles that are played by the involved things.

This section includes guidelines on how to model occurrences as well as states and state changes of objects. It also briefly describes how graphical diagrams of process models or activity models, such as PFS's (process flow schemes), IDEF0 diagrams⁷ and WFS's (work flow schemes), could be generated from the models. It includes also the reverse process: when an activity model is created or modified graphically, this document indicates how a corresponding activity model should be produced in a formal language from such a graphical model.

The term activity in this document stands for any occurrence, including also processes and events, which may be business processes as well as biological, physical or chemical processes and events.

7.1 Modeling of activities, processes and events

An activity, process or event (an occurrence) is an interaction between various physical objects that are involved in the occurrence. We distinguish between *individual occurrences* and *kinds of occurrences*. Any occurrence or kind of occurrence is represented by a unique identifier (UID), and is denoted by a name and optionally by one or more synonyms or aliases.

Each individual occurrence shall be defined by classifying it by at least one classification relation (<is classified as (a)>) and shall have a textual definition. For example, a project to construct a building is a complex process that can be classified as building or as a building process. An example of the expression of such a fact is the following expression:

Project X is classified as building

In this expression the concept 'building' is a kind of process that classifies an individual process to build a particular building⁸.

Each kind of occurrence shall be defined by at least one <is a specialization of> relation that specifies what its supertype kind of occurrence is. This specialization relation shall be accompanied by a textual description of the

⁷ <http://en.wikipedia.org/wiki/IDEF0>

⁸ The term building has two meanings (homonyms): the building activity and the resulting physical object. Homonyms are different concepts in Gellish, distinguished by different unique identifiers (UIDs).

concept that further defines what is meant by such a kind of occurrence. These ‘definition models’ should be included in a domain dictionary, that also records what its unique identifier in the formal language is. An example of such a specialization relation and textual definition is the following formal language expression:

building is a specialization of process to produce a physical building from raw materials.

A definition model of an activity may also include the fact that the kind of activity has by definition a particular kind of physical object as its subject. For example:

building has by definition as subject a building

Note that the formal language version of the latter expression uses two different UIDs to denote and distinguish the building activity and the physical building respectively. Furthermore, the language defining ontology defines the used kind of relation such that the left hand object shall be an occurrence and the right hand object shall be a physical object. This definition enables software to verify whether that is indeed the case by using the hierarchy of concepts in the taxonomic dictionary.

Names of kinds of occurrences can have various syntactic forms. For example, a particular activity may be classified for example as an ‘inspection’ or as ‘inspecting’ or as ‘to inspect’. All those inflections denote the same concept. Semantically it does not matter whether a noun or a verb is used to classify the activity⁹. Therefore, these terms are indicated as synonyms in the formal dictionary and they share the same UID.

The creation of domain taxonomic dictionaries with these kinds of definition models is further discussed in the chapter 10.

7.2 Sequences of activities

An activity topology or sequence of activities is typically described in one of two ways:

1. By specifying that a physical object that is output from one activity is

⁹ Semantically there is a difference in time, because inspecting typically indicates an ongoing process. However, in a formal language the tense is irrelevant, because the time that a fact or occurrence takes or took place or will take place is recorded separately through the time of begin and end of existence or validity of the fact or occurrence.

input in a next activity.

2. By specifying that one activity is performed after another, and optionally specifying separately which physical objects are involved in each of the activities.

The first method is used for example as the basis for the IDEF0 activity modeling method. The second method is used for example as the basis for the Work Flow Schemes (WFS), which are used to schedule and present activity sequences for example for commissioning and start up processes of new facilities or for turnaround processes (shut down, maintenance, commissioning and start up) of existing facilities.

These sequences of activities can be specified for sequences of individual activities or as sequences of kinds of activities in general. Sequences of individual activities are for examples descriptions of measurements and observations that actually took place (see also chapter 8); it can also be an example of a typical individual process, such as a procedure that is described as an (imaginary) individual process that should be repeated in reality by copying. However, in most cases sequences of activities are general sequences between kinds of activities. For example, maintenance or usage instructions about all individual objects of a particular type can be described as a chain or network of activities.

The specifications of both ways to express sequences are described below.

7.3 Graphical representation of activity sequences

In an *IDEF0 Diagram* an activity or kind of activity is denoted by a rectangular *Process box*:

The name of the activity and optionally its code (alias, when available) is written in the box. The size of the boxes on a page shall be identical and not less than 10x15 mm, and depends on the size of the wrapped text. The character size of the text shall be selectable once for all pages.

The activity is often denoted as the ‘function’ being the activity or process that is performed by or enabled by (= the function of) the mechanism. Thus a process box is usually depicted as follows:

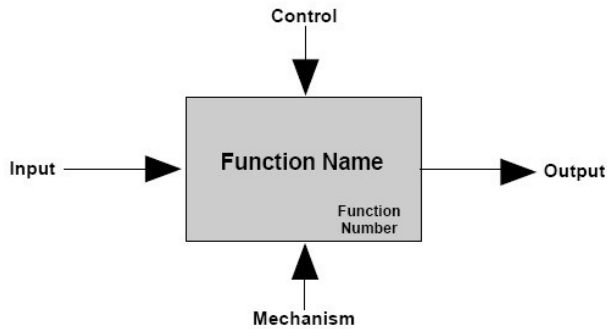


Figure 39, A process box in IDEF0

In IDEF0 diagrams physical objects are represented by *Process connectors (lines)*.

- Each Process connector connects a box with another box or with the outside ‘world’ (the edge of a page). Each connector has a direction from output to input and an arrow at the end that represents an input, control, mechanism or subject.
- Each Process box has various ‘ports’. A port is a physical point that can act as a connect point for a Process connector on a Process box symbol. In IDEF0 diagrams ports are located at the middle of the sides of each Process box.

A Process connector (physical object or kind of physical object) starts or end at a specific port on a Process box, dependent on its kind of role towards the activity (and thus dependent on its kinds of relation):

An Input ends at a Process input port at the left hand side of a Process box.

An Output starts at a Process output port at the right hand side of a Process box.

A Controller ends at a Process control port at the upper side of a Process box.

A Mechanism (or one of its subtypes performer, enabler or aid) ends at a Process mechanism port at the lower side of a Process box.

A Subject end at a Process input port at the left hand side of a Process box and also starts at a Process output port at the right hand side of the same Process box.

In WFS diagrams physical objects are indicated by *Physical object boxes* at the left hand side of the diagram. They represent the physical object (e.g. a system) that is subjected to the various activities.

7.4 Physical objects involved in occurrences

7.4.1 Kinds of involvements

If only the two roles, the roles of the performer and of the subject are of interest, then it is possible to express the occurrence as a binary relation. For example, the fact that pump P-101 pumps fluid S-1 can be expressed as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
1	P-101	101	194201	pumping	2	S-1

The UID of the fact (in the example fact 101) is the identifier of the occurrence. In this case it is not given a name, but could be called ‘pumping of S-1’ as a concatenation of the relation type and the subject.

If the roles of more objects are relevant, then a number of facts are required to specify every object that is involved in the occurrence with its specific role or its roles.

Physical objects or *kinds of physical objects* can play various roles in the activities, processes and events. Therefore, physical objects that are involved in occurrences are related to those occurrences by kinds of relations that imply those roles and are defined by those kind of roles and the required kinds of role players.

Examples of involved physical objects are: buildings, factories, roads, vehicles, equipment, instruments, pipes, streams, signals, electronic files, organizations, persons, etc.

The kinds of relations that are used in a formal language to express that and how physical objects are involved in individual activities are all subtypes of the relation type <is involved in>. A number of them are illustrated in Figure 40.

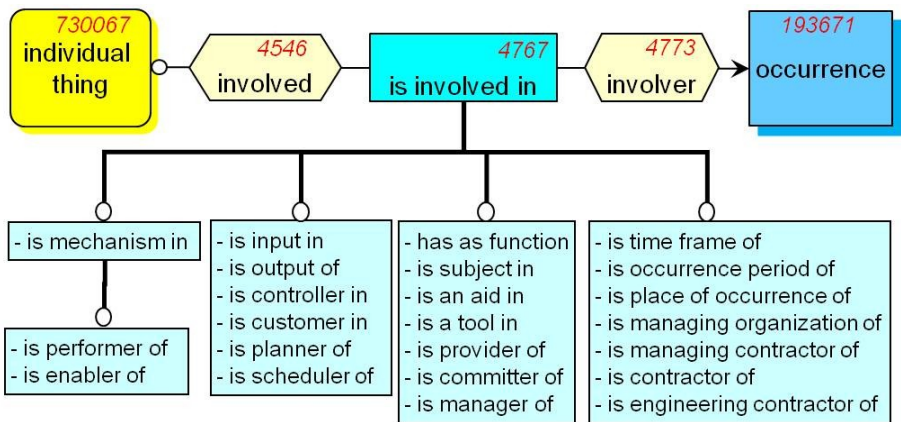


Figure 40, kinds of involvements of individual physical objects in individual occurrences

The relations between activities and physical objects that are used in IDEF0 diagrams are described in a formal language as follows:

activity	has as input	physical object
activity	has as output	physical object
activity	is controlled by	physical object
activity	uses as mechanism	physical object
activity	has as subject	physical object

Equivalent kinds of relations that are used to classify relations between kinds of activities and kinds of physical objects that describe knowledge about possibilities are the following:

kind of activity	can have as input a	kind of physical object
kind of activity	can have as output a	kind of physical object
kind of activity	can be controlled by a	kind of physical object
kind of activity	can use as mechanism a	kind of physical object
kind of activity	can have as subject a	kind of physical object

It is also possible to specify that one or more of the above kinds of relations may be constrained to a requirement that is only applicable in a particular context. In that case the formal language phrases begin with <shall ...> instead of <can ...>. For example:

pumping shall have as input a stream of liquid

Some kinds of activities have relations that are by definition the case. Such activities are defined by relations where the formal language phrase begins with <is/has by definition ...> instead of <can ...>. For example:

water treatment has by definition as input a stream of water

A physical object or kind of physical object can have a kind of role in these relations that either an Input, Output, Controller, Mechanism (with its subtypes performer, enabler or aid) or Subject.

Note, a kind of relation may also be a subtype of one of the above kinds of relations and a kind of role may also be a subtype of one of the above kinds of roles. Standard subtypes are available from the formal dictionary.

7.4.2 Properties of physical objects

Activities involve various physical objects in different roles. Each physical object that plays a role in an activity can be defined in the form of a product model, independent of the activity, just as any other physical object.

Note that most properties are properties of the physical objects and not properties of the activities.

For example, the model of a casting process includes a wall that is casted from a batch of concrete, whereas the batch as well as the wall has its own properties.

This is described for a kind of activity ‘concrete casting’ as follows:

concrete casting	has by definition as input a	batch of concrete
concrete casting	can have as output a	concrete wall
batch of concrete	shall have as aspect a	volume
concrete wall	is a specialization of	wall
wall	shall have as aspect a	height
wall	shall have as aspect a	width
wall	shall have as aspect a	thickness
etc.		

Note that the fourth line, on which it is specified that a concrete wall <is a specialization of> wall, is already included in the formal language taxonomic dictionary and implies that the facts about a wall are inherited by a concrete wall. Thus the requirements that a wall shall have a height, a width and a thickness (as is specified for example in the three lines below that line) are inherited by a concrete wall.

7.4.3 Physical objects subjected to activities

Sometimes an activity ‘operates on’ a physical object, which means that the physical object has a role as input and has also a role as output of the same

activity. In addition to that, that same physical object will also be an output of a previous activity and an input of following activity. This means that in such a case the physical object has two times a role as input and two times a role as output. The specification of the topology of such a process might be ambiguous. Such a situation can unambiguously be expressed in a formal language in any of the following two methods:

- 1) By expressing that the physical object <is subject in> the activity. This means that the activity ‘operates on’ the physical object, whereas the physical object before and after the activity is the same physical object.
- 2) By distinguishing the pre-state and the post-state of the physical object as separate states (‘temporal parts’) of the physical object. This means that the pre-state and the post-state of the physical object are defined as different ‘objects’, each with its own aspects (properties), although the whole and the parts share aspects that are not redefined per state. The relation between the whole physical object S1 and its states St1 and St2 is expressed by the relation type as is illustrated in the following expression:

Pre-state St1 is a state of physical object S1

The difference between the two methods is clarified using the example process in Figure 41.

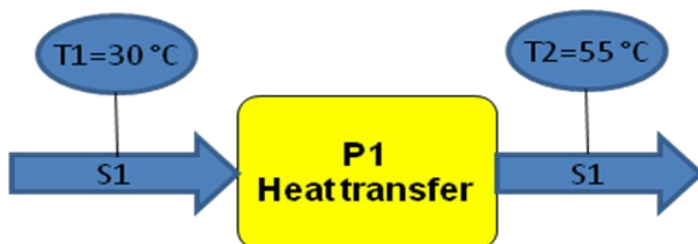


Figure 41, A heat transfer process

The example of Figure 41 illustrates that a stream S1 is input a heat transfer process P1, whereas in the state St1 before being heated the stream has a temperature T1 of 30 deg C. The same S1 is output of that process, but in a heated state St2, whereas its temperature T2 is 55 deg C after being heated.

In the first of the two modeling methods to describe the process there is only one physical object S1 that ‘passes through’ the activity and that possesses all properties before as well as after the activity. This means that the activity P1 ‘operates on’ S1, which is specified as follows:

S1 is subject in P1

By using the inverse phrase for the same kind of relation the statement can be expressed also as:

P1 has as subject S1

For kinds of activities on kinds of physical objects this would become:

stream can be subject in a heat transfer process

In case of a batch process, T1 occurs earlier than T2. Therefore, the fluid can be defined to have one temperature (T of S1) that varies over time and that is quantified by different values at different moments in time.

This can be specified as follows:

S1	has as aspect	T of S1	
T of S1	is classified as a	temperature	
T of S1	has on scale a value equal to	30	deg C
	at	10.50 h	
T of S1	has on scale a value equal to	55	deg C
	at	11.30 h	

In such a case the temperature 'T of S1' typically means the (average) temperature of the whole batch of fluid (a physical object), which temperature is independent of a location in the physical object.

However, it may be required to record that the heated object has a temperature distribution varying by location (at the same moment in time). This is also the case in a continuous process in which a part of the fluid is not yet heated and another part of the fluid is already heated. This means that the temperatures upstream and downstream of the heating process are temperatures at different locations. Typically they are both measured continuously, thus both at the same time. In those cases it is required to define either different temperatures at different locations, or different parts of the fluid, each part with its own temperature. For example, different temperatures at different locations are defined as follows:

S1	has as aspect	T of S1 at inlet	
S1	has as aspect	T of S1 at outlet	
T of S1 at inlet	is classified as a	temperature	
T of S1 at outlet	is classified as a	temperature	
T of S1 at inlet	has on scale a value equal to	30	deg C
	at	11.30 h	
T of S1 at outlet	has on scale a value equal to	55	deg C
	at	11.30 h	

Note that in the above example the two temperatures are both measured at the same time.

The second method to describe the process implies that there are two different ‘objects’ involved, each of which possesses its own aspects. This is done by distinguishing the stream as it is in the pre-heated state from the stream as it is in the heated state. This is done by defining two ‘states’ of the stream, whereas one has a role as input and the other has a role as output. This is specified as follows:

cold S1	is a state of physical object	S1
heated S1	is a state of physical object	S1
cold S1	is an input in	P1
heated S1	is an output of	P1

In this second method the two states of the physical object each have their own properties.

For example:

cold S1	has as aspect	T of cold S1
T of cold S1	is classified as a	temperature
T of cold S1	has on scale a value equal to	30 deg C
	at	11.30 h
heated S1	has as aspect	T of heated S1
T of heated S1	is classified as a	temperature
T of heated S1	has on scale a value equal to	55 deg C
	at	11.30 h

The definition of a ‘physical object in a state’ implies that the aspects of the physical object as a whole are also aspects of the states, as long as they are not superseded by the aspects of the state. For example, if S1 has a pressure of 3 bar, whereas for the states there is no pressure specified, then cold S1 as well as heated S1 are assumed to have a pressure of 3 bar.

Both models of the activity shall result in identical diagrams that represent the process, such as in IDEF0 diagrams or PFS’s (process flow schemes).

7.5 Sequences of activities in IDEF0

Activities can be decomposed into sub-activities, just as other objects can be decomposed into parts. For example, a *project* is a large activity (planned or actual) which can be decomposed into project phases, which on their turn can be decomposed into actions which are smaller activities. Such a set of activities are usually put in sequence, taking into account the available

resources and other constraints, for example by a ‘critical path planning’ activity. The result is a network of activities, such as illustrated in Figure 42.

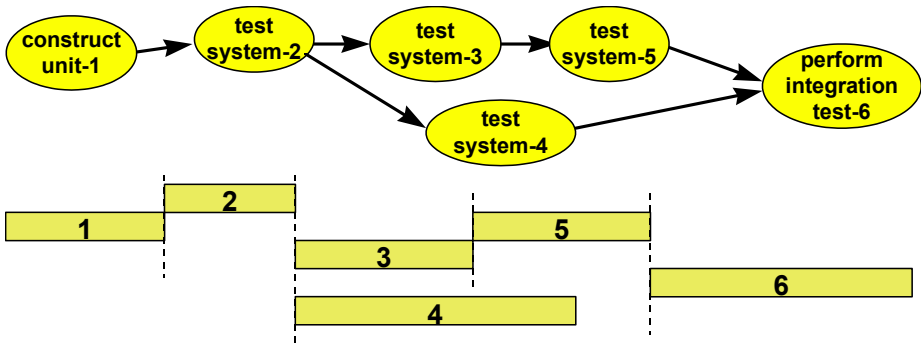


Figure 42, Critical path planning of activities

This network can be documented by succession relations between activities as follows:

action-1	is succeeded by	action-2
action-2	is succeeded by	action-3
action-2	is succeeded by	action-4
action-3	is succeeded by	action-5
action-4	is succeeded by	action-6
action-5	is succeeded by	action-6

Each of those activities can be further defined in terms of use of resources and duration. For example:

action-1	requires a role as	performer	by object	person-1
action-1	has property	20 h	which is classified as	aman-hour

The sequence of activities in an IDEF0 diagram follows from the roles of the physical objects in the activities.

However, the sequence may also be specified explicitly. If both kind of data occur, then the explicit specification is dominant over the derived specification.

An explicit activity sequence is specified as follows:

A2	occurs after	A1

and for the kinds of activities:

tightness test	can occur after a	cleaning

Whereas also subtypes and inverse phrases of those relation types may appear (for example <occurs before> or <shall occur after a>.

7.6 Duration, begin and end, where and when

Occurrences, such as activities and processes typically have a particular duration and a date-time of begin and end.

The various relations between the occurrence and the time require different kinds of relationships as is illustrated in Figure 43.

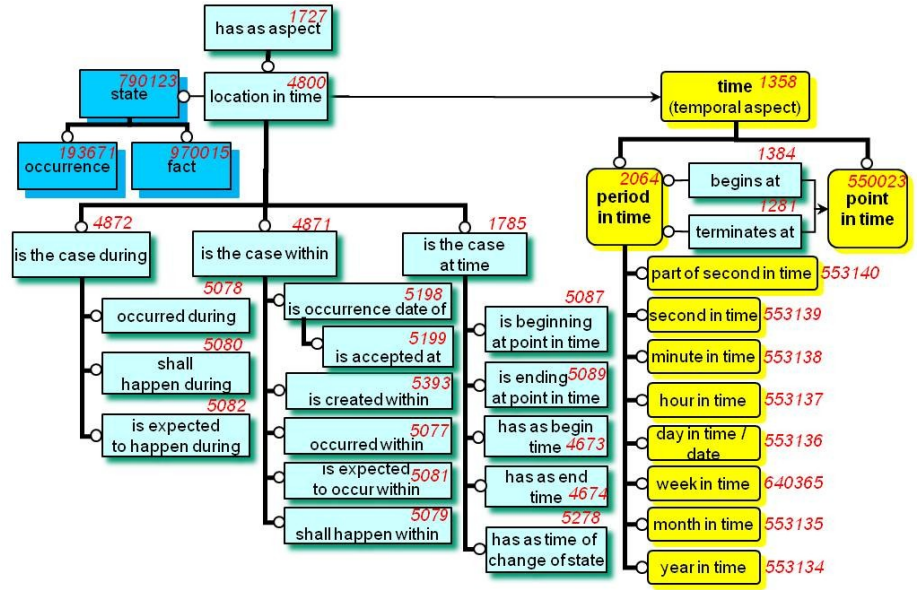


Figure 43, Relations between states, occurrences and time

A duration of an occurrence is an aspect that is possessed by the activity. It is recorded as follows:

A1	has as aspect	T1
T1	is classified as a	duration
T1	is quantified on scale as equal to	50 min

The location where something began, or will begin or end is always modeled in the present tense, whereas from the time aspect it should be concluded whether the begin or end is in the past, present or future. The model is as follows:

A1	is beginning at location	place A
A1	is ending at location	place B

There are various ways in which activities can be related to time.

An activity can begin or happen in time in the following ways:

- begin or end at a point in time (which may be indicated by a period in time, such as a minute)
- happen within a period of time (somewhere within that period, for example on a particular day)
- happen during a specific period of time, from the start to the end of the period.

7.7 Cause and effect

Usually we say that an occurrence is initiated (caused or triggered) by another occurrence. However, more precise it may be that the occurrence (or its beginning or ending) triggers the creation (start) or termination (end) of the other occurrence. This means for an accurate modeling of causes and effects that we need to distinguish between an occurrence (activity, process or event) and its creation process and its termination process, which both are parts of the occurrence. Then we can express also that a creation or termination (that is part of an occurrence) is initiated by the creation or termination process of the same or another occurrence. These variations are illustrated in Figure 44.

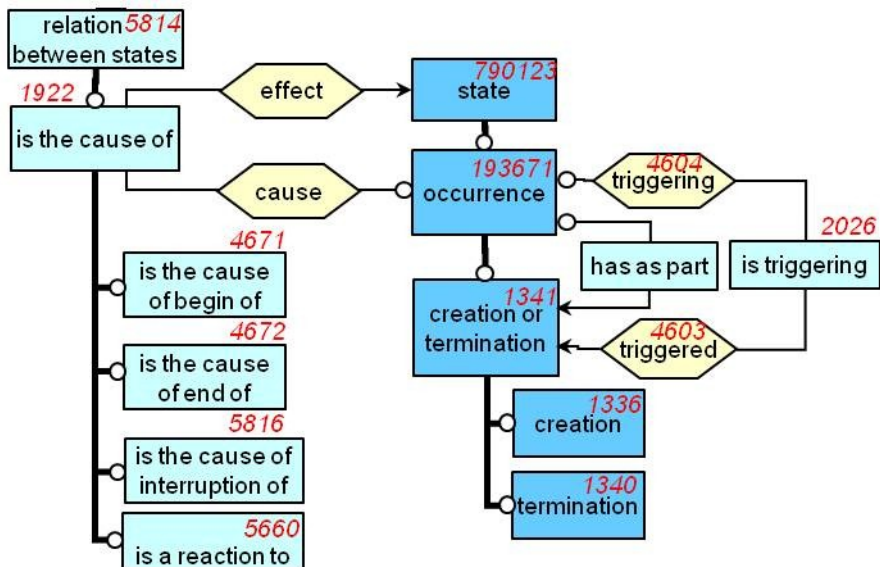


Figure 44, Kinds of relations to model cause and effect

Figure 44 illustrates that an occurrence or one of its subtypes (a creation or termination) can cause a state or one of its subtypes (typically another occurrence). This can be refined by using one of the subtypes of <is the cause of>, to express that an occurrence is the cause of the begin or end or interruption of a state or occurrence. The figure also illustrates that a creation or termination may be initiated or triggered by an occurrence, without being its cause.

7.8 States and state changes

States can be static as well as dynamic. Static states are states that are described by aspects with values that do not change over time. Dynamic states are typically transitions from one steady state to another, but they can also be continuous changing processes or occurrences. A special kind of dynamic state is a steady state, which is for example a fluid flowing with constant flow rate, or a train driving at constant speed. In steady state processes most aspects do not change, but the total volume transported or the position of the moving object is changing with a constant rate.

The modeling of timing of states has many options each defined by its own kind of relation. The time can be specified by a single value, which nevertheless specifies a (usually short) period in time or it can be specified by a period that is defined by two time values. This is illustrated in Figure 45.

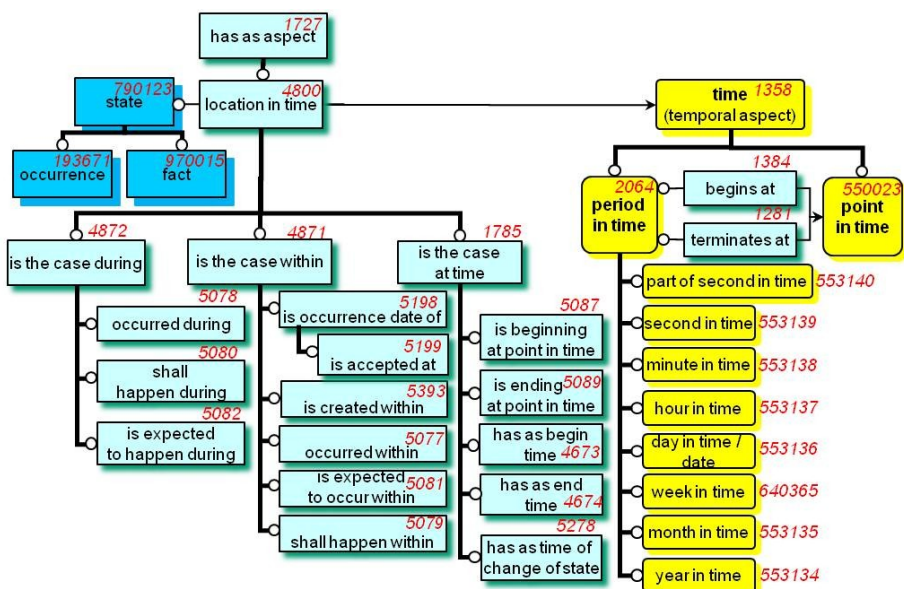


Figure 45, Model of states and state changes

The relation type that can be used to specify that a physical object is in a state (which is called a ‘temporal part’ in ISO 15926-2) relates a physical object to a state. For each state it can be expressed

- when the state begins
- where the state begins
- where the state ends
- when the state ends
- what the sequence of states is,
- which state change took place
- when a (sudden) state change took place

The time sequence of states cause that they can overlap, but there can also be a gap between end and begin. The various options are illustrated in Figure 46.

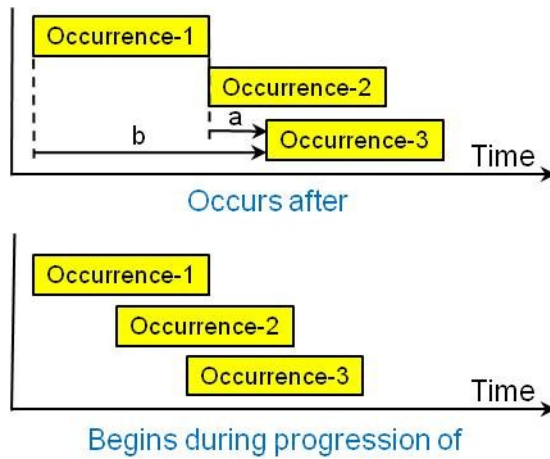


Figure 46, Gap and overlap between states and occurrences

The location where a state or occurrence takes place can be modeled by using the relation types of Figure 47.

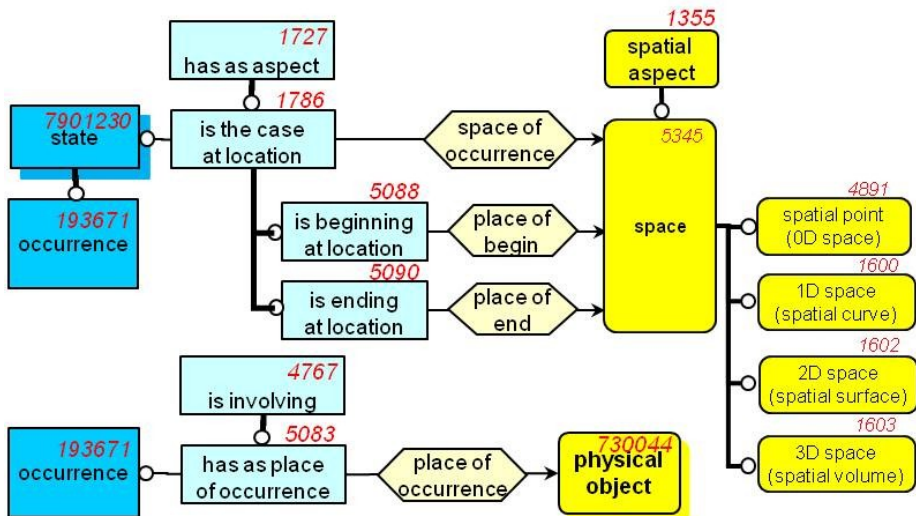


Figure 47, Locations of states and occurrences

These possible expressions for states are also applicable for occurrences, because an occurrence is a subtype of a state.

The relations between states, such as state changes, can be modeled using the relation types in Figure 48.

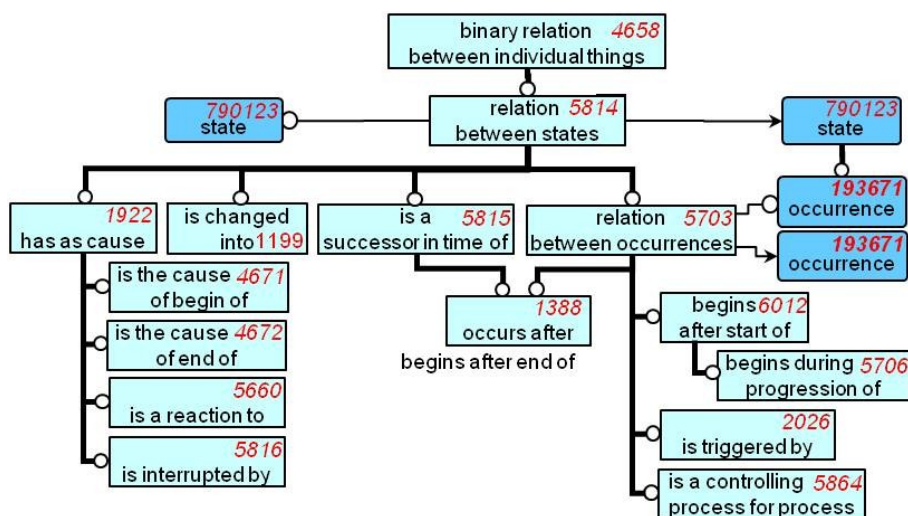


Figure 48, Relations between states

7.9 Process designs and control

7.9.1 Processes

Processes are defined in a formal language as subtypes of occurrences. So the various types of processes, such as physical processes e.g. distillation, transport, (bio)chemical processes, e.g. chemical reaction, bio-treating, etc. are defined in the dictionary in the subject area ‘occurrence’.

This means that processes can be described as a sequence of occurrences which operate on fluid or solid process material. It is not necessary to include in that description the physical items that enable the process to take place. For example, a distillation process can be modeled as follows:

distill-3	is classified as a	distillation	
fluid-4	has aspect	flowrate-4	
flowrate-4	has on scale a value equal to	30	m3/h
flowrate-4	is classified as a	volume flow rate	
fluid-4	is input in	stage-8	
fluid-5	is output of	distill-3	
fluid-5	has a role as a	top product	
fluid-6	is output of	distill-3	
fluid-6	has a role as a	bottom product	
fluid-7	is input in	distill-3	
fluid-7	has a role as a	reflux	

distill-3	has as aspect	N stages	
N stages	is classified as	number of stages	
N stages	is quantified as	15	
Stage-8	is a part of	distill-3	
Stage-8	has as aspect	Stage nr	
Stage nr	is classified as a	stage number	
Stage nr	is quantified as	8	

Note that the roles top product, bottom product and reflux are in fact subtypes of input and output.

7.9.2 Batch processes & state transitions

Batch processes consume and produce discrete batches of fluids. During a production cycle the process can be ‘steady state’ (see next paragraph), but the process as a whole is characterized by the fact that it is moving from one process state to another. This type of transition state operation is also a characteristic of *start-up* and *shut-down* of a unit as well as *mode changes* which are all parts of processes that are characterized by a change from one state to another.

A typical example of a batch processes is illustrated in Figure 49, where a batch of fluid is transported from storage and loaded into a reactor, then heated, after which a reaction takes place (which requires a residence time), after which the fluid is cooled and then unloaded and transported to another storage facility.

Such a batch process can be described by a sequence of ‘activities’ where the fluid is subject in each activity. It is useful to distinguish two slightly different situations: one where a fluid has a role as feed in a process and *another* fluid has later a role as product from that process, and other situations where the same fluid has a role as input to as well as output from the same process. In the latter case we only need to specify that the fluid has a role as subject in one or a sequence of activities. Although a formal language does not prescribe that this distinction should be made, it is done here to illustrate the two options.

Note that the role ‘feed’ is a subtype of ‘input’ and the role ‘product’ is a subtype of ‘output’ according to the hierarchy of roles in the dictionary.

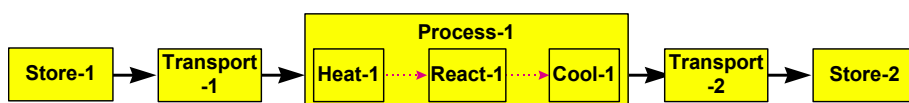


Figure 49, A batch process

This topology can be described by using the various roles of the fluids relative to the processes (activities) as follows:

Name of left hand object	Name of kind of role	Name of relation type	Name of right hand object
fluid-1	product	is output of	store-1
fluid-1	subjected	is subjected to	transport-1
fluid-1	feed	is input in	process-1
fluid-2	product	is output of	process-1
fluid-2	subjected	is subjected to	transport-2
fluid-2	feed	is input in	store-2

Process-1 can be decomposed as follows:

heat-1	is a part of	process-1
react-1	is a part of	process-1
cool-1	is a part of	process-1

whereas:

Name of left hand object	Name of kind of role	Name of relation type	Name of right hand object
fluid-1	subjected	is subjected to	heat-1
fluid-1	feed	is input in	react-1
fluid-2	product	is output of	react-1
fluid-2	subjected	is subjected to	cool-1

Note that the role ‘subjected’ implies that the fluid that enters a process stage is the same fluid as leaves that process stage, only its conditions at the inlet, such as temperature and pressure will be different from the conditions later at the outlet. Thus, it implies that the same fluid has a role as ‘feed’ to as well as ‘product’ from the process stage. The fluid that enters the reaction process (the feed to react-1) is different from the fluid that is product from that reaction. Therefore, fluid-1 has the role as feed and fluid-2 has the role as product from react-1.

Finally the sequence of process stages can be described as follows:

transport-1	is succeeded by	process-1
heat-1	is succeeded by	react-1
react-1	is succeeded by	cool-1
cool-1	is succeeded by	transport-2
transport-2	is succeeded by	store-2

7.9.3 Process cases

Process cases are sub-processes that take place under various different conditions and capacities. They are often defined for a design to ensure that hardware is designed such that it is suitable to operate under the various conditions and capacities. Process cases thus typically include extreme operating conditions, such as minimum and maximum flow rates, pressures and temperatures. A characteristic is that the various input and output materials (streams) in the different process cases are used in combination with usually the same hardware. So a process case is a sub-process that is an imaginary part of a bigger process that uses different inputs and outputs, but may involve the same hardware (equipment) as other process cases. Thus process cases are described in the same way as other processes. In a model it will be visible that pieces of hardware appear in various roles in different process cases in which different inputs and outputs apply. As the inputs and outputs are not defined as input and outputs for the hardware, but for the different processes, there is no ambiguity of which input belongs to which output.

7.9.4 Energy transport and conversion processes

Energy is not just consumed or produced by equipment, as is often suggested, but energy is transported from one place to another and converted by equipment from one form into another. For example, a pump is not just a pressure and velocity increaser, but primarily an energy converter. Conversion from one form of energy to another is a process that can take place with the use of various types of equipment.

For example, an electrically driven pump starts with electrical energy supplied to the electric motor. There, the electrical energy is transformed into kinetic energy of rotation, which is transported via the shaft and a gear box to the shaft of the pump, where the kinetic energy is transformed into potential energy (pressure) of a fluid, which is partly transformed into kinetic energy (velocity) of the fluid.

This can be described in a similar way as the transport and conversion of fluids, by treating energy as a physical object. At the start and the end of such a process the energy is derived from and given to an energy carrier, by changing its properties, such as its voltage or its enthalpy.

7.9.5 Process Design & Process Simulation

Process design starts with the definition of a series of unit operations which are physical or (bio)chemical processes which transform fluids or solids from one state to another. Usually a process engineer assumes that raw material is

available for the main process at certain input conditions. The fact that it arrives by a transport process and is then stored in a storage facility such as a tank is usually only later made explicit.

For example, if the first main process is a steady state distillation process, then the feed stream (a physical object) is pumped via a heating process to a distillation process. These processes can be simulated as a series of operations on the fluid, such as a pressure increase, a heat transfer and a series of vapor-liquid equilibrium stages of the fluid in counter current flow. This can be done even without any equipment involved.

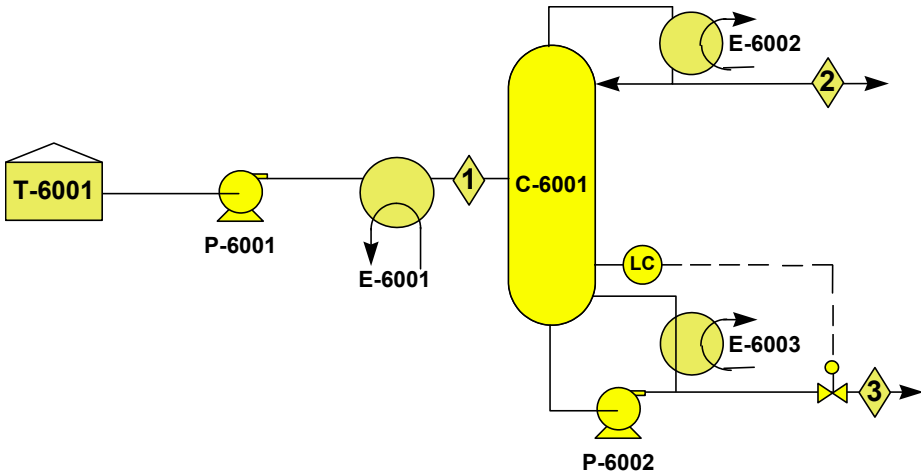


Figure 50, Simple distillation process

The description of the process can be formally expressed as follows:

Name of left hand object	Name of relation type	Name of right hand object	Unit of measure	Validity location
stream-1	is classified as a	mixture		
pumping-1	is classified as a	pumping		
heating-1	is classified as a	heating		
distill-1	is classified as a	distillation		
stream-1	is input in	pumping-1		
stream-1	is output of	pumping-1		
stream-1	is input in	heating-1		
stream-1	is output of	heating-1		
stream-1	is input in	distill-1		
stream-2	is output of	distill-1		
stream-2	has a role as a	top product		
stream-3	is output of	distill-1		

stream-3	has a role as a	bottom product		
stream-1	has aspect	p1		
p1	is classified as	pressure		
p1	has on scale a value equal to	1	bar	T-6001
stream-1	has aspect	T1		
T1	is classified as	temperature		
T1	has on scale a value equal to	20	C	T-6001
stream-1	has aspect	v1		
v1	is classified as	volume flow rate		
v1	has on scale a value equal to	9	m3/h	T-6001
pumping-1	has aspect	dp		
dp	is classified as a	differential pressure		
dp	has on scale a value equal to	3	bar	
heating-1	has aspect	hfl		
hfl	is classified as a	heat flow rate		
hfl	has on scale a value equal to	5	kJ/h	
distill-1	has aspect	N1		
N1	is classified as a	number of stages		
N1	is quantified as	15		
etc.				

Note that the location where the stream-1 properties have their values is expressed using the relation of the kind ‘is the case at location’ between the fact and a location or point in space. The ‘location in space’ is here the physical object T-6001.

Further details of measurement and observations at a particular moment in time and at a particular locations is described in chapter 8.

7.9.6 Process Control

The control of a process starts with the definition of 'functions' to be performed. For example, a level control (LC) in a vessel that is controlled via the control of the flow rate of the liquid pumped out of the vessel (see Figure 50).

Control, as well as measurement and actuation are occurrences. Assume that we describe the level control of Figure 50 on a high level, and then we describe only one activity, which is performed by the control loop as a whole. This can be done as follows. The level of the liquid in the vessel, being a

property of that liquid, is represented by a signal in a control loop. This signal has a role as input in the control activity, whereas another signal has a role as output of that control activity. That second signal represents the flow rate of the fluid at the outlet of the pump.

The components of the process in the control loop can then be expressed as follows:

LC-1	is classified as a	level control
loop-1	is classified as a	control loop
loop-1	is performer of	LC-1
Liquid-1	has aspect	level-1
level-1	is classified as a	level
Liquid-2	has aspect	flowrate-1
flowrate-1	is classified as a	volume flow rate
Signal-1	is input in	LC-1
Signal-2	is output of	LC-1

This is schematically illustrated in Figure 51.

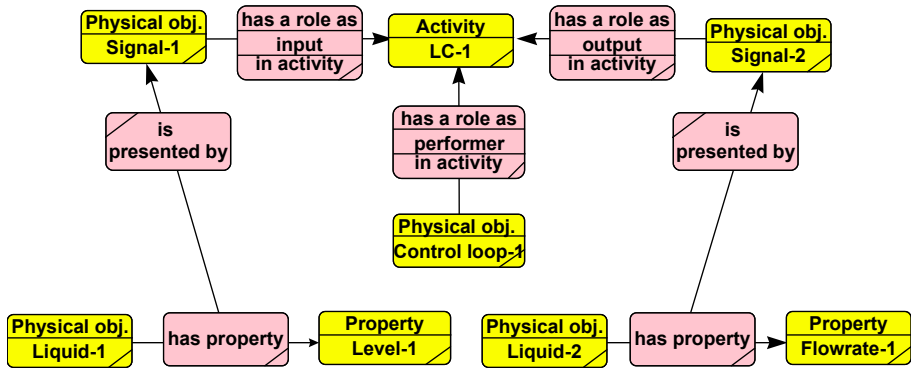


Figure 51, Instances of Objects involved in Level Control

In the above example, we introduced two signals, because in the actual world activities can only have physical objects as input and output. So, the dashed lines in Figure 50 represent physical signals, whereas the circle that contains the characters LC represent the activity performed by the control loop as a whole.

Note that at this stage and abstraction level the materials that perform the various sub-activities, such as a measuring activity, a regulation activity and an actuation activity are not mentioned. However, from the schema of Figure 50 it can be seen that it is intended to influence the flow rate using a control valve. In other words, the flow rate is caused by the throttling activity of the

final control device, in this case a control valve. The above description is also valid for control of the flow rate in another way, for example via the speed of the motor of the pump. A control engineer also knows that the first signal is generated by a measuring activity performed by a transmitter. And that the regulation activity can be performed for example by a PID controller. If we add the measuring activity we can extent the model as is illustrated in

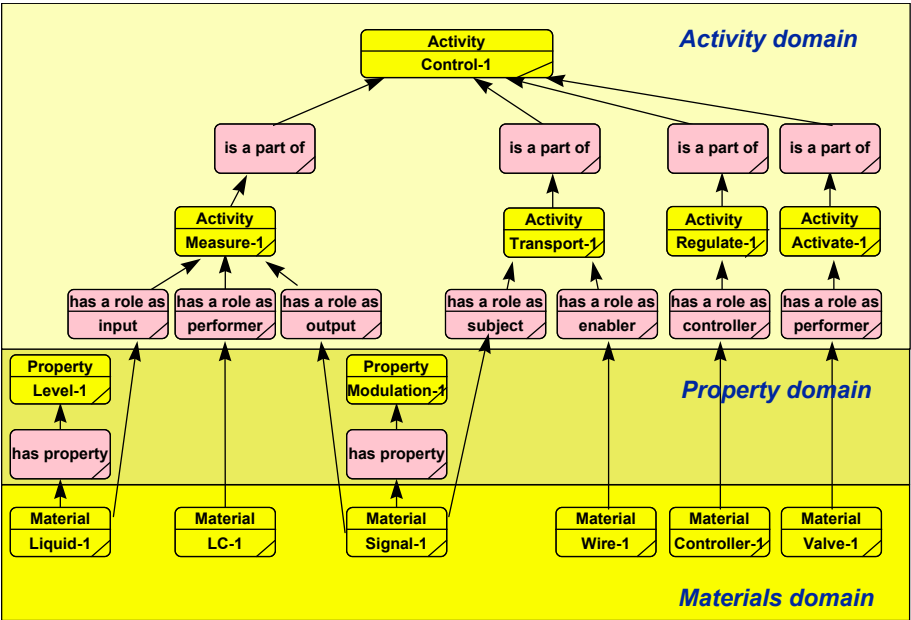


Figure 52, Control loop details

These details of the control process can be modeled as follows:

measure-1	is a part of	LC-1
measure-1	is classified as a	measuring
TR-1	is classified as a	level meter
TR-1	is a performer of	measure-1
liquid-1	is subjected to	measure-1
signal-1	is output of	measure-1

The role of the control valve can be described as a throttling activity with signal-2 in the role of set point input. This can be done as follows (see also

):

FC	is a part of	LC-1
FC	is classified as a	throttle (and as actuate?)
CV	is classified as a	control valve
CV	is a performer of	FC

A PID controller can be the performer of the regulation activity.

R-1	is classified as a	regulate
PID-1	is classified as a	PID controller
PID-1	is controller in	R-1
signal-1	is input in	R-1
signal-2	is output of	R-1

Finally we can model the transport activity of the signal through a wire as follows:

transport-1	is classified as a	transport
core-1	is classified as a	core
core-1	is enabler of	transport-1
core-1	is a part of	cable-1
etc.		

7.9.7 Control logic and function blocks

Loop diagrams describe control logic usually in the form of ‘function blocks’ and lines between them which represent the ‘information’ flowing from one function block to another.

From the description in the previous paragraph it can be clear that these function blocks are activities, which are usually defined by algorithms. The lines between the function blocks represent signals which represent information. The information consists usually of just one relation, such as a property of a substance or a state of an item.

The topology of the control logic is defined by the description of the roles of the signals as various subtypes of input and output in those activities.

Note, that function blocks are often called functions, whereas the algorithms are also often called functions. To avoid confusion in this document we use the terms activity and algorithm respectively.

Standard control activities (‘function blocks’) are defined in IEC 1131-3 and extended by the VDI as documented in VDI/VDE 3696. These standard control activities are included in the dictionary, in order to support the definition of control logic in a system (and vendor) independent way. The function blocks ABS, ARCCOS and ADD in Figure 53 are examples of

them. The first two have both the same input roles for signals called U and EN and output roles V and ENO. The latter has several other input signals, called U1, U2, etc.

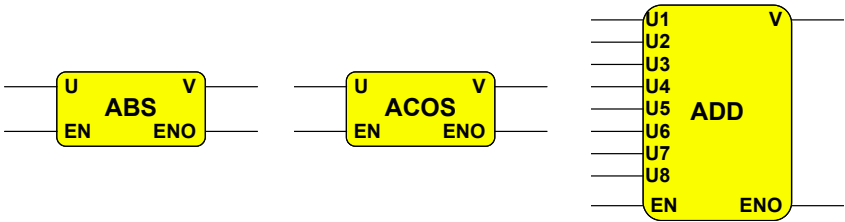


Figure 53, Control Function block examples

In the Gellish Formal English dictionary these standard roles are defined with their full name and definition, which is abbreviated as follows:

U	is a kind of	input
EN	is a kind of	input
V	is a kind of	output
ENO	is a kind of	output
U1	is a kind of	input
U2	is a kind of	input
etc.		

Furthermore it can be expressed that:

signal	can have a role as a	input
signal	can have a role as a	output

The fact that a signal can have a role as a input and that U, EN, etc. are defined as a kind of input, means because of the inheritance rules that:

signal	can have a role as a	U
signal	can have a role as a	EN
signal	can have a role as a	U1
etc.		

The standard activities (function blocks) are defined in the dictionary as follows:

ABS	is a kind of	execute algorithm
ABS	is defined by	f1
f1	is a qualitative subtype of	algorithm
		y = abs(x)

And finally it is defined which roles are recognized per activity:

U	is by definition a role in a	ABS
EN	is by definition a role in a	ABS
V	is by definition a role in a	ABS
ENO	is by definition a role in a	ABS
U	is by definition a role in a	ACOS
EN	is by definition a role in a	ACOS
etc.		

Because all function blocks are predefined in this way in the dictionary, it is possible to define control logic by defining specific or typical signals and function blocks by classifying them as a signal and as ABS, ACOS, etc. and then define the control logic topology as follows:

signal-1	with role	U	is input in	ABS-1
signal-2	with role	V	is output of	ABS-1
signal-2	with role	U	is input in	ACOS-1
etc.				

7.10 Fabrication and assembly

Fabrication consists of a sequence of activities in which raw material or intermediate products are converted into components, which are then assembled into final products. This implies temporal succession of activities and occurrences, classified according to their specific nature. Each of those activities involves various objects as resources in various roles, such as performer, enabler, subject, tool, input, output, etc.

The design object with its associated specification information is involved in the fabrication and assembly activity as specification, whereas the actual (real) physical object has the role of result (a subtype of output) of that activity. The actual physical object 'is a realization of' the imagined design object. After fabrication and assembly the actual object is subject to a test activity to verify whether it satisfies the design specification and can execute the test activities. If that is the case the actual physical object 'fulfils' the requirements expressed by the design object and the test activities which it should be able to perform: its capabilities are satisfactory. Then the physical object can be 'approved for operation'.

Assembly of machines and equipment produced in series is often done in an assembly line where a temporal sequence of activities takes place. Those activities are defined as a sequence of typical activities. Each typical activity involves a typical object which should be connected to another typical object.

Every typical activity can be defined by a string of text which gives

instructions of for the execution of the assembly activity. Such a string of *text* and can be written according to various other standards, such as SGML, HTML or XML. In that case it is necessary to classify the text as appropriate after which an application could present the text to a user by using any text processor which is able to present text in such a format.

An actual activity takes such a typical activity as a prescription of how and in which sequence the actual activity should be executed.

7.11 Construction & commissioning

The commissioning (testing), start-up (CSU) and initial operation activities are executed per (sub) *system* and start-up block. The project CSU schedule therefore needs to be developed by defining *commissioning systems* and various levels of subsystems and by identifying activity sequences that need to be executed per (sub)system. These systems and activities need to be integrated in the Facility Information Model to enable to link them to the documentations of the various systems and equipment and to enable their re-use during maintenance and turnarounds.

Each system shall be defined as part of a facility decomposition hierarchy, also called an asset breakdown structure. This means that for each system the following aspects shall be specified:

- It shall be indicated by a unique identifier and one or more names
- It shall be specified to be a part of a higher level assembly
- It shall be classified by a standard kind of system
- Its components shall be specified.

Those various aspects shall be specified as is described in the following paragraphs.

The facts that describe the identification, composition and classifications of the systems and their components are all part of the *System Definition data set*.

For start-up it may be necessary to combine systems into start-up blocks. These are assemblies that can only be started up and operated as a whole.

Start-up blocks can also be integrated in a system composition data set by adding lines that specify which systems are parts of which start up blocks. For example, the fact that the HP and MP Steam distribution systems together form Start-up block SU-1 can be expressed as follows:

UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
103	HP Steam distribution system	1,190	is a part of	200	SU-1
104	MP Steam distribution system	1,190	is a part of	200	SU-1

Furthermore, it shall be specified which systems are described in which documents. This includes the drawings (P&ID's) on which the system scope is specified, but also other documents that provide information about the systems. This will enable to search for all documents about the system during CSU and at other occasions. The system to documents relations shall be specified together with a relation between the document and the electronic files (e.g. DWG and PDF files) in which the documents are physically stored and the location where the file is stored. An example of how this should be done is presented in Table 5.

UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
102	Steam distribution system	1,727	has as aspect	110	Colour of SSD on P&ID
110	Colour of SSD on P&ID	1,225	is classified as a	550,003	colour
110	Colour of SSD on P&ID	5,020	is qualified as	551,759	red
102	Steam distribution system	5,046	is described in	107	P&ID of unit 1200
107	P&ID of unit 1200	1,726	is a qualification of	490,305	Piping and instrument diagram
107	P&ID of unit 1200	4,996	is presented on	108	P&ID of unit 1200.pdf
108	P&ID of unit 1200.pdf	1,225	is classified as a	490,533	electronic data file
108	P&ID of unit 1200.pdf	1,227	is an element of	109	C:\Systemization
109	C:\Systemization	1,225	is classified as a	492,017	directory

Table 5, Relations between system and documents

The above relation enables to use semi-intelligent PEFS's to easily find the system definitions and the properties and documents about the systems.

7.11.1 Commissioning and start-up activity definitions

Commissioning and start-up (CSU) activities and their relations to systems and resources are specified in a CSU activity definition data set.

Each CSU activity shall be identified by a unique identifier (UID) and a name (tag name). An activity name shall consist of the tag name of the item (system or component) that is subjected to the activity, concatenated with a single space, the letter A and an activity sequence number.

Each activity shall be defined by three facts:

- The activity shall be classified by a kind of activity that is selected from the common project dictionary. A classification shall be accompanied by a detailed description of the activity. A classification may be accompanied by Clarifying notes and Remarks (in a separate column that is not shown in the table below).
- The activity shall be related to the item (system or component) that is the subject of the activity.
- The activity shall be defined to be part of an aggregated activity.

Table 6 presents an example of the definition of an activity (Note: in order to save space the columns with UID of fact and UID of relation type are not shown in some of the tables below):

UID of left hand object	Name of left hand object	Description	Name of relation type	UID of right hand object	Name of right hand object
121	ST-1201 System A1	Walk down system and check isolations, etc.	is classified as a	194,567	tightness test
121	ST-1201 System A1		has as subject	105	ST-1201 System
121	ST-1201 System A1		is a part of	120	Commissioning of Unit 1200

Table 6, Activity definition table

Optionally an activity may be related to a document that contains a procedure for the execution of the activity. It is also possible that a kind of activity is related to a procedural document.

7.11.2 Relations between activities

Relationships between activities specify that an activity can only start when another activity is completed. This shall be specified by a <occurs after> relationship between the activities. This shall be documented as follows:

UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
122	ST-1201 System A2	1,388	occurs after	121	ST-1201 System A1

Table 7, Activity sequence definition

7.11.3 Duration and resources

For each activity it shall be specified what its estimated duration will be, according to which time frame ('calendar') it should be executed, how many man hours are expected and which kind of people are required. This shall be specified as follows:

UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	UoM
121	ST-1201 System A1	1,727	has as aspect	201	A1 duration	
201	A1 duration	1,225	is classified as a	550,005	duration	
201	A1 duration	5,025	has on scale a value equal to	920,269	16	h
121	ST-1201 System A1	1,727	has as aspect	202	A1 time frame	
202	A1 time frame	1,225	is classified as a	640,128	time frame	
202	A1 time frame	5,025	has on scale a value equal to	520,054	10	h/d
121	ST-1201 System A1	1,727	has as aspect	203	A1 manhours	
203	A1 manhours	1,225	is classified as a	553,520	manhours	
203	A1 manhours	5,025	has on scale a value equal to	920,269	16	h
121	ST-1201 System A1	5,684	is permitted to be performed by a	990,216	pipe fitter	

Table 8, Specification of duration and resources of activities

7.11.4 Special tools, temporary facilities and consumables

The need for special tools, temporary facilities and consumables shall be specified by relations with the required resources. Furthermore, the properties of those resources can be specified as well. For example as follows:

UID of left hand object	Name of left hand object	Name of relation type	UID of right hand object	Name of right hand object	UoM
121	ST-1201 System A1	uses as tool	210	Overhead crane	
121	ST-1201 System A1	has as input	211	Cleaning fluid	
211	Cleaning fluid	is classified as a	431,735	soap	
211	Cleaning fluid	has as aspect	212	Cleaning fluid quantity	

212	Cleaning fluid quantity	is classified as a	550,043	volume	
212	Cleaning fluid quantity	has on scale a value equal to	927,200	6	m3

Table 9, Activity tools and consumables

7.11.5 Procedures to execute activities

When activities are described in procedures that are captured in separate documents, then the CSU Activity definition data set can be extended with a relation between the activity and a document as follows:

UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
121	ST-1201 System A1	5,046	is described in	154	procedure to test the tightness of a piping system
154	procedure to test the tightness of a piping system	1,726	is a qualification of	491,000	test procedure

Table 10, Documents that contain procedures to execute activities

The procedure that describes how the activity should be executed is essentially a further decomposition of the activity (or of a kind of activity) and the specification of the sequence of the tasks. Such a procedure can either be described in a document or it can be expressed as data in another similar activity definition table.

An electronic physical file can be related to the document in the same way as is described above for the files with drawings and documents that are linked to the system that they describe.

7.12 Maintenance

Maintenance focuses on *activities* (occurrences) that have to take place and usually need to be scheduled. There may be various organizations and persons involved, as well as physical objects. A maintenance activity or process usually starts with a *request* which includes a specification or definition of a requested activity about a physical object.

For example, the request: ‘please inspect motor-1’. This request can be expressed formally as follows:

request: action-1 is classified as a inspection
 being an inspection of motor-1, because ...
 etc.

action-1	has as subject	motor-1
action-1	is performed by	operator-1

The requested activity will usually be scheduled, which means that the same action-1 is allocated to a person for execution at a certain moment, probably in a temporal sequence with another activity by that person. This can be described as follows:

person-2	is performer of	action-1
action-1	is started at	start time-1
start time-1	has on scale a value equal to	2.30 h
start time-1	is classified as a	duration
action-1	is performed after	action-0

This may lead to many other scheduled actions, such as ordering and repair, which are all described in a similar manner.

When actual actions are taking place, then they are different from the above requested and scheduled ones, because the actual activities are real observed ones, whereas the requested/scheduled activities were not observed, but were a product of a human's mind, which we call 'intended activities'.

For example, assume that the actual response to the request was not performed by person-2, but by person-3 and at a different time than scheduled. Then the actual activity can be described and related to the requested activity as follows:

action-3	is classified as a	inspection	<i>actual inspection of</i>
<i>motor-1</i>			
action-3	is a realization of	action-2	
motor-1	is involved as subject in	action-3	
person-3	is involved as performer in	action-3	
action-3	is started at	moment-2	
action-3	is terminated at	moment-3	

7.13 Scope and lay-out of diagrams of activities

Activities can be decomposed into partial activities.

In a formal language each activity composition between an individual whole and one of its parts is expressed by a <is a part of> relation or its inverse, a <has as part> relation. A conceptual composition relation between a kind of activity and a kind of part is expressed by a <can be a part of a> relation or its inverse <can have as part a> relation. These kinds of relations can be used for

composition of physical objects as well as for the composition of occurrences, such as activities, processes and events.

Activity models can be presented graphically in various kinds of diagrams. For example:

According to the IDEF0 standard for representing activity models a root diagram can be created for a selected activity and lower level diagrams can be created for the components of that activity. If the number of activities on a particular level exceeds the maximum that fits on a page, then additional pages for the same sheet shall be created.

The components of an activity are presented in a work flow sheet (WFS) also on new (lower level) diagrams. The size of a WFS sheet is usually bound to a maximum (typically A0). This limits the number of physical objects about which activities are recorded, because they determine the number of rows on a diagram. If that number exceed the maximum that fits on a sheet (60) or if the number of activities on any of the rows exceed a set maximum, then additional sheets shall be created.

All activities on a row in a WFS are about the same physical object. The sequence of presenting physical objects is by default determined by the sequence in which the physical objects are defined by a classification or specialization relation in a collection of formal language expressions.

7.13.1 Interacting with models via schematic diagrams

Pointing to a box or a symbol (lines, boxes) in a drawing means pointing to a pair of coordinates in the drawing. This should initiate that the software determines which symbol or text is closest to the coordinates of the pointing device. Thus the software identifies the graphic symbol and then should determine via the formal expressions in its database that the symbol or text represents a particular activity or a physical object.. Then the software should put that ‘object’ in focus and present facts about that object, including also documents about the object. The user can then perform actions on the presented objects. The inverse is may also be implemented, so that searching for an object might not only find drawings about the object or activity, but also the coordinates of the symbol on the drawing that represents the object or activity.

The way to model how documents are related to activities or physical objects is described in par. 5.3 Step 2, Create document models.

8 Observations and measurements modeling

8.1 Measured properties

The recording of values of properties is usually pretty straight forward. For example, the internal diameter D1 of a pipe P1 can be recorded in a standardized, computer interpretable way in a formal language as follows:

P1	is classified as a	pipe
P1	has as aspect	D1
D1	is classified as a	internal diameter
D1	has on scale a value equal to	30 mm

In these four expressions of facts there are a number of standard concepts and phrases selected from the formal dictionary: the concepts pipe, internal diameter, 30 and mm as well as the phrases that express relation types.

Computer interpretable recording of the measurement of properties becomes more complicated when the measured value becomes time and location dependent and when more information about the observation shall be recorded, such as the performer, the tools and the method.

This document provides guidance on the computer interpretable recording of measurements and observations in such complex cases.

Formal definitions of concepts that are used in this document can be found in the formal dictionary. Their didactical descriptions in this document may deviate from the formal definitions.

P.S. This document is based on the Draft ISO standard “Reference Data for Observations and Measurements”, by David Leal (2008) (ISO TC184/SC4/WG3 N2537). The definitions (reference data) that are provided in that draft frequently originate from the formal dictionary. Therefore the definitions from the formal dictionary are used instead.

8.2 Observing activities and observations

8.2.1 What is an observation and a measurement?

The term observation has two meanings:

1. An observing activity
2. Information that results from an observing activity

A resulting observation (information) is a set of statements (expressions of facts) that is produced by an observing activity. But in addition to that core we often want to record additional information about an observation, such as the performer, tools and circumstances under which the observation took place.

A measurement (measuring activity) is a specialization of observation (activity), where an instrument is used as a tool to sense and to quantify the measured value.

NOTE 1: An observing activity does not necessarily produce any statements about the observed physical object, because it can fail.

NOTE 2: An activity such as reading about Roman civilization in a library can create information about Roman civilization, but it is not an observing activity because the Roman Empire does not participate in it.

NOTE 3: An observing activity can contain sub-activities that are only indirectly related to the observation. Such a sub-activity could be the transport of a measuring device to a place where a measurement is made, or the conditioning of a sample before a measurement is made.

8.2.2 Modeling of observations and measurements

Information that can be recorded about an observing activity includes:

Observation act-1	1225	is classified as a	observation
Observation act-1	4761	has as performer	Observing person-1
Observation act-1	4763	uses as tool	Measuring instrument-1
Observation act-1	5058	is observing aspect	Observed aspect-1
Observed aspect-1	1225	is classified as a	kind of aspect
Observed aspect-1	5025	has on scale a value equal to	number UoM
	Date Observed aspect-1	5020	is qualified as
		qualitative aspect	
Observation act-1	4786	has as result	Recorded facts-1
Recorded facts-1	1225	is classified as a	collection of facts

In the above set of facts it is assumed that a number of things are already defined (classified) elsewhere, such as the classification of ‘Observing person-1’ and of ‘Measuring instrument-1’. The relation type phrases and the lower case things (in red) on the right hand side are standardized terms that are selected from the formal dictionary. The only time dependent fact is the date-time of the validity of the quantification of the observed aspect (see below).

Typically it will also be specified somewhere that the observed aspect is possessed by an observed physical object and sometime it is also recorded explicitly that the observed physical object is subject in the observation act.

These two facts can be specified as follows:

Observed physical object-1	1727	has as aspect	Observed aspect-1
Observation act-1	4760	has as subject	Observed physical object-1

The above relations are also depicted as part of Figure 54.

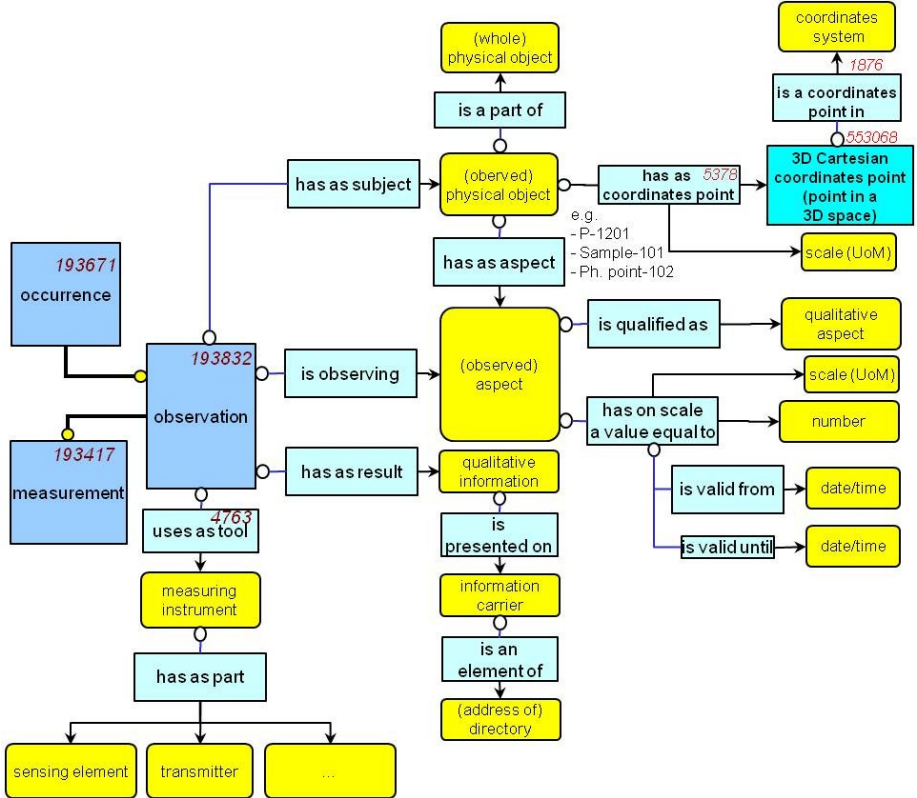


Figure 54, Model of observations

The above facts are illustrated by the following examples.

Example 1: Assume that Fred Bloggs observes the operating temperature of "MyBearing" B-1 at 2007-03-15T16:00 and records that it is "hot". The activity can be recorded as follows:

UID of	Name of left	UID	UID of	Name of relation	UID of	Name of	Date-

left hand object	hand object	of fact	relation type	type	right hand object	right hand object	time of start of validity
101	Observation-1	201	1225	is classified as a	193832	observation	
101	Observation-1	202	4761	has as performer	103	Fred Bloggs	
101	Observation-1	203	4763	uses as tool	104	150T-001	
101	Observation-1	204	5058	is observing aspect	105	Temp of B-1	
105	Temp of B-1	205	5020	is qualified as	589197	hot	2007-03-15T16:00

The bearing and its temperature can be defined as follows:

102	Bearing-1	206	1225	is classified as a	131770	roller bearing
102	Bearing-1	207	1727	has as aspect	105	Temp of B-1
105	Temp of B-1	208	1225	is classified as a	550415	operating temperature

Example 2: In addition to or instead of qualification of the temperature in Example 1 as ‘hot’, the observing activity can also measure and *quantify* the temperature. This requires other relation types and a unit of measure (UoM). For example, the quantification of the temperature at two successive minutes can be recorded as follows:

105	Temp of B-1	109	5025	has on scale a value equal to	920376	80	degC	2007-03-15T16:00
105	Temp of B-1	110	5025	has on scale a value equal to	920381	90	degC	2007-03-15T16:01

8.2.2.1 Time variation of measured values

A property is defined in a formal language as a characteristic which value may vary in time and space. So, even when the value varies, the values are quantifications of the same single property. Therefore, the table above contains only one property ‘Temp of B-1’, also when the value (magnitude) of that temperature is quantified multiple times at different moments. A property, such as that temperature, may also vary in space, because the temperature may be different on different locations in the bearing. In this particular example the possessor of ‘Temp of B-1’ is the (whole) bearing, so that it may be concluded that the whole bearing is assumed to have that temperature. For a description of the recording of the variation of property values in space, see paragraph 8.4.

Example 3: Consider the transmitted power as a function of time for "Line-2", as is shown in Figure 55.

Figure 55, Transmitted power as a function of time

The following statements can be made:

- the function from 10:00 to 11:00 has a mean value of 10000 kW;
- the function from 11:00 to 12:00 has a mean value of 15000 kW;
- the function from 10:00 to 12:00 has as mean value 12500 kW.

Thus, the *mean* power transmitted through Line-2 from 2007-03-14T10:00 to 2007-03-14T12:00 is 12500 kW as measured by Measuring device-2. This measurement is recorded as follows:

11 1	Observation -2	211	1225	is classified as a	19341 7	measuremen t			
11 1	Observation -2	212	4763	uses as tool	114	Device-2			
11 1	Observation -2	213	5058	is observing aspect	115	Power in Line-2			
11 5	Power in Line-2	214	1225	is classified as a	55302 0	electric power			
11 5	Power in Line-2	215	5712	has on scale a mean value equal to	92534 4	12500	kW	2007-03- 14T10:0 0	2007-03- 14T12:0 0

The two moments in time, being the start date-time of the validity of the fact and the end date-time of the validity of the fact, represent contextual facts that can be recorded in columns 9 and 10 in the Gellish Expression Format (for details about stored value and date-time representation see the document ‘Gellish Syntax’ [Ref. 4]). Thus the above expression does not specify the variation of the value during the validity period. Such a variation can be described by splitting the period in smaller periods (discretization of the period).

If a property is described by boundary values, such as a property that is within a range, or greater than or less than a particular value, then other relation types should be used. For example, instead of relation type 5712 <has on scale a mean value equal to> it is possible to use one of the following relation types that map properties to numbers:

5025 has on scale a value equal to

5468 has on scale a value within numeric range

5027 has on scale a value less than
 5490 has on scale a value less than or equal to
 (synonym: has on scale a maximum value of)
 5489 has on scale a value greater than
 5026 has on scale a value greater than or equal to
 (synonym: has on scale a minimum value of)

Somewhere else in the database it should be specified that:

112 Line-2 1727 has as aspect 115 Power in Line-2
 114 Measuring device-2 1225 is classified as a 71061 power meter

8.2.2.2 Accuracy of a measurement (tolerances)

A measurement is never exact. So a measured value always has some uncertainty. This uncertainty can be described by a range around a pivot value. For example, if a measurement results in the fact that diameter D1 <has on scale a value equal to> 30 mm, then the value 30 is the pivot value, whereas it might be specified that the lower tolerance is -0.3 mm and the upper tolerance is +0.4 mm (thus the real value is greater than 29.7 mm and less than 30.4 mm). So, the range is -0.3;+0.4 mm. The range shall be defined in the applicable domain dictionary either as a qualification of mathematical range (an absolute value) or as a qualification of a ratio. In the latter case the upper and lower tolerances shall be interpreted as fractions of the pivot value. The tolerances can be recorded in a formal language as follows (see the ‘Gellish Syntax’, columns 76 and 77: UID and name of accuracy of mapping):

16 D-1 5025 has on scale a value equal to 920283 30 mm -0.3;+0.4

In engineering in the metric system the pivot value has by default a tolerance or accuracy of plus or minus the half of the last digit of a decimal representation of the pivot value. For example, the value 30 implies a default accuracy range of -0.5;+0.5, whereas the default accuracy range of the value 30.0 is -0.05;+0.05.

During design, a tolerance for the allowed deviation from a specified (pivot) value can be recorded in the same way.

8.2.3 An observation device

The description of an observation device can be relevant for the information about an observation and for the functioning of an instrument loop by which a process is controlled.

An observation device can be a measuring instrument, a gauge or just a recorder, such as a video camera with or without sound registration. Therefore, the observation device shall be classified, for example as follows:

22	115F-001	1225	is classified as a	70073	coriolis flow meter
26	115F-002	1225	is classified as a	70499	dall tube flow meter

A measuring instrument (70254, also called a meter) has some or all of the following parts:

- sensing element (70912, also called a sensor or probe)
This responds to a physical phenomenon and produces a signal that depends upon the presence or the magnitude of the phenomenon.
A measuring instrument contains at least one sensing element.
- transmitter (70700)
This receives a signal from a sensing element and sends a signal with the same information to a remote receiver.
A measuring instrument contains at least one transmitter.
- indicator (70188)
This receives a signal and presents the information to a person close by.
A measuring instrument may contain an indicator.

Note: A sensor and indicator without a transmitter is called gauge (70541), which is an instrument, but is not a *measuring* instrument.
- information processor (70633)
This receives input information and creates output information according to an algorithm.
- recorder (70352)
This receives an input signal and records values on an information carrier or medium.

This knowledge can be summarized in a formal language as follows:

70254 measuring instrument	5519 has by definition as part a	70912 sensing element
70254 measuring instrument	5519 has by definition as part a	70700 transmitter
70254 measuring instrument	1191 can have as part a	70188 indicator
70254 measuring instrument	1191 can have as part a	70633 processor
70254 measuring instrument	1191 can have as part a	70352 recorder

An individual measuring device can be part of an instrument loop, together with other parts, such as a control valve. The specification of such possibilities results in a composition hierarchy.

For example, Unit U-11500 contains a flow instrument loop 1, which has as parts among others a flow meter and a control valve, whereas the flow meter has as parts a flow sensor and a transmitter.

This can be represented in a composition hierarchy as follows:

U-11500	process unit
115FICA-001	instrument loop
115F-001	coriolis flow meter
115FE-001	flow sensor
115FT-001	transmitter
115FCV-001	control valve

The classification and composition can be specified in a computer interpretable way in a formal language as follows:

20	U-11500	121	1225	is classified as a	160104
	unit				
20	U-11500	122	1190	has as part	21
21	115FICA-001	123	1225	is classified as a	70239
	loop				
21	115FICA-001	124	1190	has as part	22
22	115F-001	125	1225	is classified as a	70073
	flow meter				
22	115F-001	126	1190	has as part	23
23	115FE-001	127	1225	is classified as a	70530
22	115F-001	128	1190	has as part	24
24	115FT-001	129	1225	is classified as a	70700
21	115FICA-001	130	1190	has as part	25
25	115FCV-001	131	1225	is classified as a	820020
	valve				

8.3 Recording of textual or pictorial results

An activity can create information. The information that is created is then encoded on an information carrier that is an output from the activity.

The information can have several forms, such as:

- (1) A collection of facts that are expressed as a collection of formal language expressions, as is illustrated in paragraph 1.1.

- (2) A textual description that is encoded on an information carrier, of which multiple copies in various formats may exist.

8.3.1 Collections of formal language expressions

An activity can have as result information that is expressed as a collection of computer interpretable expressions.

For example: Consider the above described human observation activity example. The resulting collection of facts (ObsResults-1) consists of facts 101 through 109. The facts can be explicitly included in the collection by an inclusion relation for each fact.

The definition of the collection and its relation to the activity is recorded as follows:

101	Observing act-1	4786	has as result	106	ObsResults-1
106	ObsResults-1	1225	is classified as a	970178	collection of facts

Note: The information carrier for these expressions is the physical database or file that contains the formal language expressions. The relation between the expressions in a database and the database itself, including the location of that database, is usually not explicitly modeled. But it can be done in a similar way as is described in the next paragraph for the location of information.

8.3.2 Information on information carriers

An activity can also have as result information that is expressed as a piece of text, sound, pictures or video. Such information can vary from one or a few words to a whole story and a complete video with a combination of pictures and sound. The information can be provided in two ways: (1) directly as text string in the database or (2) it can be provided indirectly by being encoded on a carrier, such as one or more sheets of paper or in digital form in a computer memory (file or database).

When the information is provided directly, then the information (text string) has a UID and a name (for example: 31 ObsResults-1), whereas that text should be provided in the “description” column in an Expression Table as follows:

1	Observing act-1	4786	has as result	31	ObsResults-2
31	ObsResults-2	1225	is classified as a	910171	description
<i>“cooler-1 is located in...”</i>					

In the case that the information is provided indirectly be a reference to an information carrier, then the information carrier is modeled as a separate

object (say ‘Carrier-3.pdf’, or ‘Carrier-4.wav’, being an electronic data file), whereas the location (address) of the file should be recorded. This way of modeling results in the following expressions:

1	Observing act-1	4786	has as result	32	ObsResults-3
32	ObsResults-3	1225	is classified as a	910171	description
32	ObsResults-3	4996	is presented on	33	Carrier-3.pdf
33	Carrier-3.pdf	1225	is classified as a	490533	electronic data file
33	Carrier-3.pdf	1227	is an element of	34	
http://www.example.com/					

Note: If the information carrier itself is an output of an activity, then the activity is a document production process. In that case the relation type <has as output> should be used, instead of <has as result>.

8.3.3 Relations between resulting information and described objects

Information can be described in natural language as human interpretable text and stored as such in a computer, or it can be modeled in a computer interpretable way. One of the advantages of modeling in a computer interpretable way is that the objects about which an expression provides information and the expression itself are separate objects each of which can be used to make relations with other objects and both can be used for search and retrieval. This difference between free text and modeled information can best be illustrated on an example where there is a choice between recording the information about an observation as a piece of text (a string of characters), and recording the same information as a computer interpretable expression. This is illustrated by the following example.

For example:
Assume that an observation results in fact 130, which is that "cooler-1 is located in building-1". That information can be regarded as information about cooler-1, but also as information about building-1. When this fact is recorded as a piece of text, then the fact that the information is about those two objects can only be recorded in a computer interpretable way by explicitly modeling of the relation between the text and those two objects. This depends further on the question whether the piece of text is provided directly in the description field, together with the classification of the results, or whether it is provided as the content of an information carrier.

Modeling the text directly in the description field is done as follows:

31 ObsResults-2 1225 is classified as a 910171 description “cooler-1 is located in...”

In that case the reference between the described objects and the piece of text is modeled as follows:

35 Cooler-1 1273 is described by information 31 ObsResults-2
36 Building-1 1273 is described by information 31 ObsResults-2

When the piece of text is recorded on a separate document, then the relations are not made directly to the information, but the relations are made to the information carrier on which the information is encoded. In that case the relation between the described objects and the information carrier can be described as follows:

35 Cooler-1 4720 is described in information carrier 33 Carrier-3.pdf
36 Building-1 4720 is described in information carrier 33 Carrier-3.pdf

However, the alternative method to model the fact explicitly as a computer interpretable expression is simpler and enables that the computer can search for all objects with the same kind of relation with one of the related objects. For example a search for all objects that are located in Building-1 can be formally expressed as follows:

135 Cooler-1 130 5138 is located in 136 Building-1

In the latter case the relations between the objects and the fact are explicit and more precise because of the computer interpretability of the expression.

8.4 Measurement of location dependent properties

A location dependent property is modeled as a property of a physical point, being a negligible small physical space, which is part of the measured physical object. The location of such a physical point can be defined in various ways. For example:

1. Implicitly by giving the physical point a name that consists of a concatenation of the measured physical object and the measuring instrument or the measuring location.
Example 1: the temperature T1 of stream S1 at the location of 160T-001. The physical point is a part of S1 and is called ‘S1 at 160T-001’. This is specified as follows:

S1	has as part	S1 at 160T-001
S1 at 160T-001	has as aspect	T1

Note 1: It is not practical to create a direct relation between the instrument (160T-001) and the measured aspect (T1), as there is already a relation between them via the measuring activity.

Note 2: The part of a flowing fluid stream is in this case determined by the space through which molecules flow. Thus the measurement over time measures a property of a variable mass.

Example 2: the height H1 of the river Rhine at Koln. The physical point is a part of the Rhine and is called ‘Rhine at Koln’. This is specified as follows:

Rhine	has as part	Rhine at Koln
Rhine at Koln	has as aspect	H1

In the latter case the relation between the physical point and the location can be modeled explicitly as follows:

Rhine at Koln	is located in	Koln

2. Explicitly by specifying the coordinates of the point in the same coordinate system in which also the measured object is described

For an example, a hot spot H1 is a part of vessel V1 is located at the physical point P1, where the temperature T1 is measured. The vessel is described in a cylindrical coordinate system C1. The coordinates (r, phi, z) of the hot spot P1 are located at a radius of 900 mm, an angle of 0.35 radian and a height of 300 mm. This is modeled as follows:

P1 in hot spot	is a part of	V1
P1 in hot spot	is classified as a	physical point
P1 in hot spot	has as aspect	T1
T1	is classified as a	temperature
P1 in hot spot	has as coordinates point	C1
C1	is classified as a	cylindrical coordinate point

C1	has on scale as coordinate values	900, 0.35, 300
		mm, rad, mm
C1	is a coordinates point in	CS1
CS1	is classified as a	cylindrical coordinate system

A cylindrical coordinate system is defined as follows:

cylindrical coordinate system	has by definition as first dimension a	radius
cylindrical coordinate system	has by definition as second dimension a	angle
cylindrical coordinate system	has by definition as third dimension a	height

9 Integration of 1D, 2D and 3D product models

This section discusses modeling of two-dimensional (2D) graphics and drawings and three-dimensional representations of shape (3D models) as well as the integration of 2D graphics and 3D shape information with non-graphical (1D) information.

9.1 Levels of information integration

There are several levels at which 2D and 3D data can be integrated with 1D information. First, data about objects can be integrated with data about their positioning in 3D (position and orientation) in the real world. For example, Figure 56 illustrates information that specifies a position of an object relative to a point O_1 in the real world.

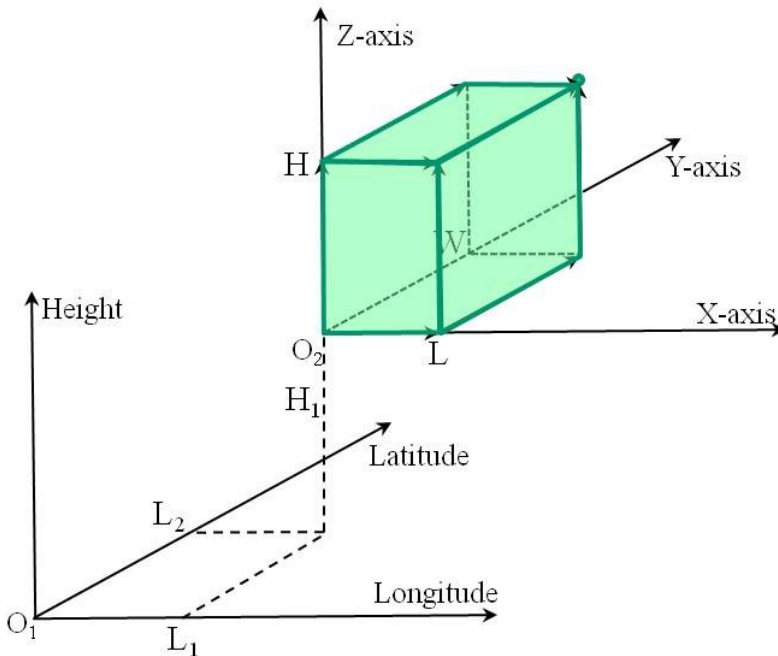


Figure 56, Positioning in a (real world) coordinate system

The modeling of such a position requires that (a point O_2 on) the object is positioned in a reference coordinate system and a specification of the distances in the directions of the coordinates. In a similar way the rotation

angles with the three planes of the coordinate system specify the orientation of the object. Such information about an object specifies a position and orientation in 3D, although the various coordinate values are just ordinary 1D values. Therefore, the information can be directly integrated with other information about the object.

The point O_2 is the origin of a second coordinate system in which the shape of the object can be described and it is a point that is used for positioning the object relative to the reference point O_1 . Such a point is called a datum. A datum on an object is a physical point that typically but not necessarily coincides with the origin of the coordinate system in which the shape of the object is defined. The precise modeling of this is further described in par. 9.3.6 Positioning in 3D coordinate systems.

Data about objects can also be integrated with data about documents and with graphical representations of the objects on documents or screens. This can be done on various levels of detail as is illustrated in Figure 57.

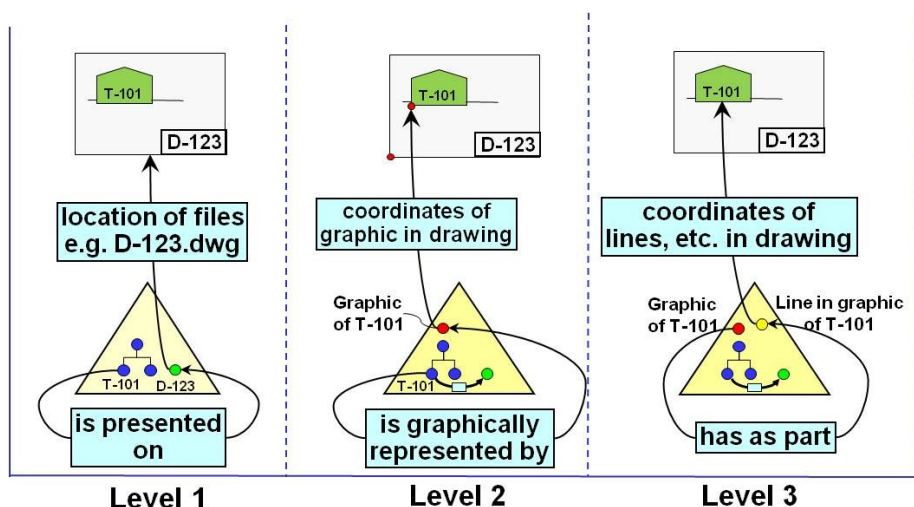


Figure 57, Levels of integration of drawing data in product models

The lower part of Figure 57 illustrates product models that are integrated at various levels of integration with (data on) documents and drawings or views that is represented in the upper part of the figure.

- At level 1, the product model includes relations between the objects and the files that contain documents and drawings as a whole. For example as a reference to a pdf or dwg file.

- At level 2, the product model includes relations between the objects and graphical representations of those objects. Those graphical presentations can be schematic symbols or drawings to scale. The model also includes the coordinates of the graphics in the coordinate system of the drawings or the coordinates of text that accompanies the graphics (such as tag names).
- At level 3, the product model also includes relations that express how the graphical representation is constructed. For example, including decompositions into lines and surfaces or geometric models of shapes as wire frames or as solid models. This also includes the coordinates of those graphics on the drawings.

The content of the files at level 1 may be in any format and is not integrated with the product model, apart from the fact that it is recorded that the document or drawing contains information about a particular object. Thus, at level 1, the product model contains relations between objects and electronic data files. For example, a model may contain the relation that expresses that tank T-101 is presented on a drawing D-123. That drawing is stored in a file in a particular format, for example as D-123.dwg. That drawing may also be stored in another file format, for example as D-123.pdf. This can be expressed in the product model as follows:

T-101	is presented on	D-123
T-101	is classified as a	tank
D-123	is classified as a	drawing
D-123	is presented on	D-123.dwg
D-123.dwg	is classified as a	AutoCAD file
D-123	is presented on	D-123.pdf
D-123.pdf	is classified as a	Acrobat file

The product model should also contain the locations of the files (e.g. their URIs), so that application software can find the files, and by using the file types (e.g. dwg) or the classification of the files (e.g. as AutoCAD file) software can launch an applicable application system (viewer) and display the drawings and other documents.

At this first level of integration the various files with documents and drawings will however contain information that partly overlaps with data that is also available in other documents and databases. The consistency of such duplicate information is not guaranteed by the model, because the content of the documents and drawings is not modeled explicitly. Thus the consistency needs to be managed manually. Data integration on this first level is described in par. 5.3.

Level 2 and level 3 include the integration of product data with graphical information in 2D drawings and views and with 3D geometric models that represent 3D shapes. Those higher levels of data integration enable more automated management of the consistency of the data. The higher levels of integration require that graphical and shape information is modeled in a common formal language, using a consistent methodology. The methodology implies among others that the unique identification of all objects and facts, including also graphical objects and shape representations is managed through a systematic change management process. When such a methodology is applied, it becomes possible to create an integrated 1D, 2D and 3D product model, although it is still possible that the various data sets are managed independently by different parties and nevertheless form integral sections of one consistent product model.

Thus, an integrated 1D, 2D and 3D product model is a model that is stored in one or more databases or files and that integrates conventional product data with data about the position and orientation of the objects in a coordinate system that may be positioned in the real world (on a datum) and with data about graphical objects on drawings and with geometric models that represent shapes (2D views and 3D models).

One dimensional (1D) data about products consists of the aspects that can be qualified by single values, such as a diameter, length, color, material of construction, strengths, temperature and pressure. 1D data also include relations between objects, such as assembly relations, connection relations with other products and involvements in processes and activities, etc. 2D data is information that can be expressed graphically in a 2D drawing. It includes shape and position information in drawings to scale, such as top views, side views and cross sections, but also schematic diagrams that use symbolic representations and that are not to scale. A product model in 3D is a model of a physical object that includes a specification of its shape in three dimensions. When it is a part of a bigger assembly, such as a building, a transport system, a ship or a process plant, then the model also includes its position and orientation in a 3D space. The 2D and 3D shape information in various drawings and models should be kept mutually consistent and should also be kept consistent with the 1D size information. This is a strong argument for their integration.

2D and 3D product models are typically created by using software packages of various suppliers. The resulting models are typically stored in the proprietary formats of those packages. For example, in DWG, DXF, CMG, DGN or RVT format. Each of such formats has its own limitations on the type of data that can be stored. Data that is stored in one format can only be

converted to another format when a single or bi-directional converter is created and when the receiving format is suitable for storing all kinds of data that is to be converted.

Non-2d and non-3D information about a product is typically stored in other proprietary systems with their own proprietary formats or data structures. For example the structure of ERP databases or PLM databases. Each of such systems has a different structure of its data and also uses different definitions for the concepts that it applies. Therefore, exchange of data between systems and integration of data of different sources raises all sorts of data conversion issues.

9.2 Characteristics of 1D, 2D and 3D product models

An integrated product model typically contains one dimensional (1D), two dimensional (2D) and three dimensional (3D) aspects of physical objects. A product model may also contain higher dimensional aspects such as shape variations as a function of time. Furthermore, a model may be integrated with documents and data sets in proprietary file formats, such as doc, dwg, ppt, pdf or xls formats.

1D aspects are defined by the fact that the magnitudes of 1D aspects can be quantified by single numbers (scalars) on scales or by single qualitative qualifications. For example, a diameter can be quantified by a single number, such as 5, on a scale, such as mm. Flatness can be qualified as ‘flat’ and a temperature as ‘hot’.

2D aspects typically describe positions in a two dimensional flat plane or points on a curve. The magnitude of these kinds of aspects is normally described by two values each on its own scale. An example of a 2D aspect is the location of building B1 within an area A1. The position can be described by two value (200 m, 130 m) in respectively the x and y direction from the origin of a Cartesian coordinate system of area A1. Another example is the operating curve of a pump, which can be described by a differential pressure as a function of flow rate.

A 3D spatial shape can be described in the form of a geometric model (which is mathematical model) in its own coordinate system, being a coordinate system in which the object is in rest. Such a geometric model requires quantifying the shape in three dimensions, being the three directions in the coordinate system. This can be done by composing complex geometric models from basic geometric models of shapes. Examples of basic geometric models of basic shapes are points, lines, surfaces and volumes, such as cylinders, blocks, spheres, etc. Each of those basic geometric concepts can be

described by a mathematical function (formula) in a coordinate system. The complete shape of an object is modeled by specifying the functions that compose the complete model and by specifying the positions and the orientations of those geometric functions in one 3D coordinate system.

4D models are for example models of 3D shapes that vary in time, or the operating space of a centrifugal pump, where the flow rate and differential pressure are dependent on the speed of rotation and the diameter of the impeller. These higher order dimensional aspects are outside the current scope of this document.

9.3 Coordinate systems specification

The 2D or 3D shape of an object or symbol is normally defined with respect to the 'own' 2D or 3D coordinate system of the described object or symbol. Those shapes may then be positioned in the coordinate system of a bigger assembly, which on its turn may be positioned in again another coordinate system, such as the geographic coordinate system of the earth. When a view or drawing is created, then the shape is positioned in the coordinate system of the drawing, within the drawing area. The ratio between the size of the graphical presentation of an object in a drawing and the size of the object in the real world is called the scale factor of the drawing. Examples of scale factors are: a scale factor of 1:1 for a small object or a scale factor of 1:100.000 for distances on a geographic drawing.

Before coordinate points can be defined it is necessary to first define its coordinate system by specifying its dimensions.

A coordinate system is a property space with multiple dimensions. Thus a 3D coordinate system is a property space with three dimensions, typically but not necessarily distances. An individual coordinate system is defined by a definition of its dimensions. The pattern for a specification of the dimensions is illustrated in Figure 58.

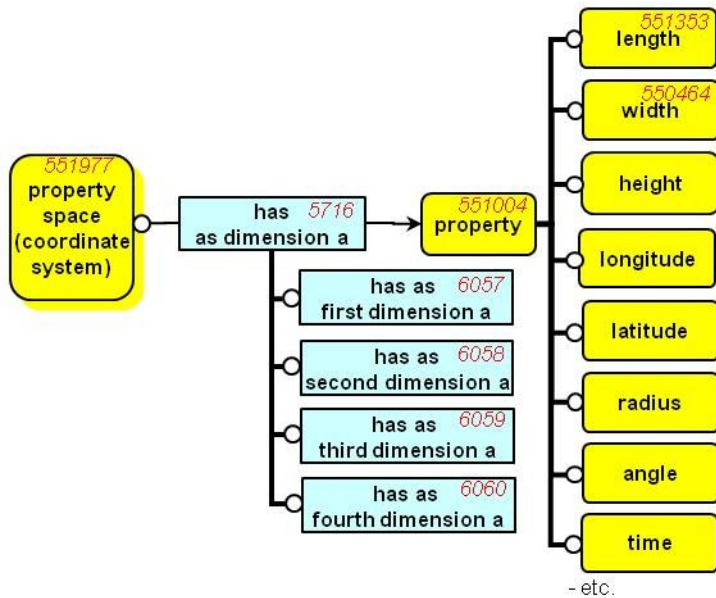


Figure 58, *Specification of the dimensions of a coordinate system*

A coordinate system that is possessed by an individual object and in which that object is in rest is an example of an individual coordinate system. In other words, when the object is moved, then that coordinate system moves with it, so that the coordinates in which the shape of the object is described do not change, unless the shape of the object changes over time. The relation between a physical object and its possessed coordinate system is illustrated in Figure 59.

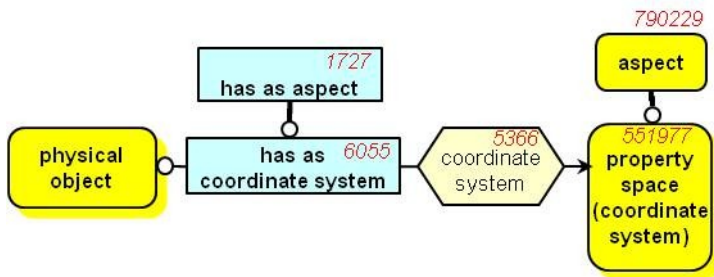


Figure 59, *Possessed coordinate system*

The relative position of objects in relation to each other is described by the position and orientation of objects relative to a possessed coordinate system. However, the shape of an object should be described in its own coordinate system.

Each coordinate is thus a property. Particular properties can be quantified on their own scale. Those scales may be different or may be the same for all dimensions. For example, we can define a particular 3D coordinate system that has three dimensions that are three distances, (typically a length a width and a height) whereas a coordinate in each dimension can be quantified on a length scale, such as mm.

Note that in mathematics a coordinate system typically has no scales, but in physics and technology coordinates are quantified on scales.

Thus, normally each coordinate is quantified by a number on a scale. However, it may be specified that the scale of any coordinate in the same dimension (direction) is by default the same for each value in that direction in that coordinate system. For example, if it is specified that x-coordinates are quantified by default in dm3/s, then every point in that coordinate system has an x-coordinate value with by default that same scale (unit of measure). This means that the scale can be defined once for each dimension and then it is not required to specify the scale for each value in the coordinate system. A default scale can be specified together with the specification of the associated dimension. Thus a pattern for the definition of a coordinate system is as follows:

Ph.object-1	has as coordinate system	Coord.System-1	
Coord.System-1	has as first dimension a	length	mm
Coord.System-1	has as second dimension a	width	mm

There are several kinds of coordinate systems. For example:

- Cartesian coordinate systems (2D and 3D)
- Cylindrical coordinate systems
- Spherical coordinate systems
- Geographic coordinate systems
- Parametric coordinate systems

The formal language taxonomic dictionary contains definitions of the first four kinds of concepts, but anybody can add his own concepts or propose to add concepts to the standard language definition. The next paragraphs describe how coordinate systems of these kinds are defined in the formal language.

9.3.1 Cartesian coordinate systems

A coordinate system is typically a [Cartesian coordinate system](#). A Cartesian coordinate system is defined by two or three axes that point in directions that

are perpendicular to each other. The axes in a Cartesian coordinate system are normally called the x-axis, the y-axis and the z-axis. There are left handed and right handed 3D Cartesian coordinate systems. The z-axis in a right handed 3D Cartesian coordinate system has the direction that a corkscrew goes when turning from the x to the y-direction.

For a spatial coordinate system, each of the three dimensions is a distance. The formal dictionary defines that as follows:

924995	3D Cartesian coordinate system	4664	has by definition as dimension a	550212	distance in x-direction
924995	3D Cartesian coordinate system	4664	has by definition as dimension a	550213	distance in y-direction
924995	3D Cartesian coordinate system	4664	has by definition as dimension a	550214	distance in z-direction

A right handed 3D Cartesian coordinate system is illustrated in Figure 60.

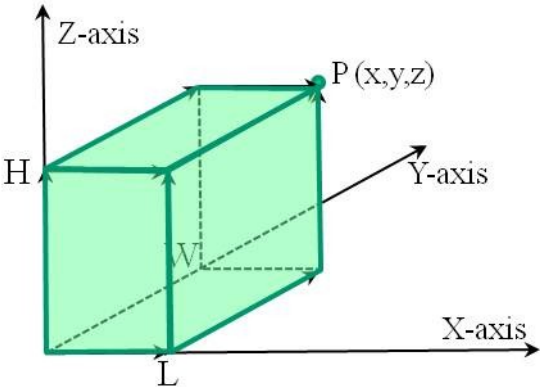


Figure 60, A right handed 3D Cartesian coordinate system

The coordinates of a point (P) in a particular (individual) coordinate system that is classified by this kind is by definition specified by three distances x, y and z, being three distances in the direction of the X-axis, Y-axis and the Z-axis. The three scales for the distances in the three directions are normally the same. Thus, if the scale for a particular 3D spatial Cartesian coordinate system is specified as mm, then the scale of the x-direction, the y-direction and the z-direction are all in mm. This can be specified in one go by specifying the scale for the whole system.

For example, if a physical object, such as the solid block in Figure 60, is defined with respect to a particular (individual) 3D coordinate system CoordSys-201, then that coordinate system shall be defined by a

classification and implicit or explicit definition of the scales of its dimensions as follows:

10 1	CoordSys-201	1225	is classified as a	924989	right handed 3D Cartesian coordinate system	
10 1	CoordSys-201	6057	has as first dimension a	550212	distance in x- direction	mm
10 1	CoordSys-201	6058	has as second dimension a	550213	distance in y- direction	mm
10 1	CoordSys-201	6059	has as third dimension a	550214	distance in z- direction	mm

The representation of a shape (or space) can be implemented in various ways, such as a solid model, a surfaces model or a wire frame model. Each implementation is based on one or more geometric formulae that are defined in one of the possible coordinate systems. When a shape is described in a Cartesian coordinate system, then that shape is essentially defined by one or more mathematical functions (algorithms or formulae) in three dimensions, abbreviated as x, y and z, which function has the sizes of the shape as its parameters.

The various possible implementations mean that a shape can be visualized (presented) through the use of various different computer executable programs, for example in the form of Macros, written in some computer interpretable programming language.

The modeling of shapes is further clarified in the next chapter.

9.3.2 Cylindrical coordinate systems

A [cylindrical coordinate system](#) has by definition three coordinate axes, as is illustrated in Figure 61.

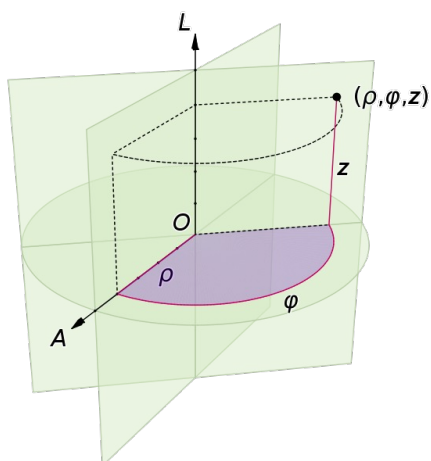


Figure 61, A cylindrical coordinate system

The coordinates are by definition: a distance or radius from the centre (ρ or ρ ho), an angle (φ or ϕ i, also called the azimuth angle) relative to a reference direction and a z-axis that is perpendicular to a chosen plane in which the radius and the angle are defined. The positive z-axis is in the right hand corkscrew direction. If the chosen plane is assumed to be horizontal, then the z-axis is called the height. This is defined in the dictionary as follows:

928237	cylindrical coordinate system	4664	has by definition as dimension a	550681	radius
928237	cylindrical coordinate system	4664	has by definition as dimension a	550029	angle
928237	cylindrical coordinate system	4664	has by definition as dimension a	550126	height

For example, a particular 3D cylindrical coordinate system CoordSys-301 is defined as follows:

10 2	CoordSys-301	1225	is classified as a	928237	cylindrical coordinate system	
10 2	CoordSys-301	6057	has as first dimension a	550681	radius	mm
10 2	CoordSys-301	6058	has as second dimension a	550029	angle	deg
10 2	CoordSys-301	6059	has as third dimension a	550126	height	mm

9.3.3 Spherical coordinate systems

A spherical coordinate system is a 3D coordinate system as is illustrated in Figure 62.

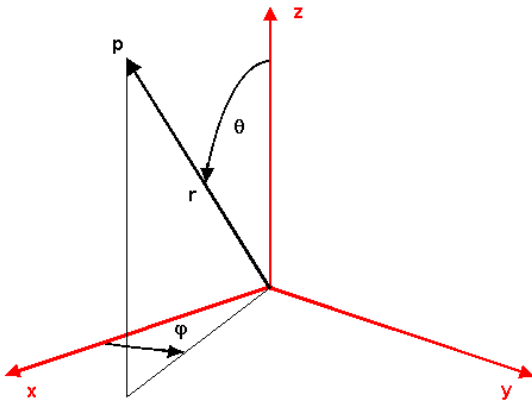


Figure 62, A spherical coordinate system

The position of a point (p) in a spherical coordinate system is specified by three coordinates (r, θ, ϕ): the radial distance of that point from a fixed origin (radius or r), its inclination angle (theta or θ) measured from a fixed zenith direction, and the azimuth angle (phi or ϕ) of its orthogonal projection on the x-y plane.

The second angle (the inclination angle) is often replaced by an elevation angle measured from the reference plane perpendicular to the zenith direction. Note that this is the same kind of spherical coordinate system, but the second parameter is classified as an elevation angle. An elevation angle is 90 degrees ($\pi/2$ radians) minus an inclination angle.

The radial distance is also called the radius or radial coordinate, and the inclination may be called colatitude, zenith angle, normal angle, or polar angle.

The sign of the azimuth is determined by choosing what is a positive sense of turning about the zenith. This choice is arbitrary, and is part of the coordinate system's definition: right handed or left handed.

A spherical coordinate system may be defined in the dictionary as follows:

92825 2	spherical coordinate system	466 4	has by definition as dimension a	55068 1	radius
92825 2	spherical coordinate system	466 4	has by definition as dimension a	55370 5	inclination angle
92825 2	spherical coordinate system	466 4	has by definition as dimension a	55370 5	elevation angle
92825 2	spherical coordinate system	466 4	has by definition as dimension a	55370 6	azimuth angle

For example, a particular 3D spherical coordinate system CoordSys-401 is defined as follows:

103	CoordSys-401	1225	is classified as a	928252	spherical coordinate system	
103	CoordSys-401	5716	has as dimension a	550681	radius	mm
103	CoordSys-401	5716	has as dimension a	553705	inclination angle	deg
103	CoordSys-401	5716	has as dimension a	553706	azimuth angle	deg

Note: If other choices are made for the definition of the angles, then they shall be classified by the otherwise defined kinds of angles.

9.3.4 Geographical coordinate systems

A [geographic coordinate system](#) is a coordinate system that is related to the earth. It enables every location on Earth to be specified in three coordinates, typically using a spherical coordinate system.

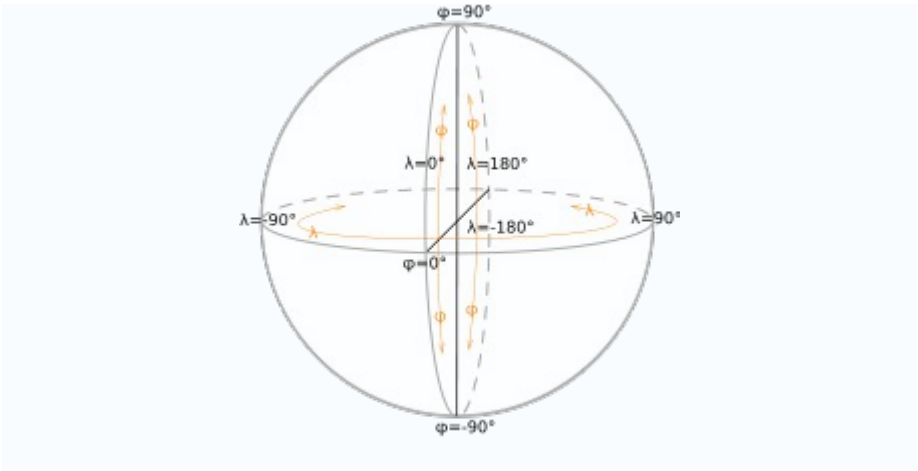


Figure 63, Latitude phi (ϕ) and Longitude lambda (λ) in a geographic coordinate system

In geography and astronomy, the elevation and azimuth (or quantities very close to them) are called the latitude and longitude, respectively; and the radial distance is usually replaced by an altitude or depth above or below the earth surface or a representation of it, such as the geoid.

Thus, a spherical geographic coordinate system may be defined in a formal language domain taxonomic dictionary as follows:

928252 spherical coordinate system 1446 is a specialization of 5366 coordinate system
 928252 spherical coordinate system 4664 has by definition as dimension a 553707 altitude
 928252 spherical coordinate system 4664 has by definition as dimension a 550095 depth
 928252 spherical coordinate system 4664 has by definition as dimension a 553060 latitude
 928252 spherical coordinate system 4664 has by definition as dimension a 553059 longitude

The altitude (with a positive value in upward direction) or depth is measured from a datum. Typically being the mean sea level (MSL) surface or above ground level. To specify which reference datum is used for a particular height, the altitude or depth shall be classified by one of the subtypes of altitude or depth, such as a true altitude (means sea level altitude) or an absolute altitude (ground altitude).

The radius relative to the centre of the earth can be specified either as an altitude or as a depth (with a positive value in downward direction), both with reference to a datum. Furthermore, the depth can be specified in various different ways, especially as true vertical depth (TVD) or as measured depth (MD) along a path that is not necessarily vertical. Furthermore the reference datum can vary, so that a subtype of TVD or MD shall be selected for classification of an individual length. For example as true vertical depth

below ground level (TVDGL). For example, the radius of an individual point that is 300 m below ground level can be specified by two equivalent but different properties, one in upward direction, and the other with a positive value in downward direction. The two equivalent ways are:

107	H1	1225	is classified as a	553708	true altitude	
107	H1	5025	has on scale a value equal to	1000000300	-300	m

and

108	H2	1225	is classified as a	553713	TVDGL	
108	H2	5025	has on scale a value equal to	920466	300	m

Note that $H1 = -H2$.

9.3.5 Standard individual coordinate systems

9.3.5.1 RD coordinate system

The RD coordinate system, an abbreviation of [Rijksdriehoekskoördinaten-systeem](#), is an individual 2D Cartesian coordinate system for the Netherlands that is based on the European Terrestrial Reference System (ETRS89) as a reference coordinate system (see the next paragraph). The first dimension (x-coordinate) is a distance measured from west to east, the second dimension (y-coordinate) is measured from south to north. The origin (0,0) of the RD coordinate system is defined at a point 155 km west and 463 km south of the point of the main tower in Amersfoort, called the ‘Onze Lieve Vrouwentoren’. The RD coordinate system is included in the dictionary of the formal language with UID 928254. Thus it does not need to be defined by users. It can be used by specifying coordinates points in that system. This can be done for example as follows:

103	Point-123	1876	is defined in coordinate system	928254	RD coordinate system

A Dutch national altitude as a third dimension is a height relative to NAP. The NAP (‘Normaal Amsterdams Peil’) is a reference height that slightly differs from the geoid. The European Vertical Reference System (EVRS) uses the NAP as its reference surface.

9.3.5.2 European Terrestrial Reference System

The European Terrestrial Reference System 1989 (ETRS89) is a Earth-Centered, Earth-Fixed ([ECEF](#)) [geodetic Cartesian reference coordinate system](#), in which the [Eurasian Plate](#) as a whole is static. The [coordinates](#) and [maps](#) in [Europe](#) based on ETRS89 are not subject to change due to the [continental drift](#). It is included in the dictionary with UID 928255. The ETRS89 uses the NAP as its reference surface for altitude measurement.

9.3.5.3 World Geodetic System

The World Geodetic System ([WGS84](#)) is a particular worldwide reference coordinate system that is defined for Global Positioning (GPS). It comprises a standard [coordinate frame](#) for the [Earth](#), a standard [spheroidal](#) reference surface (the *datum* or [reference ellipsoid](#)) for raw [altitude](#) data, and a [gravitational equipotential surface](#) (the *geoid*) that defines the *nominal sea level*. The coordinate origin of WGS 84 is meant to be located at the Earth's [center of mass](#); the error is believed to be less than 2 cm. http://en.wikipedia.org/wiki/WGS_84_-_cite_note-0

In WGS 84, the meridian of zero longitude is the [IERS Reference Meridian](#). http://en.wikipedia.org/wiki/WGS_84_-_cite_note-Manual-1 It lies 5.31 [arc seconds](#) east of the [Greenwich Prime Meridian](#), which corresponds to 102.5 meters (336.3 feet) at the latitude of the [Royal Observatory](#).

The World Geodetic coordinate system can be defined in the formal language as follows:

101	WGS84	1225	is classified as a	928252	spherical coordinate system

9.3.6 Positioning in 3D coordinate systems

Positioning of a physical object relative to another physical object is only specified precisely when it is known how a particular coordinate system of the positioned object is positioned in a particular coordinate system of the reference object. This is especially the case when there are several coordinate systems defined for the same object, such as is the case with the coordinate systems of the earth. Therefore, positioning of objects should in fact be expressed as positioning of a coordinate system in a reference coordinate system. For example:

Positioning of A to B	1225	is classified as a	positioning in a 3D space
Positioning of A to B	5722	has as reference	Coordinate system of A

		object	
Positioning of A to B	5825	has as positioned object	Coordinate system of B

In many cases the shape of objects is defined in a particular coordinate system, called ‘its own’ coordinate system, or formally, a coordinate system in which the object is in rest. This means that the ‘own’ coordinate system moves with the object, when the object moves. In a context where a positioned object and the reference object each has only one unambiguous coordinate system in which its shape is defined, then we can define a positioning relation as a relation between the physical objects, whereas it is implicit that the positioning is in fact between the coordinate systems in which the shapes of the objects are defined. Then the positioning relations becomes as follows:

Positioning of A to B	1225	is classified as a	positioning in a 3D coordinate system
Positioning of A to B	5722	has as reference object	Physical object A
Positioning of A to B	5825	has as positioned object	Physical object B

The relation between the two specifications is expressed as follows:

Physical object A	1727	has as aspect	Coordinate system of A
Physical object B	1727	has as aspect	Coordinate system of B

This is a short way of expressing that the physical objects have shapes that are modeled by geometric models that consist of correlations, whereas those geometric models (and thus those correlations) are defined in coordinate systems of the physical objects.

The reference object is usually called a datum, which is a physical object that acts as the origin of a reference coordinate system. A datum may be a large physical object, but typically it is a physical point. When a datum is a physical point on a larger physical object, then it shall be defined as a part of (or a feature of) the larger physical object. The location of the point relative to the larger physical object may be described as text in a non-computer interpretable form providing a definition of the point (provided together with its classification), or it may be specified (as well) by coordinates in the coordinate system in which the shape of the larger object is defined. For example:

Datum point-1	1190	is a part of	OLV tower	
Datum point-1	1225	is classified as a	physical point	located at the top of the OLV tower

				of Amersfoort

Thus, the position and orientation of an object can be specified by the location and rotation angle(s) of its own coordinate system with respect to a reference coordinate system.

For example,

Assume that building B-101 has a shape that is described by a geometric model, called Model-B-101. That model is defined in its own (individual) Cartesian coordinate system CoordSysB-101.

The shape, geometric model and coordinate system relate as is illustrated in Figure 64.

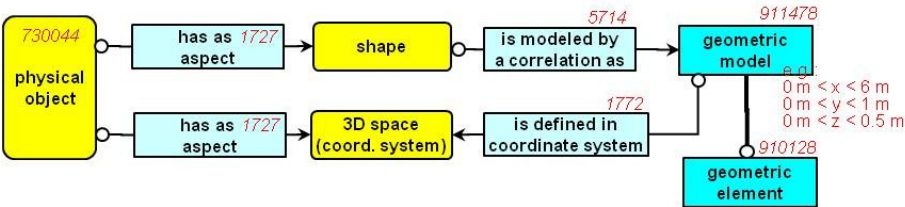


Figure 64, Relation between shape, geometric model and coordinate system

These relations can be specified as follows:

102	B-101	1727	has as aspect	103	Shape-B-101
103	Shape-B-101	1225	is classified as a	55025 6	shape
103	Shape-B-101	5714	is modeled by a correlation as	104	Model-B-101
104	Model-B-101	1225	is classified as a	91147 8	geometric model
104	Model-B-101	1772	is defined in coordinate system	105	CoordSysB- 101
105	CoordSysB- 101	1225	is classified as a	92498 9	right handed 3D CCSsystem

The table above illustrates that the positioned coordinate system CoordSysB-101 is classified as a right handed Cartesian coordinate system, abbreviated as right handed CCSsystem. Note that this may be a different *kind* of coordinate system as the kind of the reference coordinate system.

Furthermore,

Assume that coordinate system CoordSysB-101 is positioned for example

relative to the World Geodetic System WGS84 on a particular longitude and latitude and in a particular orientation.

This positioning can be expressed as follows:

106	Positioning of MB in WGS	122 5	is classified as a	5718	positioning in a 3D space	
106	Positioning of MB in WGS	572 2	has as reference object	101	WGS84	
106	Positioning of MB in WGS	582 5	has as positioned object	105	CoordSysB-101	
106	Positioning of MB in WGS	534 8	has as parameter	107	L1 of B-101	
107	L1 of B-101	502 0	is qualified as	110	24° 28' 0"	
107	L1 of B-101	122 5	is classified as a	55305 9	longitude	
106	Positioning of MB in WGS	534 8	has as parameter	108	L2 of B-101	
108	L2 of B-101	502 0	is qualified as	111	54° 25' 0"	
108	L2 of B-101	122 5	is classified as a	55306 0	latitude	
106	Positioning of MB in WGS	534 8	has as parameter	109	O1 of B-101	
109	O1 of B-101	502 5	has on scale a value equal to	112	0, 0, 0	deg
109	0, 0, 0	122 5	is classified as a	55372 7	orientation in 3D	

The positioning is specified in the above table by using a ‘positioning in a 3D space’ function. This is done by creating a particular individual ‘Positioning of Model-B-101 in WGS83’ function (abbreviated as ‘Positioning of MB in WGS’), which function is classified as a ‘positioning in a 3D space’ function. Such a function is a correlation that is related to a reference object and a positioned object and up to four parameters, a latitude, a longitude and optionally an orientation and a height.

Notes:

1. The parameters specify properties that are valid in the coordinate system of the reference object.
2. The latitude and longitude are ‘qualified’ by values in the conventional way. This means that they are to be interpreted as

qualitative aspects, with ‘names’ in the form of text strings. They can also be expressed as a list of three values that are to be interpreted as values on a degree, minute and second scale. The latter notation for the longitude would become:

L1 of B-101	5025	has on scale a value equal to	113
24, 28, 0		deg,min,sec	

3. In the above specification the height (altitude) is not explicitly specified. For WGS84 the default altitude is ‘on the surface of the earth’.

9.4 Shape aspects of Physical Objects

Shape is a spatial aspect of a physical object. For example: an individual sphere has a shape (spatial aspect) that is classified as a spherical shape. In other words its shape is classified as spherical.

9.4.1 Shapes

Physical objects follow the rules of physics. They usually have mass and energy and shape. The term shape of an object has physical meaning. The shape does not exist independent of the physical object.

A shape of a physical object is defined in the formal language taxonomic dictionary as a (subtype of) property, which is a volume of space occupied by the object. The shape of a surface of a physical object is defined as the area of space that is occupied by the surface of the physical object. Note that such an area is an area in a three dimensional space.

Examples of shapes are: the shape of a cylindrical shell (of a vessel), or the shape of the surface of a cylindrical shell, or the shape of an annotation element, such as a symbol on a drawing.

Note that an annotation element consists of a physical substance on a carrier, e.g. ink on paper. Examples of annotation elements are: symbols, text and lines (stripes). Note, that lines have mass (the mass of the ink), width and thickness.

The concept hierarchy for a shell is:

shell	is a subtype of	solid item
solid item	is a subtype of	physical object

The concept hierarchy for a line is:

line (stripe) is a subtype of graphical element
graphical element is a subtype of physical object

A physical object can be decomposed into physical elements, such as cubicles, each of which has a shape. A surface of a physical object can also be decomposed into partial surfaces. For example, the surface of a cylindrical shell can be decomposed into an outer surface, an inner surface and two surfaces with a ring shape on both ends of the cylindrical shell.

The shape of a physical object is defined in the formal language as a spatial aspect which is possessed by the object. Therefore the concept 'shape' is defined in the library of standard kinds of spatial aspects. In other words 'shape' is an instance of kind of spatial aspect.

9.4.2 Kinds of shapes

Various standard kinds of shapes can defined as subtypes of shape. This is for example the case in the formal language domain dictionary 'mathematics and shapes'. Examples of kinds of shapes are: cylindrical shape, spherical shape, block shape, etc. Some of the kinds of shapes can be used to classify volumes as well as surfaces of volumes.

An example of the definition of the shape of a cylindrical shell is as follows:

shell-1 has as aspect shape of shell-1
shape of shell-1 is classified as a hollow cylindrical shape

This can be sufficient, because a hollow cylinder may be described by a solid model (such as a csg model) that is defined by standardized correlation with as parameters a radius with values between Ri en Re and a length L (as described in par 9.4.7).

However, it is also possible to decompose the shape and define its surfaces separately. This can be done as follows:

shell-1	has as part	inner surface-1
shell-1	has as part	outer surface-2
shell-1	has as part	front surface-3
shell-1	has as part	back surface-4
inner surface-1	has as aspect	shape-1
shape-1	is classified as a	cylindrical shape
outer surface-2	has as aspect	shape-2
shape-2	is classified as a	cylindrical shape
front surface-3	has as aspect	shape-3
shape-3	is classified as a	flat ring shape
back surface-4	has as aspect	shape-4

shape-4	is classified as a	flat ring shape

9.4.3 Representation of shapes - Geometric Models

Physical objects follow the rules of physics. *Mathematical objects*, on the other hand, follow different rules, those of mathematics. Mathematical objects don't have mass or energy. They are just information. Mathematical objects can be distinguished (subtyped) in algebraic objects and geometric objects. Geometric objects, such as cylinders, have a shape, which can define/represent the shape of physical objects, such as the outer surface of a shell.

Algebraic objects don't have shape, although sometimes they can be represented by geometric objects. For example, an equation might be represented by a curve. Geometric objects have shape, for example, a straight line, a curve, a parabola, a circle or a sphere. Note, that geometric (straight) lines don't have width, nor thickness, nor mass.

So, the concept hierarchy for a parabola is:

parabola	is a kind of	curve
curve	is a kind of	mathematical space
mathematical space	is a kind of	aspect

The mathematical spaces can usually be represented by mathematical correlations. Several correlations in a coordinate system that together describe a closed space form a geometric model. The components of a geometric model can be called geometric elements.

Shape aspects can be distinguished in single dimensional aspects (1D aspects), such as distance, diameter, length and thickness, two dimensional aspects, such as lines, curves, circles, surfaces and 2D coordinates in a flat plane, and three dimensional aspects, such as 3D shapes and 3D coordinates in a 3D coordinates space.

We will use two examples to illustrate how such shape aspects can be expressed:

1. The shape of a wall in a building, with an opening and a window.
2. The shape of a tank, with a bottom, a wall and a roof.

Example 1:

Wall-101 has as 1D aspects a length of 5000 mm, a height of 2300 mm and a thickness of 300 mm. Its 3D shape, ShapeW-101, can be represented by a

‘rectangular block’ in a 3D Cartesian coordinate system as is illustrated in Figure 60.

Example 2:

The shape of tank (Tank-201) is specified in a 3D Cartesian coordinate system as is illustrated in Figure 65. The tank has as 1D aspects a diameter D of 5000 mm, a height H of 2500 mm and a wall thickness of 5 mm. As 3D aspect it has a wall with a cylindrical shape. The tank could be completed with a cylindrical bottom plate and a conical roof (not shown in the figure).

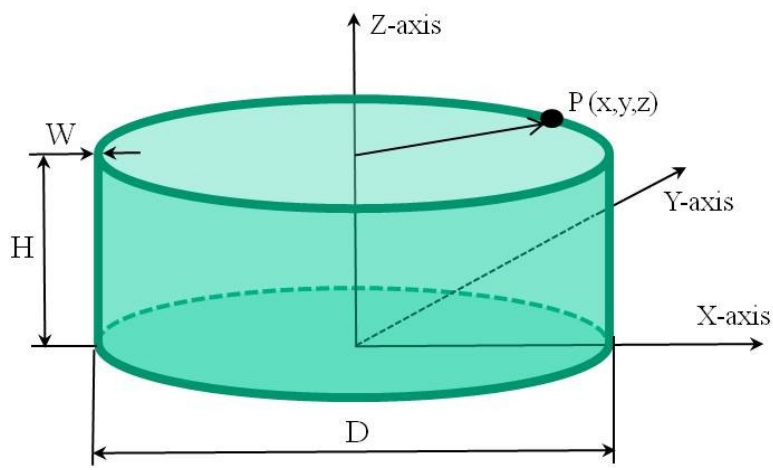


Figure 65, The shape of a cylindrical tank

9.4.4 One dimensional shape aspects (1D)

The recording of values of 1D aspects is usually pretty straight forward. The dimension is what we typically call a property or characteristic.

Example 1:

The properties of Wall-101 in a building can be recorded in a standardized, computer interpretable way as follows:

100	Wall-101	1225	is classified as a	40217	wall
100	Wall-101	1727	has as aspect	101	WL-101
101	WL-101	1225	is classified as a	551353	length
101	WL-101	5025	has on scale a value equal to	922355	5000
		mm			
100	Wall-101	1727	has as aspect	102	WW-101
102	WW-101	1225	is classified as a	550464	width

102	WW-101	5025	has on scale a value equal to	920466	300 mm
100	Wall-101	1727	has as aspect	103	WH-101
103	WH-101	1225	is classified as a	550126	height
103	WH-101	5025	has on scale a value equal to	922285	2300 mm

These and similar expressions of facts contain a number of standard concepts and phrases that are selected from the formal dictionary. For example, the concepts wall, length, 5000 and mm and also the phrases that express relation types, such as ‘is classified as a’ and the unique identifiers (UIDs) above 1000. Names of individual things and UIDs below 1000 are free.

Note: Details on allocation of other UIDs are discussed in part 4 of the Gellish Modeling Methodology.

Exercise: Define the material of construction of the wall in a similar way and select the standard concepts with their UIDs from the formal taxonomic dictionary (such as from the domain dictionary ‘Buildings and civil technology and furniture’).

Example 2:

The properties of tank Tank-201 can be recorded in a standardized, computer interpretable way in the formal language as follows:

210	Tank-201	1225	is classified as a	520686	cylindrical tank
210	Tank-201	1727	has as aspect	211	D-201
211	D-201	1225	is classified as a	550206	external diameter
211	D-201	5025	has on scale a value equal to	922355	5000 mm
210	Tank-201	1727	has as aspect	212	H-201
212	H-201	1225	is classified as a	550126	height
212	H-201	5025	has on scale a value equal to	922605	2500 mm
210	Tank-201	1727	has as aspect	213	Th-201
213	Th-201	1225	is classified as a	550303	wall thickness
213	Th-201	5025	has on scale a value equal to	920105	5 mm

In a similar way we can specify other properties of the tank, such as the material of construction.

9.4.5 Geometric models of 3D shapes

A shape of an individual physical object is an individual aspect of the object. Such a shape aspect can be represented by an individual geometric model. Such a geometric model usually consists of a composition of a number of relative simple basic geometric models, such that each of them can be represented by a standardized parameterized correlation in which the

parameters are replaced by individual aspects of the physical object that has the shape. Furthermore those basic geometric models are positioned in the right position and orientation in the coordinate system of the integrated geometric model.

There are various ways in which geometric models can be represented. The most common kinds of geometric models are ‘solid models’, especially ‘constructive solid geometry models’ (csg models) and boundary representation models, which are also called ‘wire frame models’. Such geometric models are typically composed of parameterized generic correlations or functions.

The generic correlations are typically implemented in executable programs (or algorithms) that are written in some computer programming language. A library of such programs can be implemented for example as a DLL (dynamic link library). For example, a company might offer a collection of graphical primitives in the form of a DLL library. *This document describes how such generalized correlations and their implementation in a software component library can be linked to the product model data in a neutral Database.*

The relations between executable programs (or algorithms) and the parameters in product models will be illustrated in the next paragraphs with the correlations for a block and for a hollow cylinder as primitives and for a complex geometric model.

9.4.5.1 CSG models

A csg model consists of one or a combination of more than one correlation that are instances of generic correlations for kinds of basic (convex) shapes, usually called primitives. Each generic correlation is a mathematical functions or formulae that is defined in its own 3D coordinate system and which sizes are defined by one or more parameters. An instance of such a generic correlation is created by tuning the correlation by specifying the values for the parameters that are specific for the dimensions of the individual shape and that define the correlation. The definitions of such generalized correlations include functionality descriptions and specifications of those parameters. For example, a (generic) solid sphere model is defined by a mathematical function $x^2 + y^2 + z^2 \leq r^2$ called ‘solid sphere’ with a radius (r) as parameter.

solid sphere	is a specialization of	csg primitive
solid sphere	has by definition as parameter a	radius

Thus an individual correlation (model) for shape-1, with radius R-1 is defined as follows:

Model of Shape-1	is classified as a	solid sphere
Model of Shape-1	has as parameter	R-1
R-1	is classified as a	radius
R-1	has on scale a value equal to	4.2 cm

A complex geometric csg model of an individual shape is created as a sequence of one or more unions, intersections or differences of less complex or basic models. This is illustrated in Figure 66, Shape decomposition and composition.

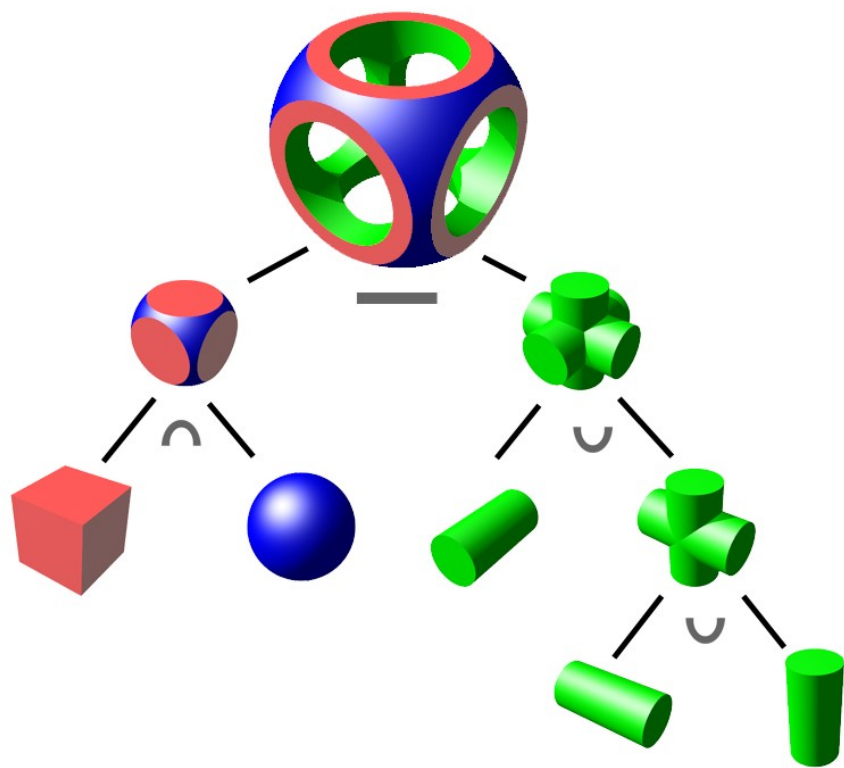


Figure 66, Shape decomposition and composition

Figure 66 shows a complex shape that is created by subtracting a union of three cylinders from an intersection of a solid sphere and a cube. In order to create such a complex shape, the shape should be decomposed into component shapes, until such a degree that each of them can be described

(modeled) by an instance of a tuned generic ‘primitive’ or basic correlation that is available in a library of algorithms. Then each individual primitive shape is described by classifying it by the kind of shape for which a correlation is available and by providing the parameter values of the individual shape as is suitable for the correlation. After that the component models shall be positioned at the right place and orientation in the coordinate system by translation and rotation. Then they shall be combined (united, intersected or subtracted) in the right sequence in order to derive the complex model.

9.4.5.2 Separate components – separate geometric models

Data exchange between various parties, such as in a supply chain, will include data exchange about components of an assembly. Therefore it is important that each component physical object in an assembly can be recognized as a separate object with its own properties and shape. Assemblies of physical objects should therefore be decomposed into single component physical objects, whereas the shape of each component is described by its own separate geometric model. This implies that the shape of a whole assembly should in general not be described by its own correlations, but should be composed of correlations that describe the shapes of its components. Thus the geometric model of an assembly consists of the combination of the geometric models of the components, whereas each component geometric model is positioned in the coordinate system of the assembly.

9.4.5.3 Standardization of (basic) geometric models for shapes

Data exchange or data integration of specifications of shapes between systems of different suppliers is often very difficult, because different software applications usually implement different correlations for the same shape and make use of different libraries of executable programs (macro’s) for shape representation. Therefore it is essential that the sending and receiving systems both use the same (standardized) correlations or software components or at least implementations of correlations that have identical functionality, names and parameters of shape representations.

Collections of generalized correlations are defined in various standards. For example, standard graphical primitives are defined in the ISO STEP standard for geometric objects ISO 10303-42 and in ISO 15926-3. Those correlations are also included in the Formal English taxonomic dictionary domain ‘Mathematics and Shapes’. Additional correlations can be defined and can be

added as (proprietary) extensions of the formal dictionary, conform the formal dictionary extension procedure.

Even single component physical objects can have complex shapes for which no executable program is available. Then it is necessary to decompose the shape into partial shapes, until a standard shape is found for which an executable program is available that can present such a shape on a drawing or that can visualize it on a screen.

9.4.5.4 Parameterized representations of shapes

A geometric correlation for a *kind of shape* is a generic mathematical function that has parameters. An instance of such a correlation has parameters that are be given values to tune the dimensions so that the individual correlation represents the shape of an individual object. A shape representation executable program or Macro is an implementation of such a generalized of one or more parameterized mathematical functions. Usually it can present any shape of a particular kind of shape, because it contains a generalized algorithm that is tuned by particular parameter values to represent individual shapes and sizes. The program or Macro will have a parameter list that has as elements the properties of the physical object or part which shape is represented by the function and that may include other presentation parameters.

For example, the Wall-101 has a shape, called ShapeW-101, that can be classified as a ‘block shape’ as is illustrated in Figure 60. This is specified as follows:

100	Wall-101	1727	has as aspect	110	ShapeW-101
110	ShapeW-101	1225	is classified as a	550810	block shape

Such an individual shape (ShapeW-101) can be modeled in principle by various different correlations. Let us call one of them SBlock-101, which is modeled by using a generic mathematical formula called ‘solid block’.

Furthermore, assume that the ‘solid block’ correlation <can be described by encoding as a>

$$F(x,y,z) = x + y + z$$

where: 0 =< x =< length
 0 =< y =< width
 0 =< z =< height

Figure 67, A mathematical function that represents a 'block shape'.

Note that the above correlation is defined in a 3D Cartesian coordinate system with the origin of the block (0,0,0) located at a corner of the block and that the correlation has three parameters: length, width and height.

Any individual shape, such as ShapeW-101, can be represented by such a parameterized correlation, by specifying the correlation (or an executable software program that implements the correlation), together with a specification of the values of the parameters.

This is done for example as follows: Assume that for the generation of a picture conform the above ‘solid block’ correlation with particular parameter values there is a proprietary Macro available, called SolidBlockMacro.

Then the shape of the wall can be specified in a formal language as follows:

110	ShapeW-101	5714	is modeled by a correlation as	111	SBlock-101
111	SBlock-101	1225	is classified as a	911239	solid block
111	SBlock-101	1876	is defined in coordinate system	150	CoordSys-101
111	SBlock-101	4962	has as parameter	101	WL-101
111	SBlock-101	4962	has as parameter	102	WW-101
111	SBlock-101	4962	has as parameter	103	WH-101

Notes:

1. On the first line it is specified that the shape is described by a particular (individual) correlation Sblock-101 that will have specific individual parameter values.
2. On the second line that correlation is classified by a predefined kind of correlation (solid block). The definition of that kind defines which kinds of parameters are required for correlation SBlock-101.
3. The correlation (911239) ‘solid block’ is a standard primitive correlation that is defined in and selected from the formal taxonomic dictionary and that is also defined in ISO 10303-214. The classification may be replaced by a classification by a subtype of that standard correlation.
4. The last three lines refer to the parameters of the correlation. The parameters were already quantified and classified in par. 9.4.4. The classification of the parameters shall comply with the requirements of the standard correlation for the kind ‘solid block’. This will enable software to determine which individual parameter value is applicable for which kind of parameter. For example, the software should use the classification of WL-101 to determine that the value of WL-101 (in mm) should be applied to the length L in the correlation ‘solid block’.

A formal language domain dictionary on shapes may be extended with additional standard correlations to represent shapes. Such definitions may include the specification of the parameters of the correlation.

A geometric model that is expressed in a formal language typically contains references to generic correlations that are defined in a formal language taxonomic dictionary. Such correlation shall be implemented in software and included in a library of correlations (e.g. as a DLL). Application software that uses the geometric model should therefore use both, the dictionary and the software library. For example, the above model refers to a ‘solid block’ correlation as defined above and therefore requires that such a correlation with the proper parameters is available in the software.

Exercise: The square opening in the wall has a shape that can also be represented by a solid block. Create a formal language representation of that shape, called ShapeO-101 in its own coordinate system.

9.4.6 Specification of components

As said before, an assembly should be decomposed until all single components are identified which shape need to be modeled.

For example, tank T-201 has a shape TankShape-201 as is illustrated in Figure 65. That shape is defined by a ShapeModel-201, which is a collection of mathematical functions. That shape model is defined in coordinate system of the tank CoordSys-201. This is specified as follows:

210	Tank-201	1727	has as aspect	233	TankShape-201
233	TankShape-201	4682	is described as	214	ShapeModel-201
214	ShapeModel-201	1876	is defined in coordinate system	201	CoordSys-201

Tank-201 is composed of three components, a wall, a roof and a base plate. The decomposition is specified as follows:

210	Tank-201	1225	is classified as a	520686	cylindrical tank
210	Tank-201	1190	has as part	215	Wall-201
215	Wall-201	1225	is classified as a	40210	wall
210	Tank-201	1190	has as part	216	Roof-201
216	Roof-201	1225	is classified as a	520069	conical roof
210	Tank-201	1190	has as part	217	Bottom-201
217	Bottom-201	1225	is classified as a	520042	base plate

Each of these three components has its own 1D properties. For the example we may assume that the properties of the wall are the same as the properties of the tank. Thus:

215 Wall-201	1727 has as aspect	211	D-201
215 Wall-201	1727 has as aspect	212	H-201
215 Wall-201	1727 has as aspect	213	Th-201

Note that the diameter of the tank wall (D-201) is the same property as the diameter of the tank (D-201). These are not two properties, but both objects share the same property (the same UID). So, if the value of that diameter changes, then it changes for both the tank and the wall. The same holds for the height and the wall thickness.

The roof is assumed to have its own different diameter. Those diameters can be specified independently, but it is also possible to correlate it to the diameter of the wall and wall thickness of the tank. (*Correlations between parameters is described below*)

9.4.7 Specification of components with standard 3D shapes

If a shape model of a single component cannot be described as a whole by one of the available standard shape correlations, then the shape has to be decomposed until each partial shape can be described by a standard shape. Assume that the shape of the wall can be represented by one of the standard shapes.

For example: Assume we create a 3D model of the wall. We distinguish between the wall and its shape. The shape of the wall is classified as a hollow cylindrical shape, as is illustrated in Figure 68.

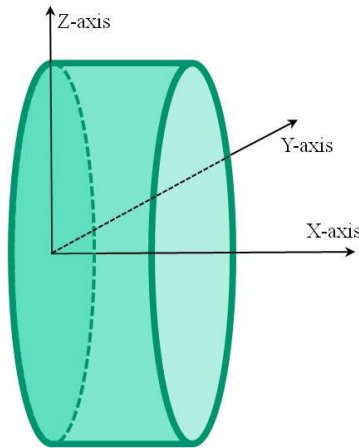


Figure 68, Definition of a hollow cylindrical shape

A hollow cylindrical shape is a standard parameterized kind of shape (550936) that is defined in the dictionary. An individual shape is related to such a kind of shape by a classification relation as follows:

215 Wall-201 1727 has as aspect 221 Cyl-201
 221 Cyl-201 1225 is classified as a 550936 hollow cylindrical shape

Kinds of shapes can be mathematically represented in various ways by mathematical functions. For example as by wire frame models or by ‘constructive solid geometry’ (CSG) models, which correlations (csg primitives) are specified in ISO 10303-42 and included in the formal dictionary. For example, a hollow cylindrical shape *in general*, can be mathematically represented by the parameterized geometric function ‘hollow cylinder’. The definition of that function is available in ‘Shapes’ domain dictionary, where it is specified that:

UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	Definition
550936	hollow cylindrical shape	2068	can be mathematically represented by a	911012	hollow cylinder	
911012	hollow cylinder	1911	is a concept that is described by	920015	hollow cylinder correlation-1	
920015	hollow cylinder correlation-1	1716	is a qualitative subtype of	928258	expression of a correlation in a 3D right handed Cartesian coordinate system	$r_i^2 \leq y^2 + z^2 \leq r_e^2;$ $0 \leq x \leq L;$ $r_e > r_i > 0;$ $L > 0$

The formula that describes a basic shape ‘hollow cylinder’ is defined in its own 3D Cartesian coordinate space (x, y, z). The formula requires three parameter values: an external radius (re), an internal radius (ri) and a height (L).

Note 1: Apparently the x-axis indicates the height of the cylinder and the origin (0,0,0) of its coordinate system is on the centre line in the bottom plane.

Note 2: Such a formula shall be implemented in software, for example as a Macro in JAVA code or in C++ code. The software that interprets the Formal English database content shall execute such code, in combination with the parameter values for an individual shape, to generate a presentation of a particular shape on an output device.

This means that an *individual* shape, such as Cyl-201, is described by an individual formula (F-201) that has three parameters. That formula is defined in the individual coordinate system WallCS-201 of Cyl-201. In other words, the shape of the component (Cyl-201) occupies a part of the space that is defined in the coordinate system of that component (WallCS-201).

This can be specified as follows:

221	Cyl-201	5714	is modeled by a correlation as	222	Func-201
222	Func-201	1225	is classified as a	911012	hollow cylinder
222	Func-201	1876	is defined in coordinate system	231	WallCS-201

The last step is the specification of the parameters and the values of the parameters of the correlation.

The parameters that are required for a particular kind of correlation for a standard shape are (normally) defined in a domain dictionary. For example, the formal language domain dictionary for shapes may contain the following specification of the concept ‘hollow cylinder’:

911012	hollow cylinder	5725	has by definition as parameter a	550343	external radius
911012	hollow cylinder	5725	has by definition as parameter a	553155	inside radius
911012	hollow cylinder	5725	has by definition as parameter a	550126	height

This means that the parameters for the individual correlation Fun-201 can be specified as follows:

222	Fun-201	4962	has as parameter	223	Re
222	Fun-201	4962	has as parameter	224	Ri

222	Fun-201	4962	has as parameter	225	L

Where:

223	Re	1225	is classified as a	550343	external radius
224	Ri	1225	is classified as a	553155	inside radius
225	L	1225	is classified as a	550126	height

Note: As all parameters are classified explicitly it is not necessary to specify the sequence of the parameters, nor group the parameters in a list.

These three 1D parameters of Cyl-201 differ from the 1D dimensional properties of Wall-201. However, their values can be derived from the values of the properties of Wall-201 by applying the following individual correlations:

$$Re = D1 / 2$$

$$Ri = D1 / 2 - W1$$

$$L = W1$$

Note: These correlations assume that the quantification of the parameters is done with a consistent set of scales (units of measure).

The first of these three correlations ($Re = D1 / 2$) is a mathematical law that relates a radius of a circle to its diameter. That law in general $f(r,d)$ can be defined by the formula $r=d/2$, with radius r and diameter d . Furthermore the function is a qualitative subtype of a division function. This is specified as follows:

220	$f(r,d)$	1726	is a qualitative subtype of	911685	division function

The division function has by definition two parameters: a radius and a diameter. The parameters are specified as follows:

220	$f(r,d)$	5725	has by definition as parameter a	550681	radius
220	$f(r,d)$	5725	has by definition as parameter a	550188	diameter

The second correlation is a subtraction function ($y = a - b$) and the third correlation is an equality function ($y = x$). The way to specify such

correlations in Formal English is illustrated below by the specification of the first correlation ($Re = D1 / 2$).

As the name of the individual function we use preferably the formula itself (denoted as a string of text such as: “ $Re = D1 / 2$ ”). Its parameters are denoted by their individual names as follows:

221	$Re = D1 / 2$	1225	is classified as a	220	$f(r,d)$
221	$Re = D1 / 2$	4962	has as parameter	223	Re
221	$Re = D1 / 2$	4962	has as parameter	211	D1

The individual parameters Re and D1 are classified already as external radius and diameter, so that software can unambiguously determine which parameter has which role in the correlation and can then calculate the missing value Re from the known value D1.

EXERCISE: Specify the correlations for the parameters Ri and L in a similar way.

9.4.8 Positioning of component shapes in a 3D coordinate system

The shape of an assembly of components is defined in the own 3D coordinate system of the assembly. Each shape of a component is defined in the own coordinate system of the component, with its own origin and its own orientation. It is possible to place the shapes of the components directly in the 3D coordinate system of the location where the assembly is built, but normally the shapes are first positioned in the coordinate system of the shape of the assembly.

For example: Above it is described how standard shapes of simple components, such as Cyl-201, are defined in their own coordinate systems. The next step is that those shapes (or those coordinate systems) need to be positioned in the coordinate system of the shape of the assembly. For example, shape Cyl-201 has to be positioned in CoordSys-201 of TankShape-201.

Note: Below it is specified that one shape is positioned relative to another shape. However, semantically this means that the coordinate system in which the positioned shape is defined is positioned relative to the coordinate system in which the reference shape is defined.

This done in Formal English using the positioning function as follows:

234	Positioning C in T	1225	is classified as a	5718	positioning in a 3D space	
234	Positioning C in T	5722	is a correlation with as reference object	201	CoordSys-201	
234	Positioning C in T	4886	is a correlation with as subject	231	WallCS-201	
234	Positioning C in T	1777	has as coordinate	235	x1,y1,z1	
235	x1,y1,z1	1225	is classified as a	553729	displacement in 3D	
235	x1,y1,z1	5025	has on scale a value equal to	236	0, 0, 0	mm
234	Positioning C in T	5841	has as orientation	236	a1,b1,c1	
237	a1,b1,c1	1225	is classified as a	553727	orientation in 3D	
237	a1,b1,c1	5025	has on scale a value equal to	236	0, 0, 0	rad

The displacement (x1, y1, z1) specifies the position of (the origin of) the coordinate system of the positioned shape (the subject) in the coordinate system of the reference shape. The orientation (alpha1, beta1, gamma1) specifies the rotation angles of (the coordinate system of) the positioned shape with respect to the coordinate system of the reference shape. Alpha is a rotation from y to z around the x-axis, beta is a rotation around the y-axis en gamma is a rotation around the z-axis.

The positioning of the WallShape-201 (Figure 68) in the coordinate system of the TankShape-201 (Figure 65) requires a displacement of (0,0,0) and a rotation around the y-axis of $\pi/2$ radians, thus (0, $\pi/2$,0).

The coordinates are in the units of measure that are defined for the whole coordinate system (e.g. mm and rad), or in units of measure that are explicitly specified on the lines where the values are quantified on scale.

9.5 Drawings and views

9.5.1 Kinds of drawings

A drawing is a document that consists of one or more 2D graphical images in the 2D coordinate system of an information carrier. Each image may be either a symbolic representation of something, or it may be a projection of a

3D shape (with or without color and texture) on a 2D surface. Such a projection may be for example a top view, a front view, a side view or a cross sectional view.

The creation of a 2D image or view from a 3D shape requires that the shape is first expressed in the form of a 3D geometric model that consists of mathematical correlations. A view or cross section of that 3D geometric model is then projected on a surface. Such a projection results in a 2D geometric model that consists of mathematical correlations for points, lines and flat surfaces. Those 2D geometric elements are expressed as correlations with parameters in a similar way as is described for the expression of the correlations and parameters that describe the shape of 3D objects.

Finally, a drawing to scale is created by conversion of the 2D geometric elements into physical graphical elements that consists of areas of ink on paper or light on a screen. Those physical graphical elements will have sizes that relate to the 2D geometric elements by a scale factor and they will have properties such as a color and a width, depending on factors, such as the kind of thing it represents, their visibility or invisibility and colors, whether the represented object exists, is new or is removed, etc.

Each image can exist in various physical forms. For example it may exist as a:

Image on paper	This is a presentation of an image as a print on a sheet of paper. Typically the print is produced from an electronic model by graphics enabled software, using a printer or plotter. The image may be embedded in other information, such as text or other images, or the sheet may only contain the image.
Model file	This is specification of an image in the form of a digital electronic file. For example a model in the form of a dxf or dwg file or a data file with an Expression Table.
Display	This is a presentation of an image in the form of light on a monitor. It is a view that is typically produced from a model by graphics enabled software, such as a CAD system.
Print file	This is an electronic version of a presentation of an image suitable for printing by a printer driver. Typically a print file is encoded in a dedicated printer driver format, such as Postscript (ps).

Multiple pages can be stored either as separate objects (files) or they can be assembled in one whole physical document or file.

The same information can exist (be presented) in more than one form. Therefore, we distinguish between the physical objects ('copies') of a document and the qualitative information (the document) that is presented on those copies. The relation between the two can be expressed by the relation that the information <is presented on> the physical copies. An example of one drawing (one piece of information) that is presented on two physical files (with the same content) is the following:

D-123 sheet 1 Rev. A	is presented on	D-123sh1RevA.dwg
D-123 sheet 1 Rev. A	is presented on	D-123sh1RevA.pdf

Whereas

D-123 sheet 1 Rev. A	is a type of	drawing (such as a top view or a schematic diagram)
D-123sh1RevA.dwg	is classified as a	electronic data file (in AutoCAD format)
D-123sh1RevA.pdf	is classified as a	electronic data file (in Acrobat format)

Two dimensional drawings can be either *drawings to scale* or they can be *schematic diagrams* that use symbols for the representation of objects. There are also combinations of these two kinds of drawings, in which some objects are presented on one scale and other objects on a different scale (e.g. a combination of a main drawing and a detail drawing) and/or in which also schematic presentations may be included.

Production of a physical drawing by a computer, in the form of a display on a screen or a print on paper, requires information (data) defining the drawing. Such information (expressions of facts) can be used by software in computers to generate actual physical graphical elements as an image, being ink on paper or light on screens in a specific shape, which present information to people.

Thus: A *physical drawing* is a physical object that is an assembly of a *medium* (an *information carrier*) and one or more *graphical elements*.

And graphical elements are described as follows:

Graphical elements on drawings are physical objects that can be described and can have aspects and relations just as all other physical objects.

In order to make a complete drawing, the graphical elements shall be defined and placed in a position and orientation in the coordinate system of a sheet of

paper or screen (information carrier) of a given size, such as A0, A3 or some screen size. Thus we will first describe the definition of graphical elements and then their positioning.

9.5.2 Graphical elements

Drawings and the graphical elements on drawings should be distinguished from the objects which they represent. We first discuss the drawings and graphical elements and later we discuss the relation between them and the objects which they represent.

Individual graphical elements need to be defined by classifying them by defined kinds of graphical elements that are included in the taxonomic dictionary and by specification of their aspect values. In addition to that they also need to be placed at a location and in a position (an angle) on a medium. The positioning requires a coordinate system in an x-y plane that on its turn is placed in the plane of the information carrier, as is described above.

A drawing consists of several graphical elements. Graphical elements are fundamentally all the same, because they all are just patterns of ink on paper or light on a screen with a defined shape and appearance. Nevertheless, depending on their role to represent some kind of thing, we can distinguish various kinds of graphical elements such as:

1. *Scaled graphical elements* which intent to present geometric details of the shape of a physical object in the physical reality or of a kind of physical object. They are used in *drawings to scale*, which means that a *scale factor* defines the ratio between the sizes of the shape of the represented physical object and the shape (geometry) of the graphical elements. This enables people to retrieve sizes and coordinates of the represented object from the geometry of the graphical elements. Such a scale factor is a property of a drawing.
2. *Schematic symbols* which are schematic representations of physical objects or activities, processes, properties or states, or which support dimensioning or which represent a relation (a fact) or a whole semantic network of information about an object. For example, symbols which represent e.g. a control activity or a distillation process or symbols which represent a flow direction, a direction of rotation, a slope, etc.
3. *Character strings*, being *text* or *numbers*, which represents information, such as names or property values.
4. *Annotation elements*, which are graphical elements that are added to drawings for clarification and as an aid to interpretation. For example, text

boxes, axes, measure lines, etc.

5. *Diagram elements*, which present correlations. For example: line and bar diagrams that represent mathematical functions such as $f(x,y)$ or physical correlations, such as pump curves.

When studied in detail it can be seen that all the graphical elements, such as *lines, points, areas* etc. in fact are *filled areas*, where e.g. a line width is the width of the area and the filling is the color of the area, which may be absent and may be transparent. From that it is clear that each graphical element is itself a physical object which has properties, such as a *shape, color, etc.* Such a shape of a graphical element may correspond to the shape of the represented thing (as on a drawing to scale) or its shape does not correspond with the shape of the represented thing (as on a schematic drawing).

The above description is illustrated by two examples. The first example is the definition of a typical straight line that is defined in its own coordinate system.

An example of a graphical element is a straight line, as is illustrated in Figure 69.

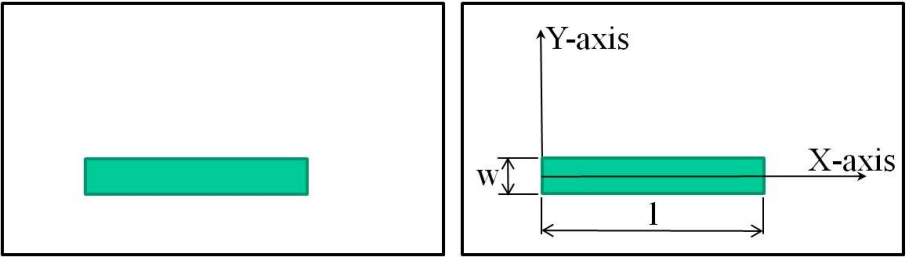


Figure 69, Graphical definition of a straight line

A straight line has the form of a rectangular filled area. Assume that the individual blue straight line has a length of 10 mm and a width of 0.5 mm. A model of that line can be expressed as follows:

line-1	is classified as a	straight line	
line-1	has as aspect	l1	
l1	is classified as a	length	
l1	is quantified on scale as	10	mm
line-1	has as aspect	w1	
w1	is quantified on scale as	0.5	mm
w1	is classified as a	width	
line-1	has as aspect	cl1	
cl1	is qualified as	blue	

The line is classified as a straight line. The definition of a straight line requires a mathematical model of the shape of the line (rectangular filled area) in the form of a correlation. Thus the line has a shape aspect that is classified as a rectangular shape and is modeled by a mathematical correlation, being a ‘rectangle in 2D’. This can be expressed as follows:

line-1
shape-l1
shape-l1
w1)

has as aspect

is classified as a

is modeled by a correlation as

shape-l1

rectangular shape

rectangular area in 2D (l1,

Name of left hand object	Name of relation type	Name of right hand object	Definition
rectangular area in 2D (l, w)	is a type of	correlation	$0 < x < l0; -0.5w < y < 0.5w$

The mathematical correlation of a ‘rectangle in 2D’ is defined as a mathematical model (formula). The mathematical model implies that a ‘native’ 2D coordinate system (x,y system) is used in which the model is defined. The relation between the shape aspects of a physical object, such as line-1 (with aspects l1 and w1) and the (parameters of) a mathematical correlation, such as the parameters for a rectangular area (l and w) is discussed in paragraph **9.4.5 Geometric models of 3D shapes**. When the mathematical correlation is implemented in drawing software, then the line can be printed or displayed.

Every real line will also have a direction. The mathematical correlation that describes the shape of the line is defined in its ‘native’ coordinate system. Thus the direction of the individual line (line-1) requires a positioning of the line in another coordinate system. Typically it is placed in the coordinate system of a bigger assembly or it is placed directly in the coordinate system of the sheet of paper on which the line is placed.

The second example is a model of a drawing with a circle as is illustrated in Figure 70. The figure on the left hand side shows a drawing that consists of only one graphical element, being a circle on a sheet of paper.

X-axis
Y-axis
R
W

Figure 70, Circle on a sheet of paper

The drawing of Figure 70 on the left hand side can be modeled, using the annotation on the right hand side, as follows:

Name of left hand object	Name of relation type	Name of right hand object	UoM	Definition
drawing-1	is classified as a	drawing		
sheet-1	is classified as a	A3 sheet		
sheet-1	is a part of	drawing-1		
circle-1	is a part of	drawing-1		
circle-1	is classified as a	graphical circle		
circle-1	has as aspect	Rc-1		
Rc-1	is classified as a	radius		
Rc-1	is quantified on scale as	10	mm	
circle-1	has as aspect	W-1		
W-1	is classified as a	line width		
W-1	is quantified on scale as	0.5	mm	
circle-1	has as aspect	shape-c1		
shape-c1	is classified as a	ring shape		
shape-c1	is modeled by a correlation as	ring in 2D (Rc-1,W-1)		
ring in 2D (Rc,W)	is a type of	correlation		$X^2 + Y^2 = R^2$, where $Rc - \frac{1}{2}W < R < Rc + \frac{1}{2}W$

Table 11, Model of a drawing with a circle

The above model illustrates that the paper sheet (sheet-1) and the circle (circle-1) together form the physical drawing (drawing-1). Note that circle-1 is not a mathematical circle, but a physical line (stripe, classified as a graphical circle) that has a line width and a radius. The radius should be interpreted as the radius of the centre line of circle-1. Circle-1 has as aspect shape-c1. That shape is classified as a ring shape and is modeled by a mathematical correlation, called a 'ring in 2D'. The mathematical correlation of a 'ring in 2D' is defined as a mathematical model (formula) of an area between two concentric circles. The mathematical model implies that a 'native' 2D coordinate system (x,y system) is used in which the model is defined. The radii of the inner and outer circles ($Rc - \frac{1}{2}W$ and $Rc + \frac{1}{2}W$) act as boundary values for R in the correlation. Their values can be derived from the values of the parameters Rc and W. The relation between the shape aspects (Rc-1 and W-1 of circle-1) and the

(parameters of) the mathematical correlation (R_c and W of a circle in general) is discussed in paragraph 9.4.5.

Mathematical correlations, such as the correlations for a rectangular area and a ring, should be implemented in drawing software. When that is available, then the graphical elements can be printed or displayed on the basis of models such as the models of a line and a circle in the above tables.

Graphical elements are defined in their native coordinate system, but they are positioned in another coordinate system. The positioning of graphical elements in a 2D coordinate system is discussed in paragraph 9.5.4.

The above description of a drawing of a line and a circle is independent of what the drawing represents! It is only a description (a model) of filled areas of material on a carrier. The line and the circle may represent anything. For example, the circle may represent a top view of a tube, drawn to scale, but it may also be (a part of) a symbol that represents e.g. a cooler on a schematic drawing, etc.

9.5.3 Schematic graphical elements

In schematic drawings the graphical elements have a shape that generally does not correspond precisely to the shape of the represented objects. Those graphical elements are physical objects in their own right, being ink on paper (or equivalent). Such objects, for example lines, have a width, a length, a color, a shape, etc. Thus, schematic drawings consist of physical graphical elements that are positioned in a 2D coordinate system of the sheet or screen on which the drawing is presented.

For example, a cooler may be represented by a cooler symbol. Such a cooler symbol may consist of among others a circle and a line. An example of a cooler symbol on a sheet of paper is illustrated in Figure 71, Cooler symbol.

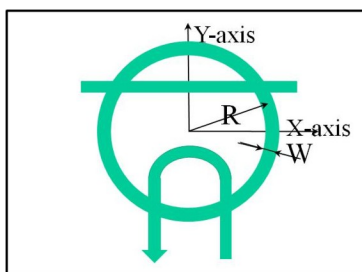


Figure 71, Cooler symbol

Symbols that do not reflect the shape of a represented object are often called *schematic symbols*. Also their location on a drawing is more or less independent of the location of the represented object in the real world, although the location typically specifies a logical ‘position’, such as a sequence in a process. For example, a ‘block diagram’ is a diagram that contains rectangles (blocks) as symbols for all represented objects. Also the shapes of symbols on schematic diagrams, such as instrument symbols, have no real correspondence with the shapes of the represented objects. Construction drawings on the other hand, contain graphical elements of which the shape does correspond with for example a front view shape of the object which is intended to be constructed.

A *schematic symbol* is typically applied in schematic drawings but may also appear on hybrid drawings.

Models of schematic drawings that only consist of graphical elements (symbols, etc.) and their locations on the drawings, without relations between the graphical elements and the represented objects are called models of dumb schematic drawing models. The essence of ‘intelligent schematic drawing’ models is that those models distinguish between the symbols and the symbolized objects, whereas an intelligent model includes for each symbol a relation that specifies which object is represented by the symbol. For example, it is called ‘intelligence’ when a model of a schematic diagram contains a relation between a tank and a tank symbol, such as:

210	Tank-201	2072	is graphically represented by	251	Symbol-
201					
251	Symbol-201	1225	is classified as a	610423	tank
	symbol				

Note that each individual symbol is defined in its own coordinate system.

Individual symbols are created by applying correlations in the form of executable programs (macros). Those correlations are typically aggregates of basic correlations (primitives) for basic symbols, such as straight lines, circles, segments, etc. which shapes are defined by mathematical functions. Those correlations are normally parameterized, so that symbols of various sizes can be created at various locations by tuning the correlation by its parameter values. From a user perspective the individual symbols are copied from typical symbols that are standardized and that are included in a symbol library. In reality the typical symbol is itself also created by using the macro with fixed (default) parameter values.

System independent data integration and data exchange requires that the names, functionality and parameters of correlations (executable programs or macro's) are standardized. Typically they will be collected in proprietary, national or international 'symbol libraries' (which are in fact 'software libraries'). Smooth data exchange and data integration is only possible if the symbol libraries are either Open or supplied and available to all involved parties.

For example, assume that a macro is available that defines a standard general tank symbol. The macro is called TankMacro-1. Its correlation is defined in coordinate system TankCS-1 and has as parameters a scaling factor, a color, a line type and a line weight. Then TankSymbol-201 is defined as follows.

251	Symbol-201	5714	is modeled by a correlation as	252	TankMacro-201	
252	TankMacro-201	1225	is classified as a	4679	correlation	
252	TankMacro-201	1876	is defined in coordinate system	253	SymbolCS-201	
252	TankMacro-201	4962	has as parameter	254	Scale-201	
254	Scale-201	5020	is qualified as	589198	1:1	
252	TankMacro-201	4962	has as parameter	255	LineColour-201	
255	LineColour-201	5020	is qualified as	551759	red	
252	TankMacro-201	4962	has as parameter	256	LineType-201	
256	LineType-201	5020	is qualified as	790383	dashed	
252	TankMacro-201	4962	has as parameter	257	LineWidth-201	
257	LineWidth-201	1225	is classified as a	1660	line width	
257	LineWidth-201	5025	has on scale a value equal to	920538	0.3	m m

9.5.4 Positioning of graphical elements in 2D

Once a graphical element is defined, it has to be positioned in the coordinate system of a larger assembly, or in the coordinate system of a drawing. For example the positioning of symbol SymbolCS-201 on drawing T-123 with drawing coordinate system T-123-CS can be specified as follows:

260	Place S-201inT	1225	is classified as a	5717	positioning in a 2D space
260	Place S-201inT	5722	has as reference object	261	T-123-CS
260	Place S-201inT	4886	has as subject	253	SymbolCS-201
260	Place S-201inT	5378	has as coordinates point	262	250, 20
260	Place S-201inT	5348	has as parameter	263	alpha-201
263	alpha-201	5025	has on scale a value equal to	920100	0
263	alpha-201	1225	is classified as a	553726	orientation in 2D

Note that the units of measure may be defined for the reference coordinate system as a whole.

Any positioning is a relative positioning of a physical object relative to another physical object.

A positioning is therefore a definition of a correlation that relates a positioned object to a reference object, plus the definition of the actual position (coordinate point) and orientation of the positioned object, relative to the reference object, which means in the coordinate system of the reference object.

The definition of the position (also called 'location') and the orientation can be qualitatively (by text), or quantitatively by numeric values on a scale. A qualitative specification may be for example by values such as 'above', 'on top of' or 'within' and by 'rotated clockwise 30 degrees', etc.

A quantitative specification of a position and orientation is by specification of the coordinate point in the 2D or 3D coordinate system of the reference object and the left angle or triple of angles for the orientation values. That coordinate point can be specified by a pair or triple of values on a scale.

For example, the position and or orientation properties consists of Δx , Δy , Δz and $\Delta \alpha$, $\Delta \beta$, $\Delta \gamma$ in their respective units of measure.

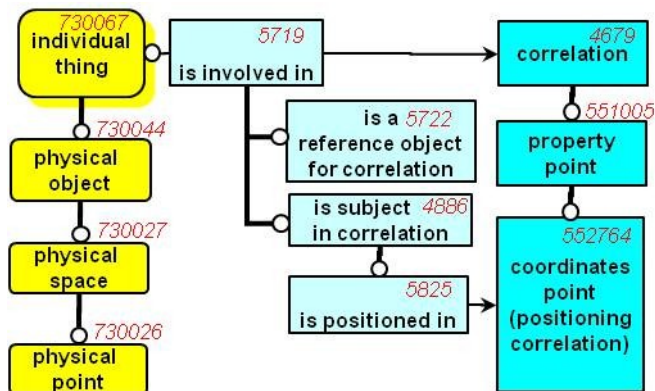


Figure 72, Relative positioning in a coordinate system

For example the positioning of Inlet-1 above tray-5 in a distillation column means that the coordinate system of Tray-5 is the reference object, whereas the location of Inlet-1 is a coordinates point in that coordinate system. Such a coordinates point is a correlation because the point in a 3D space represents a correlation between three coordinates.

Without mentioning an explicit coordinate system, the positioning correlation can be introduced as follows:

Positioning of inlet-1	has as reference object	Tray-5
Positioning of inlet-1	has as subject	Inlet-1

When the point in space is not precisely quantified, but the location of Inlet-1 is to be qualified only for example as ‘above’, then that location can be modeled by stating that the correlation has a parameter that is classified as a location aspect, whereas the parameter is qualified as ‘above’. This can be expressed as follows:

Positioning of inlet-1	has as parameter	location-1
location-1	is classified as a	location
location-1	is qualified as	above

These expressions might be replaced by a more precise specification of the height above Tray-5 as follows:

Positioning of inlet-1	has as parameter	Height-1
Height -1	is classified as a	height
Height -1	has on scale a value equal to	300 mm

The location aspect as a parameter is presented at the top of Figure 73. The precise location in a 3D space requires that the coordinate system of the reference object (e.g. Tray-5) is explicitly defined, whereas the coordinates of the coordinates point are explicitly quantified. The model that shows the relations that are required to do so are presented in Figure 73.

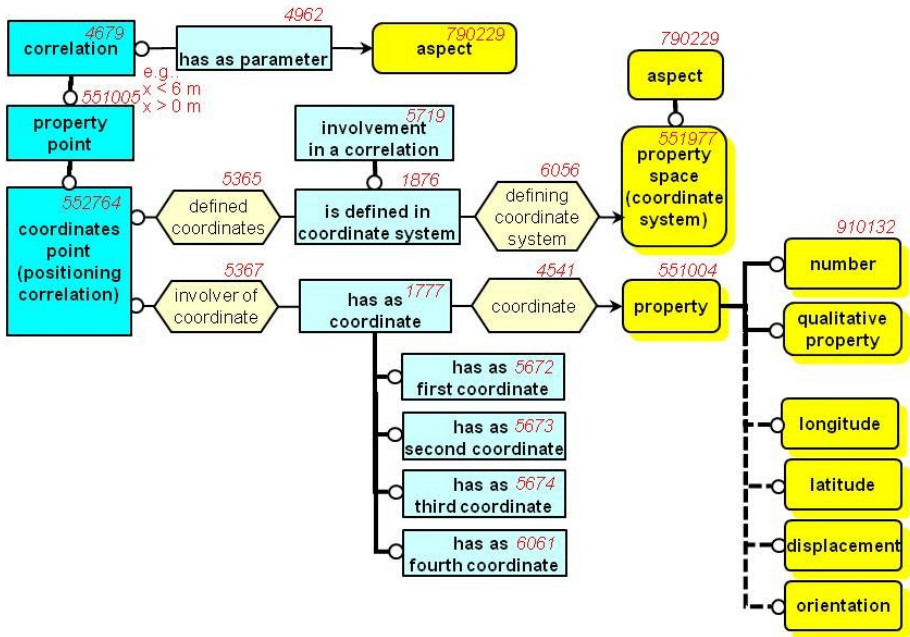


Figure 73, Positioning in a coordinate system

Instead of mentioning and quantifying each coordinate separately it is possible to use a short-cut kind of relation that relates the coordinates point to a list of values, as is illustrated in Figure 74.

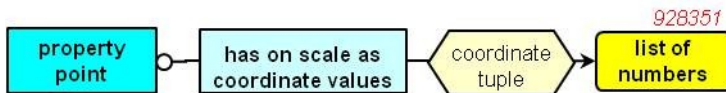


Figure 74, Short-cut relation to a list of coordinate values

A positioning relation between one graphical element and another describes the fact that a positioned graphical element *is positioned relative to* a reference graphical element. The positioning is then further defined by a position and an orientation of the positioned graphical element in the coordinate system of the reference object.

Positioning of (other) physical objects (materials) and features should be defined in the same way by a positioning relation between one physical object and another. That relation can then be defined by position and orientation of the positioned object in the coordinate system of the reference object.

In this context the term 'feature' of an item, includes things as a physical point (e.g. one that indicates the origin of a coordinate system), or a surface, a centre line, a tangent line or a reference line.

The relative position of two features of items can be defined in the same way by a positioning relation. For example, the position and orientation of the end of a nozzle, placed eccentric on a head of a horizontal vessel (see Figure 75, Nozzle position), can be described as is illustrated in Table 12.

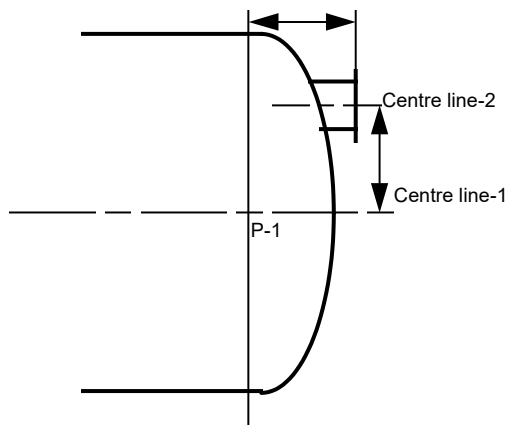


Figure 75, Nozzle position

In Table 12, the point P-1 is used as a reference point.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object	UoM
14001	1	V-1	is classified as a	101	vessel	
14002	2	S-1	is classified as a	102	shell	
14002	2	S-1	is a part of	1	V-1	
14003	3	H-1	is classified as a	103	head	
14003	3	H-1	is a part of	1	V-1	
14004	4	C-1	is classified as a	104	centre line	
14005	4	C-1	is possessed by	1	V-1	
14006	5	T-1	is classified as a	105	tangent line	
14007	5	T-1	is possessed by	1	V-1	
14008	6	N-1	is classified as a	106	nozzle	
14009	6	N-1	is a part of	1	V-1	
14010	7	F-1	is classified as a	107	flange	
14011	7	F-1	is a part of	6	N-1	
14012	8	FF-1	is classified as a	108	flange face	
14013	8	FF-1	is a part of	F-1	flange	
14014	9	C-2	is classified as a	109	centre line	

14015	9	C-2	is possessed by	6	N-1	
14016	9	C-2	is positioned rel. to	4	C-1	
14017	14016	R1	has as aspect	10	D1	
14018	10	D1	is quantified on scale as	110	300	mm
14019	10	D1	is classified as a	111	distance	
14020	14016	R1	has as aspect	11	A1	
14021	11	A1	is quantified on scale as	4	0	degree
14022	11	A1	is classified as a	112	angle	
14023	8	FF-1	is positioned rel. to	9	C-2	
14024	14023	R2	has as aspect	12	D2	
14025	12	D2	is quantified on scale as	113	500	mm
14026	12	D2	is classified as a	111	distance	

Table 12, Positioning of a nozzle

Positioning is either exact or approximate.

Approximate positioning (without coordinate systems)

An approximate definition of a positioning is possible without the definition of coordinate systems. For example with the information content: 'on top of' or 'next to'.

Exact Positioning (with Coordinate systems)

An exact definition of a positioning includes a position and orientation of the item relative to another item. To give a complete definition of a positioning of an item relative to another item it is required to define (an origin of) a coordinate system on that item and on the referenced item.

This means that for unambiguous positioning every item requires the definition of (at least) one coordinate system in which it is in rest.

To avoid the necessity of explicit definition of a coordinate system for every item the following rules apply:

1. Every item has coordinate system (and thus an origin, which is the centre of that coordinate system) that is intended to be used to place the item relative to another coordinate system and intended to be used in the definition of the shape of the item.

If the origin and the coordinate system is not defined explicitly, then the following rules apply:

2. If the item is an assembly, then the origin of the assembly coincides with the origin of the main item.

3. The origin of an item is located in the geometric gravity point of the item.
4. If an item, i.e. the main item in an assembly, has a centre line through the geometric gravity point, then the centre line coincides with the x-axis.

If one side of that centre line is longer than the other, then the longest side of that centre line coincides with the x-axis in the positive direction.

5. If the direction of the axes is still undefined, they shall be defined explicitly in the definition of the item.
This can be done for example in one of the following ways:
 - by definition of the location of one or two additional sub-origins and/or
 - by definition of one or more lines or flat surfaces and/or
 - by the definition of position and orientation of a normal vector in a point on a surface.
6. Unless defined otherwise, the coordinate system is a Cartesian coordinate system. However cylindrical coordinate systems and polar coordinate systems are also possible.

For example,

1. From rule 1 and 3 follows that the origin of a circle is located in its centre point and that the origin of a cylinder or a block is located in the middle of its centre line. (See O_1 and O_2 in Figure 76 A and B)
2. From rule 1, 2 and 3 follows that the origin of a vessel with a dome and several nozzles and with two head is still located in the middle of the centre-line of the bare vessel (see O_0 in Figure 76A). Because of lack of symmetry it is still required to define the direction of the y and the z-axis. E.g. the z-axis is in the direction of the dome. The direction of the y-axis is then defined by the Cartesian rule that the direction of the y-axis is defined by the direction of the rule of the corkscrew.

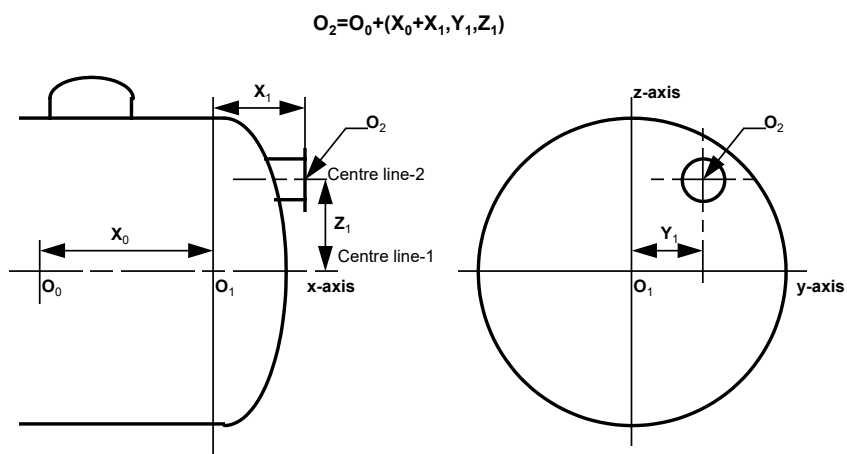


Figure 76A/B, Side and front view of a vessel

9.5.5 Assemblies of graphical elements

A graphical element, such as a symbol, can be defined as an assembly of other graphical elements. Such a composition hierarchy may continue until the shapes of the component graphical elements can be described by Macro's that are available in a library of executable programs that can present such shapes. A library of basic geometric primitives, such as a line, a triangle, a polyline and a circle, is defined in various standards. The assembly process for symbols is illustrated in Figure 77.



Figure 77, A typical pump symbol

Assume that three macro's are available: circle (radius), equilateral triangle (length) and connect point, whereas the point has no parameters. Each of the shapes is defined in its own 2D Cartesian coordinate system and has its own location of its origin. The shape of the pump symbol as a whole is then defined by creating individual graphical elements, with specified parameter values, followed by the positioning of those elements in the coordinate system of the pump symbol (assembly). The positioning of 2D graphical elements is described in paragraph 9.5.4.

9.5.6 **Standard or typical graphical elements**

Graphical elements on drawings are often selected from predefined *typical graphical elements*, such as *typical symbols*. Such typical graphical elements are usually elements of a library of *standard* typical graphical elements, such as in a symbol library. Those standard typical graphical elements are defined with a particular size, line style, etc.

The specific individual graphical elements on drawings are then ‘derived from’ the typical graphical elements either by reuse or by duplication of the aspects of the typical graphical element.

For example, assume that the straight line (line-1) that is defined above is a standardized typical symbol and is included in a library of graphical elements. Then a specific line (line-2) can be defined as a line that reuses the aspects of the typical line or a specific line (line-3) can have aspects that are derived from the aspects of the typical line. The relations between the lines can then be expressed as follows:

line-2	is a reuse on scale of	line-1	2.0
line-3	is derived on scale from	line-1	2.0

In case of a reuse on scale, the specific individual graphical element is not defined by aspects, but the aspects of the typical graphical element are intended to be used instead.

In case of a derivation on scale all the facts that define the typical graphical element are copied to become an identical new collection of facts for the derived specific graphical element. This means that the collection of facts (the model) that defines the derived graphical element is a duplicate of the collection of facts (the model) that defines the typical graphical element. (Note that the positioning of a typical symbol is not part of its definition.) The new facts and aspects shall get new UUIDs and the sizes shall be multiplied by the scale factor. Thus, in the above example, the model of line-2 does not include qualitative aspects, because the aspects of line-1 are used instead, taking into account the scale factor. The model of line-3 does include aspects, which values are derived from the values of the aspects of line-1, whereas the sizes are multiplied by the scale factor.

A library of standard or typical graphical elements may include typical lines of various types, such as dashed and dotted lines, and it may include many assemblies of lines such as boxes, pump symbols, valve symbols, nozzle symbols, etc.

For example, the *typical nozzle symbol* TN-1 of Figure 78 is defined as described below.

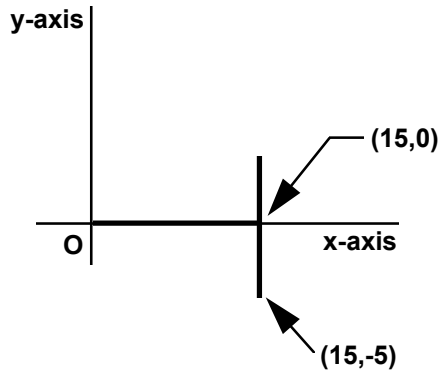


Figure 78, Definition of typical nozzle symbol N-1

A nozzle symbol is an assembly of two lines, L-1 and L-2.

Every object is (by definition) defined in a 'native' coordinate system. So the typical nozzle symbol is defined in its nozzle symbol co-ordinate system, whose origin is called O, and each line is defined in its own native line coordinate system.

The positioning of the lines means that their coordinate systems are positioned relative to the coordinate system of the nozzle symbol. The result in this example is that the typical nozzle N-1 has a defined origin and orientation, which makes a later positioning of the nozzle unambiguous.

The detailed definition of the typical nozzle symbol is as follows:

<i>UID of fact</i>	<i>UID</i>	<i>Name of left hand object</i>	<i>UID of relation type</i>	<i>Name of relation type</i>	<i>UID</i>	<i>Name of right hand object</i>	<i>UoM</i>
14001	1	TN-1	1225	is classified as a	101	nozzle symbol	
14002	2	L-1	1225	is classified as a	102	straight line	
14003	2	L-1	1190	is a part of	1	TN-1	
14004	2	L-1	1727	has as aspect	4	l1	
14005	4	l1	1225	is classified as a	104	length	
14006	4	l1	5025	has on scale as value	103	15	mm
14007	2	L-1	1727	has as aspect	5	a1	
14008	5	a1	1225	is classified as a	106	angle in the x-y plane	
14009	5	a1	5025	has on scale as value	105	0	deg
14010	3	L-2	1225	is classified as a	102	straight line	
14011	3	L-2	1190	is a part of	1	TN-1	
14012	3	L-2	1727	has as aspect	6	l2	
14013	6	l2	5025	has on scale as value	107	10	mm

14014	6	l2	1225	is classified as a	104	length	
14015	3	L-2	1727	has as aspect	7	a2	
14016	7	a2	5025	has on scale as value	108	90	deg
14017	7	a2	1225	is classified as a	106	angle in the x-y plane	
14018	8	Positioning of L-1	1225	is classified as a	5717	positioning in a 2D space	
14019	8	Positioning of L-1	5722	has as reference object	1	TN-1	
14020	8	Positioning of L-1	5886	has as subject	2	L-1	
14021	8	Positioning of L-1	5378	has as coordinates point	586749	0,0	mm
14022	8	Positioning of L-2	1225	is classified as a	5717	positioning in a 2D space	
14023	8	Positioning of L-2	5722	has as reference object	1	TN-1	
14024	8	Positioning of L-2	5886	has as subject	3	L-2	
14024	8	Positioning of L-2	5378	has as coordinates point	110	15,-5	mm

Table 13, Definition of a Nozzle Symbol

Note: in this example, the origins of the coordinate systems are not made explicit. Positioning of L-1 and L-2 implies positioning of 'its' coordinate system in the coordinate system of the reference object. However it is possible to define more than one coordinate system for an object. In that case it is necessary to define the coordinate systems explicitly as a feature possessed by the symbol and to define the positioning as a positioning of the coordinate system.

Once the typical nozzle symbol TN-1 is defined it is possible to derive a specific nozzle symbol N-1 from the typical one and to place it on a drawing.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object	UoM
14014	4	N-1	is classified as a	101	nozzle symbol	
14015	4	N-1	is derived from	1	TN-1	

Table 14, Derivation of a Specific Symbol from a Typical Symbol

9.5.7 Annotation elements on drawings

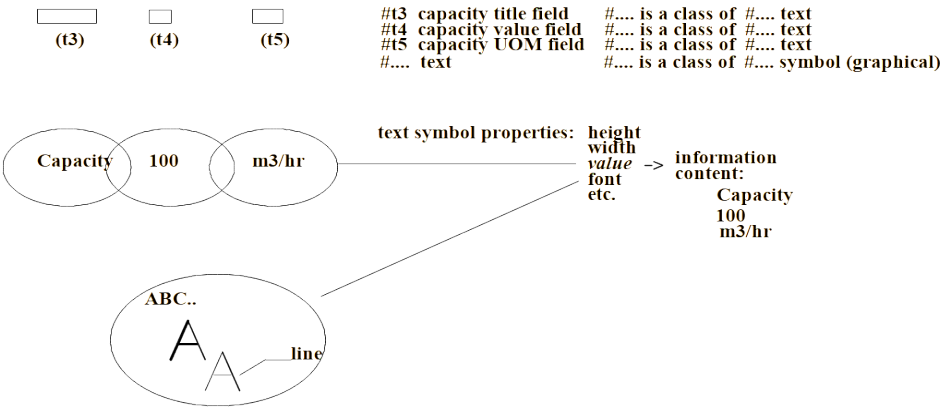
9.5.7.1 Names and values

Names of objects, such as tag numbers, values for capacities and sizes of equipment are *texts* (character strings) that are presented as physical

graphical elements (annotation) that have properties, such as font, height, width, slant angle, color, etc.

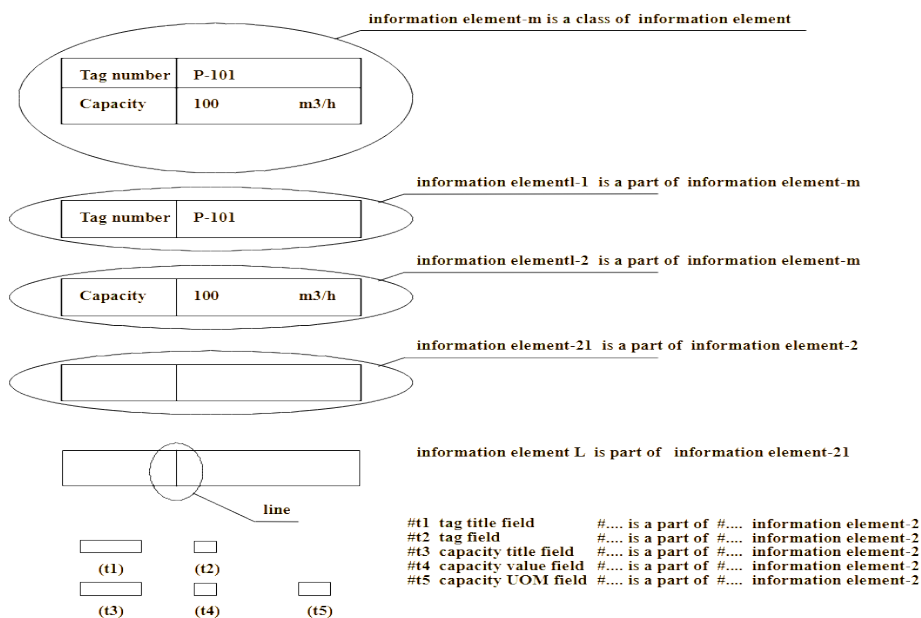
When symbols are defined that include textual components, not all values of these textual components may be available yet. In the example below the defined text symbols (fields) have names that correspond with the meaning of the content. Examples of fields are: 'capacity title field', 'capacity value field' or 'capacity UOM field'.

The *font* of text in text symbols is the layout and shape of geometric elements that make the text characters.



9.5.7.2 Assemblies of text and symbols

An intelligent set up of symbols will result in a flexible definition of geometry dependant on the amount of text to be presented and such that the drawing layout requirements are met. Parametric procedures can replace laborious drafting work. With composition relations symbols can be assembled in a modular way.



The basic shapes of geometry in symbols can be pre-defined from typical annotation elements while the actual annotation elements can have properties with the proper dimensions for a specific symbol with known text symbol elements.

Good examples of this are *functional symbols* like the 'slope symbols', the geometry and text is dependant of actual information associated with the annotated product.

9.5.8 Schematic diagrams

Schematic diagrams are drawings that are usually made for the purpose to provide overview. This is especially useful for complex facilities such as process plants. Most symbols on schematic drawings represent physical objects, although their shape and dimensions have hardly any relationship with the shape or dimensions of the represented objects.

On '*dumb graphic diagrams*' the relationship between a symbol and the object which it represents is not even made explicit. Therefore dumb graphic diagrams are diagrams that can only be interpreted by human beings, because of their ability to derive meaning from the shapes of the symbols (graphical elements).

'Intelligent schematic diagrams' are diagrams, where the symbols have explicit relations with the objects which they represent.

For example:

symbol #1 represents physical object #2 which is classified as a centrifugal pump.

Dumb graphics as well as intelligent graphics diagrams can be described in formal languages. However, the philosophy behind data integration and data sharing prefers 'intelligent graphics', because that is the only computer supported way which enables to keep the data on diagrams consistent with other data.

A human being can derive much more information from a schematic diagram than computers can (nowadays). A person interprets graphical symbols in the context of what he expects from the perspective of his specific business view point. For example, when a person sees a symbol of a pump on a diagram he might derive the following information:

- the fact that the pump has a purpose to increase of pressure in a distillation column,
- the fact that the pump is of a specific type, such as a centrifugal pump,
- the fact that it has specific connections with other items, such as a pipeline,
- the fact that the suction and the discharge are at the top of the pump,
- the fact that the pump is submerged,
- the fact that the pump is electric motor driven,
- etc.

In intelligent graphics diagrams, all such information is associated with the represented object, e.g. the pump itself. All this information may or may not be made explicit by decomposing the symbol, such that each symbol part explicitly represents one of those aspects. Nevertheless it is possible to build up such a symbol from various symbol parts and to define for each part what it represents. For example, a valve symbol might be build up from smaller symbols which represent e.g. whether it is panel mounted, whether it has a chemical seal, a normally open position, the valve type, symbolic flanges, etc.

In a true intelligent graphics system this decomposition of symbols is done and the relations with the information about the physical objects are made explicit.

Examples of kinds of schematic diagrams that can be drawn using similar constructs are:

- Process flow diagrams (PFD's)
- Piping and instrumentation diagrams (P&ID's)
- Logic diagrams
- Control diagrams
- Loop diagrams
- Instrument hook-up diagrams
- Block diagrams
- Sequence charts

9.5.8.1 Diagrams that represent processes

Process diagrams are diagrams that represent processes. They are always schematic diagrams. The symbols on a process diagram are usually intended to represent processes (such as unit operations), activities or events. The processes may be business processes or physical and chemical processes or transport processes to move material from one place to another. Diagrams that represent business processes are also called activity diagrams. Examples of such activity diagrams are (maintenance and operation) procedures and activity diagrams for commissioning and start-up.

Typically processes and activities are related to the physical objects that have the role of performer or enabler or subject in a process. For example, a pumping process is related to a fluid being pumped and a performer being a pump and a transport process is related to a material being transported and a pipe or a vehicle being the performer or enabler of the transport process. These relations imply that in a process model (sometimes referred to as an 'intelligent process diagram') the graphical symbols are directly related with the processes and activities and they are indirectly related with the physical objects that are their performer or enabler!

Sometimes the symbols on a process diagram are considered to represent physical objects that have the role of performer or enabler or subject in a process. Strictly speaking such diagrams are not process diagrams. However, indirectly they still intent to specify processes as a sequence of process steps, indicated by the flow direction of the streams. Therefore, such diagrams are nevertheless called process diagrams. An example of such a kind of process diagram is a Process Flow Diagram (PFD) according to ISO 10628.

Whether symbols on process diagrams represents processes (activities) or they represent physical objects, in any case in process models both the processes (activities) and the physical objects should be defined and there

should be relations of the appropriate kind defined between the processes and the physical objects.

Furthermore it is recommended to standardize on the convention that the lines between the symbols in process diagrams represent the *transport processes* in which the *streams* (moving batches of material information carriers or energy) have roles as *subjects*.

Finally, the fact that a line is connected to a process (unit operation symbol) means that a fluid has a role as feed to (input) or product from (output) that process, depending on the flow direction.

For example, Figure 79 illustrates a process in which a fluid stream is heated in a heat exchanger and flows to a simple distillation column (splitter) where it is split (distilled) into two streams.

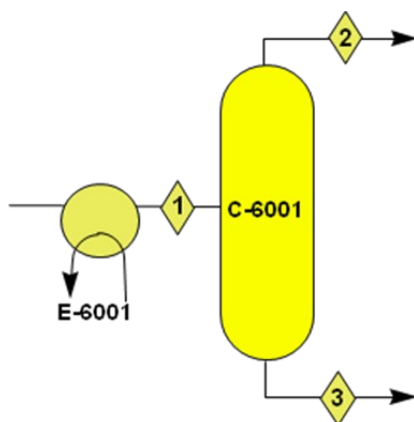


Figure 79, Simple process flow diagram

The schematic diagram of Figure 79 shows five symbols. According to the above conventions those symbols represent:

1. Fluid S1 that is flowing through heat exchanger E-6001 to distillation column C-6001.
2. The heating process by the heat exchanger E-6001.
3. The distillation process in the distillation column C-6001.
4. The top product S2 transported to a next process.
5. The bottom product S3 transported to a next process. For example transported to storage.

Every of symbol or relation between symbols represents some information. For example, symbol S1 and symbol E-6001 represent information (facts) that can be expressed as follows:

fact 1	stream-S1	is subject in	transport-1
fact 2	stream-S1	is represented by	line-1
fact 3	E-6001	is enabler of	heating-1
fact 4	E-6001	is represented by	symbol-2
fact 5	stream-S1	is output of	heating-1
fact 6	stream-S1	is input in	distill-1
(this fact is presented by the fact that the end of line-1 touches symbol-3)			
etc.			

Often various process cases are defined, which means different processes are defined. With the above convention it should be noted that when there are more process cases (i.e. different processes), then it should be made explicit that a symbol or a line on a process diagram represents more than one fact: for every process the performer and the involved stream(s).

Decomposition of processes

A process in a process unit can be described in various level of decomposition. The following two levels are relevant to provide the necessary context of the process plant definition.

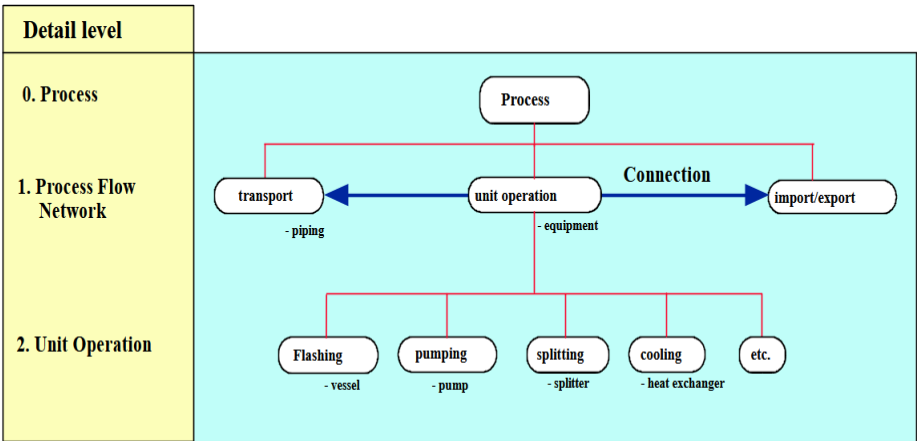


Figure 80, Process detail levels

At all levels there are streams, which are fluids that are subject to a transport process (moving material). The top level represents the streams which make up the material and energy balances of the total process.

At the next level they form the material and energy balances of the various unit operations, which are internal streams from the viewpoint of the total process. On the third level they form the material and energy streams within those unit operations.

In the next paragraphs we will discuss the representation of these processes and streams at those various levels of decomposition.

9.5.8.2 Process Flow Network level

A *process unit* is an assembly of facilities that together allow the execution of a physical or chemical or transport process, including conversion and storage of process material. Such process units are described as process flow networks.

The main objects in a process flow network are:

- The *unit operations*, to transform or transport material (incl. energy) into one or more other materials or states. They are physical, (bio)chemical or mechanical processes that occur in or with process material (fluids or solids). The processes normally take place within equipment.
- The *imports* or *exports* of process material to and from unit operations.
- The *transport* between those *imports* or *exports*.

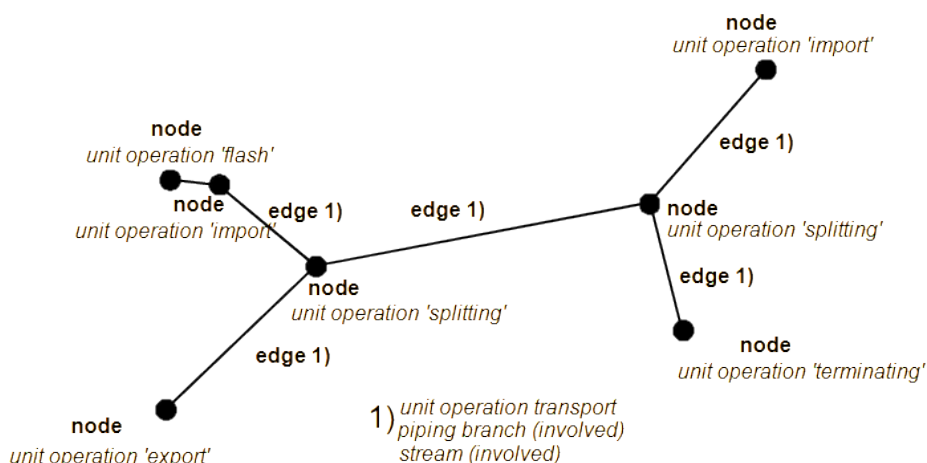


Figure 81, A process network

The relationship between unit operations which forms the network is defined by the fact that a process material (also called a *stream*) is exported from one unit operation and is imported in the next unit operation.

Processes operate on process material. Unit operations are activities, whereas process material is defined as a 'substance', which is a subtype of 'material'.

The relationship in Formal English between a 'process' and the 'process material' that is involved in that process is described by the involvement relation between 'activity' and 'material', whereas the type of involvement is defined by the classification of the involvement.

Substances (process materials) *can be involved in a* unit operation *in the role of* 'input', 'subject', 'output' or 'catalyst' or one of their subtypes. Energy, for example in the form of electricity or heat (e.g. carried by steam), can also be in a role as 'input', 'subject' or 'output' of a unit operation.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
	90,02 1	stream 1	is subject in	90,001	transport 1
	90,02 2	stream 2	is subject in	90,002	transport 2
	90,02 2	stream 2	is subject in	90,005	flow control 1
	90,02 2	stream 2	is subject in	90,006	flow measurement 1
	90,02 3	stream 3	is subject in	90,003	transport 3

	90,02 4	fluid in V6060	is subject in	90,007	level measurement 1
	90,02 1	stream 1	is output of	90,010	increase pressure by P6052
	90,02 2	stream 2	is input of	90,011	transfer heat by E6051
	90,02 3	stream 3	is input of	90,012	flash by V6060
	90,02 1	stream 1	is input of	90,015	splitting 1
	90,02 2	stream 2	is output of	90,015	splitting 1
	90,02 3	stream 3	is output of	90,015	splitting 1

Equipment is usually involved as enabler, performer or tool in a unit operation.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
		P6052	is performer of	90,005	increase pressure by P6052
	91,009	pipng branch 1	is performer of	90,001	transport 1
	91,009	pipng branch 2	is performer of	90,001	transport 1
	91,009	pipng branch 3	is performer of	90,001	transport 1
	91,009	pipng branch 4	is performer of	90,002	transport 2
	91,009	pipng branch 5	is performer of	90,003	transport 3
		splitter 1	is performer of	90,015	splitting 1
		V6060	is performer of	90,012	flash by V6060

The concept 'process' is subtypes of occurrence, because both describe 'something that happens'. Unit operations such as flashing, pumping, splitting and cooling, are subtypes of process. The same holds for control processes such as control of a level, measurement of a level and measurement of a flow rate. The classification of individual processes is illustrated in the following table.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object
	90,001	transport 1	is classified as a	220,141	transport
	90,010	increase pressure by P6052	is classified as a	220,143	unit operation
	90,016	control level-1	is classified as a	190,757	control

	90,007	level measurement 1	is classified as a	190,112	measure
	90,006	flow measurement 1	is classified as a	190,112	measure
	90,010	P6052 export	is classified as a	191,223	export

9.5.8.3 Unit Operation level

On a *unit operation* level the process is described as the physical and chemical phenomena that take place within a *process unit*, such as distillation, flashing, reaction, transport, mixing, splitting, cooling, heating, etc.

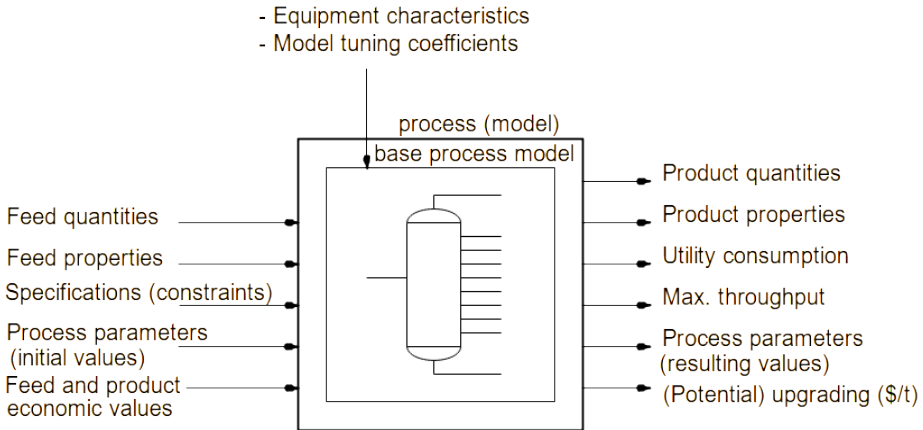


Figure 82, Process detail level: Unit operations

The physical and/or chemical phenomena can be represented by a process model (algorithm). When executed it simulates a process and defines the relationship between the feed streams, process parameters and outputs, constraints, economics and objective function, which might be optimized. For an actual process the relevant relationships are 'defined' by the real world phenomena.

The behavior of a process or of a process model is not only influenced by the fluids and process conditions, but also by some equipment characteristics, like diameters and surface areas. When a process is represented by a process model, then model tuning coefficients can be applied which tune a bare process model to the actual behavior of a real world process.

The data that describes the process on the unit operation level is not described in this issue of this report.

9.5.8.4 Schematic representation of processes

Processes are typically presented in schematic diagrams the show a network of process stages as mentioned in section 9.5.8.2. An example of such a kind

of diagram is a Process Flow Diagram (PFD), also called Process Flow Schema (PFS).

A *process unit*, described at a level of detail which we called ‘the Process Flow level’, is described as a *network*. The symbols at the *nodes* in the network represent *unit operations (processes)* and the *equipment* in which *process material* is *transformed*. The symbols which form the *edges* in the network represent the *transport (process)* of *process material* between the nodes. Altogether these relations describe the topology of the process flow network as well as the *physical objects* involved.

For example, the graphical symbols (annotation elements) in Figure 83 represent the various unit operations, which are the various activities that take place on the fluids, such as separation process (flash in V-6060), transport of its bottom liquid to pump P-6052, the pumping activity, etc. Various materials play different roles in those activities.

For example:

stream-1	is output of	pumping process-1	
P-6052	is performer of	pumping process-1	
stream-1	is subject in	transport-1	
transport-1	is classified as a	transport process	bring stream-1 from P6052 to splitter-1
stream-1	is input of	splitting-1	
splitter-1	is performer of	splitting-1	
stream-2	is output of	splitting-1	
stream 3	is output of	splitting-1	
etc.			

All this (and more) of this data about the physical objects and their behavior (processes) is presented by the symbols in this diagram.

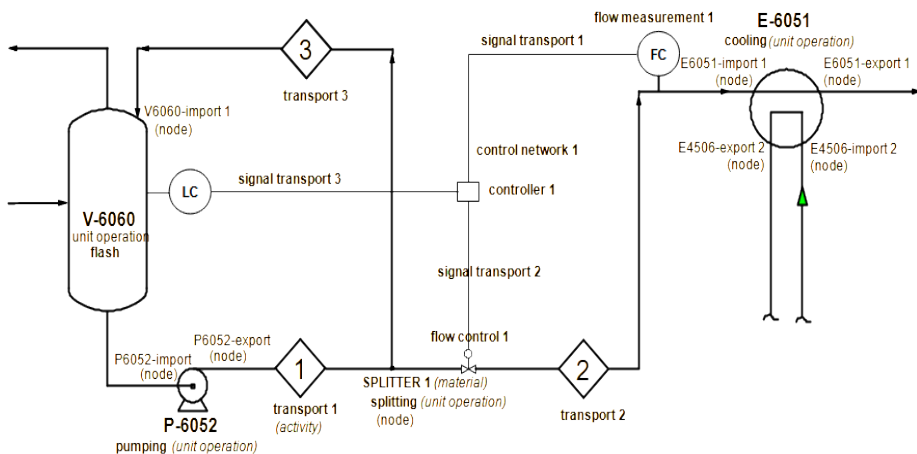


Figure 83, Process detail level: Process Flow Diagram (PFD)

9.5.8.5 Diagrams that represent networks of physical objects

A computer interpretable definition of a product does not require any graphical (visual) presentation. All information that is presented in graphics form can be made explicit as data in a product model.

To facilitate human interpretation it is convenient that the information is also presented graphically. For example in schematic diagrams (using 'symbols'), scaled views (side views, cross sectional views, etc) and geometric models (usually called 3D models). These paragraphs explain the principles of an (intelligent) graphical presentation of product data, using a Piping and Instrumentation Diagram (P&ID) as illustration material. The same principles apply for other applications, such as ortho-graphical drawings (piping arrangements), isometric drawings and other technical documents.

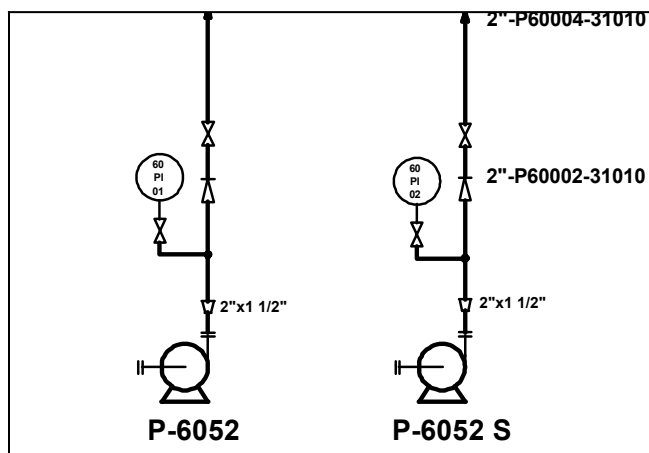


Figure 84, A schematic diagram (section of a P&ID)

Schematic drawings, such as the section of the example P&ID in Figure 84, may present items on a component level by means of symbols. Various kinds of *symbols* or *schematic elements* are defined as kinds of *graphical elements*. This implies that a symbol is a physical object (e.g. ink on paper). A symbol is intended to represent something graphically. The shapes of the symbols may symbolize aspects of the presented things but on schematic diagrams they are more or less independent of shapes and locations in the physical reality. Symbols can be textual or non textual.

The *graphical element* is the material that presents information; it is the visual carrier of information, typically as ink or toner on paper or a light emitting pattern on a screen.

In the next paragraphs it is illustrated that schematic diagrams (such as a P&ID) can present implicit information on (de)compositions and connections at various assembly levels of materials or activities (processes). To manage the consistency of the data across documents and databases it is essential to model the information in an explicit, computer interpretable form.

Schematic diagrams of a particular type usually represent objects on a particular assembly level. For example, items at some higher assembly levels may not be represented by separate symbols, but they may be represented implicitly by arrangements of symbols. For example, pipelines and piping segments are usually not represented by one symbol, but by an arrangement of symbols.

Below (in Figure 87 and Figure 89) is illustrated how arrangements of

symbols can be made explicit by indication of the branches, segments and junctions (nodes) in networks.

Piping and Instrument Diagrams (P&IDs)

The intent of schematic diagrams such as P&IDs is that the connection between represented items is visualized via a symbolic presentation. The represented items typically have a role as enablers or performers of processes. For example, equipment, piping and instrumentation material enable processes in fluids inside those components.

Because of the nature of a P&ID traditionally the interpretation of a P&ID is such that a symbol represents e.g. *a piece of equipment* or a *piping component* and that the lines between the symbols represent the *pipes*, *electrical cables* or *instrument lines*. Etc. This means that in an ‘intelligent drawing’ the graphical symbols are related with any of the physical objects in the product model, as for example is discussed in paragraph 6.12

Graphical symbols need an information carrier, such as a sheet of paper, in order to enable human observation. Therefore, the individual graphical elements are arranged in an overall coordinate system, together with drawing layout annotation, like drawing boundaries and a drawing header. Together they form an arrangement or image that is located on a carrier. One or more images and a carrier together form a *page*. Often schematic drawings are printed on separate sheets. Therefore, it is a widespread practice to call such a page a ‘sheet’. An *image* is a graphical element intended to be presented on one side of a *sheet* or on a screen. A *sheet* is an item that consists of one piece of paper or foil, intended to function as information carrier. One or more images on one or more sheets form a *document*. The sheet typically will be described by characteristics such as material of construction ‘paper’ and properties such as height, width and paper thickness. Alternatively the graphical elements can be displayed on a screen. In the latter case the expressions should result in a presentation of the graphical elements on a computer GUI.

It is recommended to relate the graphical elements to the identification of the diagram, e.g. a drawing and sheet number, rather than to the document. This will make the identification independent of the chosen information carriers which display the information to the users.

The table below shows an example of expressions that describe a schematic diagram and the relation between the symbols and symbolized objects.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object

501	101	T-123.456	is classified as a	490,183	piping and instrumentation diagram
503	102	symbol centrifugal pump P6052	is classified as a	610,054	centrifugal pump symbol
504	103	symbol tag number P6052	is classified as a	610,425	text
505	104	symbol nozzle P6052-outlet	is classified as a	610,283	nozzle symbol
610	132	A1 sheet 1	is classified as a	490,189	sheet
611	133	origin A1 on sheet 1	is classified as a	730,041	2D Cartesian coordinate system
612	134	drawing template 1	is classified as a	610,298	sheet - frame with title block
613	135	T-123.456 sheet 1	is classified as a	610672	page
527	503	symbol centrifugal pump P6052	is a part of	613	T-123.456 sheet 1
540	515	symbol pipeline P60001	is a part of	613	T-123.456 sheet 1
619	613	drawing template 1	is a part of	613	T-123.456 sheet 1
620	610	A1 sheet 1	is a part of	501	T-123.456
622	613	T-123.456 sheet 1	is a part of	501	T-123.456
612	610	A1 sheet 1	is defined in coordinate system	611	origin A1 on sheet 1
614	613	drawing template 1	is positioned relative to	611	origin A1 on sheet 1
615	503	symbol centrifugal pump P6052	is positioned relative to	611	origin A1 on sheet 1

Intelligent diagrams: relations between diagrams and product data

As said before, the term ‘intelligent diagrams’ denotes integrated information about graphical elements as well as about the represented objects and the relations between them. Such integration enables users to access information about objects in a product database via one of the symbols in a drawing and vice versa. This means that there will be a relation between an object and a symbol that represents the object. For example, such a relation can be expressed e.g. as P-101 *is graphically represented by symbol #5*’.

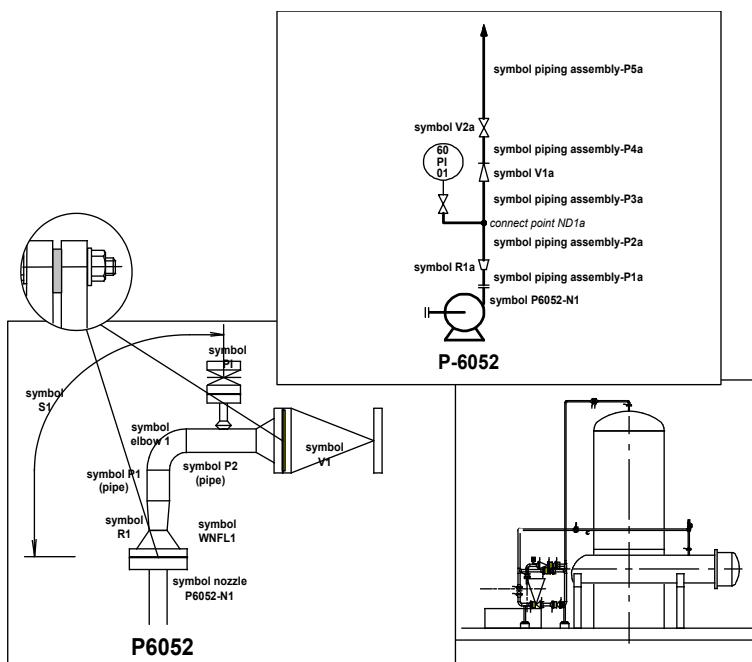


Figure 85, Symbolic representation of physical objects on various kinds of drawings (P&ID's, piping arrangements, isometrics, etc.)

Organization may define standards that specify which kinds of physical objects shall be related to graphical elements. This may for example correspond with the rules to define which items shall have tag names. One approach is that all (the coordinates of) the tag names on a drawing are related with the corresponding objects in the product. Another approach is that (the coordinates of) particular kinds of graphical symbols are related to the represented objects in the product. When a user points to a coordinate on the drawing, the software may find the nearest graphical element (symbol or text) that represents an object in the product. Once an object is found, further navigation thorough the product model and retrieval of data can be done without needing the graphics.

Representation of objects by graphical elements

542	503	symbol centrifugal pump P6052	is graphically represented by	201	P6052
543	504	symbol tag number P6052	is graphically represented by	201	P6052
544	505	symbol nozzle P6052-outlet	is graphically represented by	202	P6052-N1

The graphically represented objects P6052 and P6052-N1 in the above table are the physical objects that are discussed in chapter 6.12. Thus this relation

implies that the symbols on the diagrams can be found via the objects in the product model.

It depends on the user requirements which relations between symbols or arrangements of symbols and represented objects are made in practice. The system with which the drawings are made should be customized accordingly.

Intelligence (drafting functionality)

Application of “intelligence” in graphics refers to abilities that are provided by the data to enable software to draw conclusions and to verify various qualities, such as completeness and consistency of the graphical and non-graphical data. For example, if a symbol on a diagram is added, then intelligence enables that the symbol is not only recorded in a database, but also enables that it is recorded which object is symbolized and whether the kind of symbol corresponds with the kind of object.

In a CAD software application for drawings the majority of the information dealing with the graphics is traditionally stored in a proprietary CAD file formats (such as the dwg format). This hampers the verification of completeness and consistency with data that is stored in other formats. But, the more CAD software generates graphical data as an integrated part of product data in a neutral form, using a formal dictionary, the more intelligence can be applied, because the relations provide links with non-CAD data and enables verification of consistency of data on different places.

Another form of intelligence aims to support Computer Aided Drafting features. As is said above, the data is can usually be handled exclusively by proprietary CAD application software. However, it is possible to transform (part of) that data into a neutral form. For example, it is recommended to convert at least the data that is required to verify whether the topology and the connections in schematic diagrams is consistent with the data in other diagrams and consistent with non-CAD data. The following paragraphs describe how such information should be modeled in order to be able to exchange it between different parties.

Connectivity in networks at node and branch level

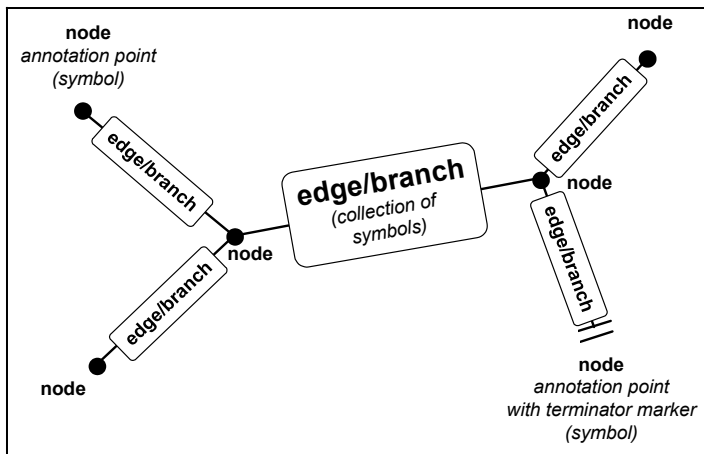


Figure 86, A network of branches and nodes

At a fundamental level a graphical presentation of a *network* is described by *edges* (*branch* is a synonym) and *nodes*. The *edges* may be *pipng branches*, *road network branches*, *electrical network branches*, etc. The *nodes* may be physical objects such as equipment, devices, connection assemblies, roundabouts, etc. They may also be denoted by roles of role players. Examples of roles are *inlets* and *outlets* and *network terminators* (see Figure 86).

A node in a schematic drawing typically constitutes of a *connection point* which can have *connector features*. A connection point connects items in a network and acts as a handle to catch and manipulate one or more graphical elements.

The connection point is usually part of a symbol presented on a drawing. A connection point may be visualized by the centre of a marker and can be controlled via CAD functionality. However, the point itself is typically not visualized.

A branch is a connection between network nodes. It consists of an arrangement of connected symbols that present objects. For example, piping components, electrical items, instruments, or road segments. Figure 87 illustrates branches and nodes in a schematic diagram of a network.

A *network terminator* is a connection to the environment or it is a dead end. For example, it can be a connection to atmosphere (an open end) or a dead end of a street. In a schematic drawing a network terminator constitutes of a *terminator marker* being an annotation point intended to mark a point that is

the end of a graphical element. For example, a terminator marker may represent a piping end closure object or a piping end open to atmosphere.

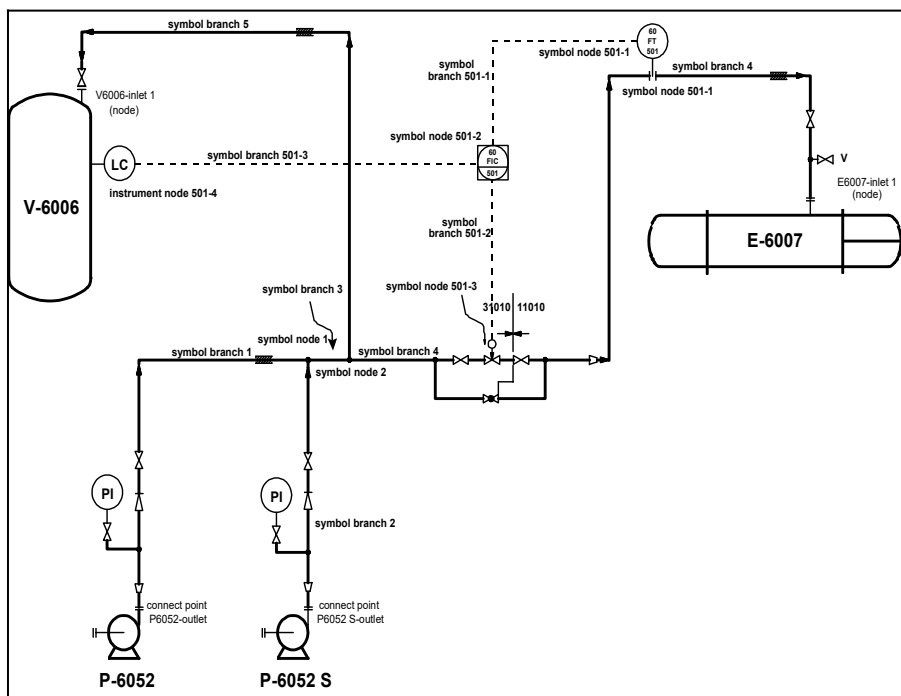


Figure 87, Network at piping branch level (part of a P&ID)

Connectivity at Equipment / Segment level

A *network segment* is an assembly of materials which scope is defined by certain constant attributes along the segment. These constant attributes are typically shown as part of the segment identifier, such as a line label. In a graphical representation it may be that the graphical elements are grouped such that the segments are identified separately.

When a network is defined at a more detailed level by the introduction of segments, then a more refined definition of material and network occurs as is shown in Figure 88 and Figure 89.

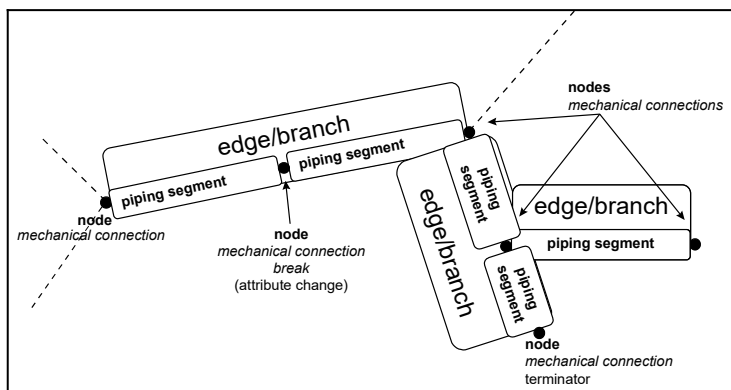


Figure 88, Network at piping segment level

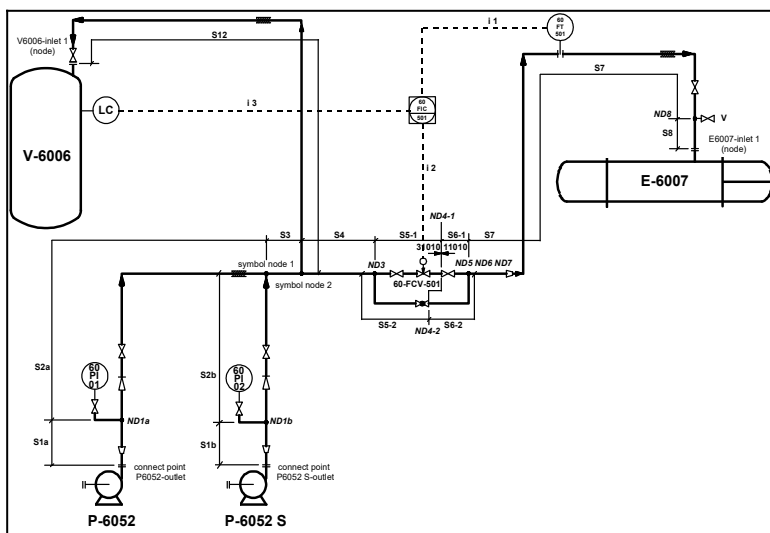


Figure 89, Graphical elements grouped in segments

The following tables show example data concerning the graphical representation of objects in a P&ID.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object

Graphical elements at branch & segment level

507	105	symbol branch 1	is classified as a	610,313	pipe segment symbol
508	106	symbol node 1	is classified as a	610,085	connector symbol
509	107	symbol node 2	is classified as a	610,085	connector symbol
510	108	symbol ND1a	is classified as a	610,085	connector symbol
511	109	symbol S1 a	is classified as a	610,313	pipe segment symbol
512	110	symbol S2 a	is classified as a	610,313	pipe segment symbol

513	111	symbol 60-PI-01	is classified as a	610,331	pressure gauge symbol

Graphical elements at higher aggregation level

515	114	symbol pipeline P60001	is classified as a	610,313	pipeline symbol
516	115	symbol pipeline tag number P60001	is classified as a	610,425	text

Connect points in annotation network at branch & segment level

518	116	connect point P6052-outlet	is classified as a	610,229	connect point
519	117	connect point ND 1a	is classified as a	610,229	connect point
520	118	connect point node 1	is classified as a	610,229	connect point
521	119	connect point node 2	is classified as a	610,229	connect point

Composition of graphical elements, equipment level

523	505	symbol nozzle P6052-outlet	is a part of	503	symbol centrifugal pump P6052
524	504	symbol tag number P6052	is a part of	503	symbol centrifugal pump P6052
525	518	connect point P6052-outlet	is a part of	505	symbol nozzle P6052-outlet

Composition of graphical elements at branch & segment level

529	510	symbol ND1a	is a part of	91,009	symbol 60-PI-01
530	511	symbol S1 a	is a part of	507	symbol branch 1
531	517	symbol S2 a	is a part of	507	symbol branch 1
532	519	connect point ND 1a	is a part of	510	symbol ND1a
533	520	connect point node 1	is a part of	508	symbol node 1
534	521	connect point node 2	is a part of	509	symbol node 2

Composition of graphical elements at higher aggregation level

536	516	symbol pipeline tag number P60001	is a part of	515	symbol pipeline P60001
537	507	symbol branch 1	is a part of	515	symbol pipeline P60001
538	513	symbol 60-PI-01	is a part of	507	symbol branch 1

Presentation of objects by graphical elements

542	503	symbol centrifugal pump P6052	presents	201	pump 1
543	504	symbol tag number P6052	presents	415	P6052
544	505	symbol nozzle P6052-outlet	presents	202	P6052-N1
545	507	symbol branch 1	presents	224	piping branch B1
546	508	symbol node 1	presents	348	node C1
547	509	symbol node 2	presents	209	node C3
548	510	symbol ND1a	presents	213	node C7

549	511	symbol S1 a	presents	237	piping segment S1
550	512	symbol S2 a	presents	238	piping segment S2
551	513	symbol 60-PI-01	presents	239	(missing ref in material)
552	515	symbol pipeline P60001	presents	291	pipeline 1
553	516	symbol pipeline tag number P60001	presents	292	P60001
554	518	connect point P6052-outlet	presents	207	node C1
555	519	connect point ND 1a	presents	213	node C7
556	520	connect point node 1	presents	208	node C2
557	521	connect point node 2	presents	209	node C3

Connectivity at branch & segment level

559	511	Conn P6052outlet-ND1a	is classified as a	1487	connection relation
560	511	Conn P6052outlet-ND1a	has as source	518	connect point P6052-outlet
561	511	Conn P6052outlet-ND1a	has as destination	519	connect point ND 1a
562	511	Conn P6052outlet-ND1a	is using as connection material	513	symbol S1 a
563	512	Conn ND1a-node 1	is classified as a	1487	connection relation
564	512	Conn ND1a-node 1	has as source	519	connect point ND 1a
565	512	Conn ND1a-node 1	has as destination	508	symbol node 1
566	512	Conn ND1a-node 1	is using as connection material	514	symbol S2 a

Connectivity at higher aggregation level

567	516	Conn P6052outlet-node 1	is classified as a	1487	connection relation
568	516	Conn P6052outlet-node 1	has as source	518	connect point P6052-outlet
568	516	Conn P6052outlet-node 1	has as destination	508	symbol node 1
568	516	Conn P6052outlet-node 1	is using as connection material	515	symbol pipeline P60001

Connectivity at Symbol level

When in the representation of a system the segments are decomposed we arrive at the level where the elements to draft the schematic diagrams are symbols that are typically selected from a CAD symbol library. The symbols that represent components are often connected via lines. Those lines typically represent pipes, cables or road segments. A line on a schematic diagram does not necessarily imply that in reality a physical object is present. For example, a line may represent a connection between two objects, whereas the objects may be welded directly against each other.

Typically the presented shape of symbols is more or less independent of the shape and location in the physical reality.

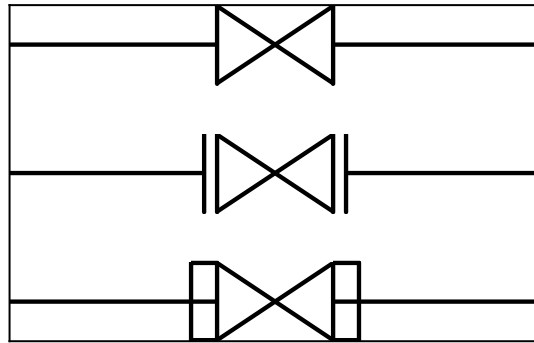
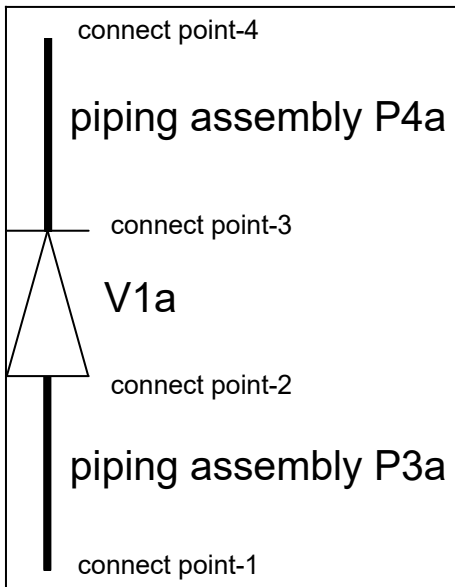


Figure 90, presentation of connection material

The level of detail regarding connections that are presented on diagrams may vary, as is illustrated in Figure 90. Sometimes a technical solution such as a flange or a welded connection is indicated. At a specific moment in the design process these details may not yet be known because certain items are not yet defined.



Connections between physical objects typically use connection materials. For example, piping components, such as a flange, valve, elbow, tee, (bare) pipe, etc. are connected or coupled via connection material, such as gaskets, bolt sets, flanges and welds. On the symbol level those connection materials may be represented by e.g. *flange symbol*, *valve symbol*, *elbow symbol*, *tee symbol*, *pipe symbol*, etc. which are connected/coupled graphically via *connect points*.

The data tables below show an example of how schematic elements are defined at symbol level.

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object

Graphical elements at component SYMBOL level

567	120	symbol piping assembly-P1a	is classified as a	610,314	pipe symbol
568	121	symbol R1a	is classified as a	610,346	reducer symbol
569	122	symbol piping assembly-P2a	is classified as a	610,314	pipe symbol
570	123	symbol piping assembly-P3a	is classified as a	610,314	pipe symbol
571	124	symbol V1a	is classified as a	610,062	check valve symbol
572	125	symbol piping assembly-P4a	is classified as a	610,314	pipe symbol
573	126	symbol V2a	is classified as a	610,462	valve symbol
574	127	symbol piping assembly-P5a	is classified as a	610,314	pipe symbol

Connect points at component SYMBOL level

576	128	connect point 1	is classified as a	610,084	connect point
577	129	connect point 2	is classified as a	610,084	connect point
578	130	connect point 3	is classified as a	610,084	connect point
579	131	connect point 4	is classified as a	610,084	connect point

Composition of graphical elements at component SYMBOL level

581	567	symbol piping assembly-P1a	is a part of	511	symbol S1 a
582	568	symbol R1a	is a part of	511	symbol S1 a
583	569	symbol piping assembly-P2a	is a part of	511	symbol S1 a
584	570	symbol piping assembly-P3a	is a part of	512	symbol S2 a
585	571	symbol V1a	is a part of	512	symbol S2 a
586	572	symbol piping assembly-P4a	is a part of	512	symbol S2 a
587	573	symbol V2a	is a part of	512	symbol S2 a
588	574	symbol piping assembly-P5a	is a part of	512	symbol S2 a
589	576	connect point 1	is a part of	571	symbol V1a
590	577	connect point 2	is a part of	571	symbol V1a
591	578	connect point 3	is a part of	572	symbol piping assembly-P4a
592	579	connect point 4	is a part of	574	symbol piping assembly-P5a

Presentation by graphical elements at component SYMBOL level

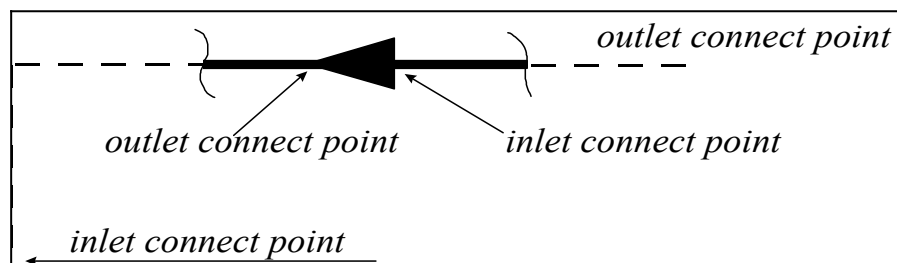
594	567	symbol piping assembly-P1a	represents	348	node C1
595	568	symbol R1a	represents	341	reducer R1
596	569	symbol piping assembly-P2a	represents	342	pipe-P1
597	569	symbol piping assembly-P2a	represents	343	elbow E1
598	569	symbol piping assembly-P2a	represents	344	pipe-P2
599	571	symbol V1a	represents	345	valve V1

Connections at component SYMBOL level

601	570	symbol piping assembly-P3a	has as source	519	connect point ND 1a
602	570	symbol piping assembly-P3a	has as destination	576	connect point 1
603	567	symbol piping assembly-P1a	has as source	576	connect point 1
604	567	symbol piping assembly-P1a	has as destination	577	connect point 2
605	571	symbol V1a	has as source	577	connect point 2
606	571	symbol V1a	has as destination	578	connect point 3
607	572	symbol piping assembly-P4a	has as source	578	connect point 3
608	572	symbol piping assembly-P4a	has as destination	579	connect point 4

Flow arrows and page connectors.

A *flow arrow symbol* is a schematic representation of the flow direction of a stream. The flow direction of a material stream is determined by the way the material is involved in the various processes. For example, material can be involved as output of one unit operation and as input for another. The flow direction should also be consistent with objects that determine or constrain



the flow direction. For example, a pump and a non-return check valve allow only flow of fluid in one direction.

Consistency can be verified by application (software) that might be triggered to derive or verify the consistency of flow direction values, based on the process conditions or on the connected piping branches / segments and in/outlets.

The engineer decides whether the flow direction of a stream is presented by a graphical element (arrow symbol or a 'directed' page connector symbol) on a schematic diagram and where that symbol is positioned. This presentation is supported by the concepts *flow arrow symbol* and *line connector symbol*. The dictionary may include sub concepts of *arrow symbol* to support FROM (normal), TO (reverse) and TWO WAY (undirected or neutral) streams.

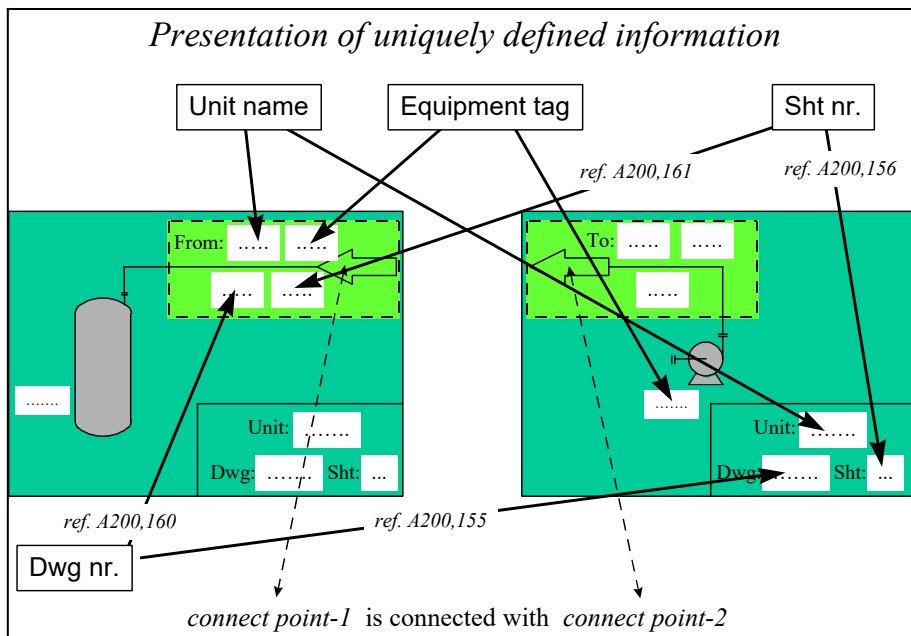
Two opposing flow arrows can be composed to a schematic presentation classified as *undirected flow arrow symbol*.

The text and numbers which appear on P&ID's are also annotation elements which represent information content. They are presented in *text fields* which are positioned on the drawing, often as part of symbol. An 'intelligent' way of assigning the information to the annotation elements is to associate the text fields with the information objects already defined in the ISO standard compliant product database, rather than assigning the information via a CAD drafting user interface directly.

In the following figure and tables is shown how identifications 'P6052', 'U-6000', 'T-1234567' and '1', assigned to the objects classified as 'pump', 'piping system' (ref chapter 6.12) and 'document', 'page' are presented via *text fields* which are part of the annotation element symbols classed as *line on page connector symbol* and *sheet - frame with title block*.

Since on the document 'T-123.456' a drawing number as well as a sheet number is shown, it has to be made explicit in what context that identification is done. The *document* comprises of something that fulfils the role of information *carrier* (paper sheet or computer screen) plus the *annotation* (page or window). A document can comprise of one or more pages. In the example the *document* has a drawing number being the identification of the document 'T-123.456'. The *page* 'T-123.456 sheet 1' has a so called sheet number being '1'.

While discussing the flow arrow symbols it was stated that consistency can be verified based on connected piping branches / segments and in/outlets. To enable to follow a line across drawing pages it necessary to establish a connection relation between the open ended lines.



Identification of sheet within the context of a drawing

UID of fact	UID	Name of left hand object	Name of relation type	UID	Name of right hand object	Validity context
625	613	T-123.456 sheet 1	is a synonym of	613	1	T-123.456

Off page and on page connections

634	138	drawing template 2	is classified as a	610,298	sheet-frame with title block
635	139	template 2, unit name text field	is classified as a	610,426	text field
636	140	template 2, Dwg nr. text field	is classified as a	610,426	text field
637	141	template 2, Sht nr. text field	is classified as a	610,426	text field
638	142	pump symbol tag text field	is classified as a	610,426	text field
639	143	line on page connector symbol-1	is classified as a	610,288	line on page connector symbol
640	144	connector symbol 1,unit name text field	is classified as a	610,426	text field
641	145	connector symbol 1,equipm. tag text field	is classified as a	610,426	text field
642	146	connector symbol 1,Dwg nr. text field	is classified as	610,426	text field

			a		
643	147	connector symbol 1,Sht nr. text field	is classified as a	610,426	text field

Composition of off page and on page connections

645	635	template 2, unit name text field	is a part of	634	drawing template 2
646	636	template 2, Dwg nr. text field	is a part of	634	drawing template 2
647	637	template 2, Sht nr. text field	is a part of	634	drawing template 2
648	638	pump symbol tag text field	is a part of	503	symbol centrifugal pump P6052
649	640	connector symbol 1,unit name text field	is a part of	639	line on page connector symbol-1
650	641	connector symbol 1,equipm. tag text field	is a part of	639	line on page connector symbol-1
651	642	connector symbol 1,Dwg nr. text field	is a part of	639	line on page connector symbol-1
652	643	connector symbol 1,Sht nr. text field	is a part of	639	line on page connector symbol-1

Presentation of off page and on page connections

654	635	template 2, unit name text field	presents	65	U-6000
655	636	template 2, Dwg nr. text field	presents	137	T-1234567
656	637	template 2, Sht nr. text field	presents	138	1
657	638	pump symbol tag text field	presents	64	P6052
658	640	connector symbol 1,unit name text field	presents	418	U-6000
659	641	connector symbol 1,equipm. tag text field	presents	64	P6052
660	642	connector symbol 1,Dwg nr. text field	presents	137	T-1234567
661	643	connector symbol 1,Sht nr. text field	presents	138	1

The next table illustrate that the presented identifications of the physical objects are defined on the line where the object is classified or on a line where an alias or synonym is specified. The preferred name may be denoted by a language community.

Identification of a pump and a piping system

415	64	P6052	is classified as a	130.206	pump
418	65	U-6000	is classified as a	10.431	piping system

For a further explanation of the positioning of the symbols (annotation elements) on the drawing see the paragraph on 'Positioning of Physical Objects'.

10 Modeling of definitions

This chapter provides guidance on how to develop formal taxonomic dictionaries-ontologies that define formalized languages.

10.1 Definitions and definition models

Definitions of concepts in conventional dictionaries usually consist of pieces of text in natural language, each of which defining an isolated lemma or concept that is denoted by a term. Defining concepts in a formal language requires that the definitions are partly or completely transformed into formal expressions. This is possible, because definitions, just like other pieces of information about kinds of things, can be expressed as collections of binary relations between concepts (kinds). Such collections of expressions of binary relations that define concepts are called definition models. Thus:

A definition model is a collection of formal expressions that define a concept.

Each formal expression contains a core that represents a binary relation between concepts. The defined concept forms the center among a number of related concepts in a definition model, whereas the related concepts have their own definition models around them. As a consequence the definition models together form one integrated network of defining relations between concepts.

Thus a formal dictionary can be called a collection of expressions of binary relations between concepts, whereas those relations can be depicted as a network of relations between concepts. Various overlapping cuts out of the network form definition models, so it is also possible to describe a formal dictionary as a collection of overlapping definition models about the defined concepts.

This chapter describes the creation of definition models. It also clarifies the distinctions between the main categories of concepts. Furthermore, it describes how to specify mappings between business specific terminology and a formal language. This chapter finally provides guidance on the use of formal taxonomic dictionaries-ontologies.

The specification of definitions, information and knowledge about individual things is described in chapters 5 - 9.

The specification of knowledge and requirements that is not by definition true for all members of a defined kind is described in chapter 11 and Requirements model respectively.

10.2 Definitions of related concepts

The binary relations between concepts that formally define concepts shall relate only concepts that also are formally defined. This means that modeling of definition or knowledge about a concept should start with the verification whether the related concepts already exist in the formal dictionary. If that is not the case then we should first define those concepts.

The modeling of the *definition* of a concept includes as a minimum the specification of the direct supertype of the concept, together with its full textual definition.



Figure 91, Minimum definition model of a concept

Note that it should be verified whether the supertype concept already exists in the taxonomic dictionary, otherwise the process should be repeated for the supertype concept, until a supertype concept is reached that is already defined in the formal language. A more extensive definition model is described in the following paragraph.

For example, when we express the knowledge fact that “a car can have a maximum speed”, then we make use of the concepts “car” and “maximum speed”, as well as a relation type that is denoted by the phrase “can have a”. All three concepts shall be defined in a (domain) taxonomic dictionary or in an extension.

A concept is defined by a number of facts about the nature of the concept that are always true for every thing of that kind. If one of those facts is not true for an individual thing, then the individual thing is not of that kind or the fact does not define the concept. Note, that for a kind of thing there are often many facts that are normally or typically true, but not necessarily. Such facts do not belong to the definition of the concept, although they belong to the knowledge about what can be the case.

The definition of a concept requires the usage of other concepts. A proper definition requires that also those other concepts are provided with

definitions. This holds especially for the direct supertype of a thing, because each concept shall be defined to be a subtype of its direct supertype. As the definition of the supertype on its turn requires the definition of its own supertype, this is a process that only terminates at the top of the subtype-supertype hierarchy, which consists of the existing top of the hierarchy of concepts that are defined in the formal taxonomic dictionary. Consequently, *every new concept shall be defined as a subtype of an existing concept in the formal taxonomic dictionary.*

10.3 The structure of a formal taxonomic dictionary

Before we describe the development process of a formal language (domain) taxonomic dictionary-ontology (called dictionary for short) we will first describe the structure of such a dictionary.

A formal dictionary for a formalized language has as backbone a taxonomy structure. This means that the defined concepts are arranged in a subtype-supertype structure that forms a hierarchical network of concepts. This hierarchy is illustrated in Figure 92.

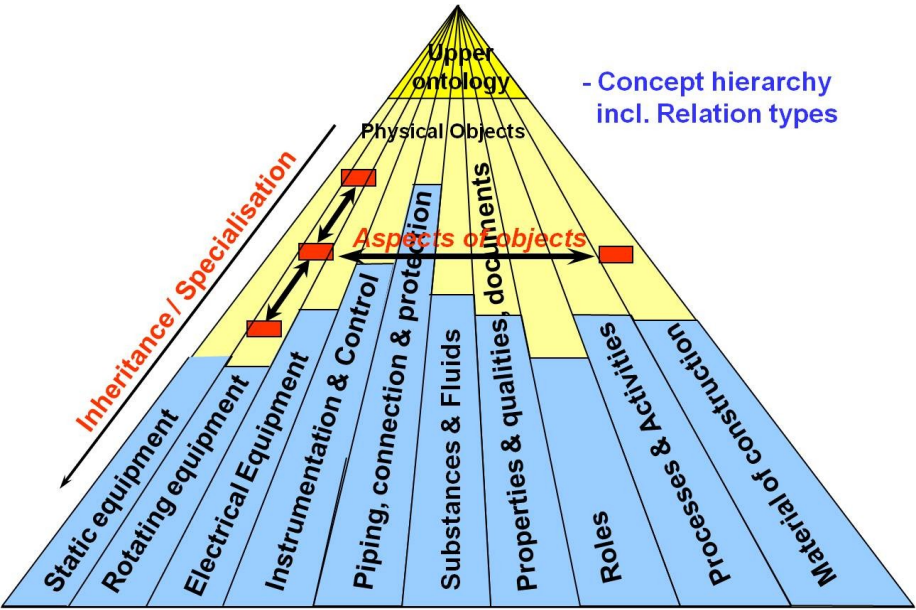


Figure 92, Taxonomic domain dictionaries with a subtype-supertype hierarchy of concepts

The structure of the taxonomy is more than a tree structure, because a concept cannot only have several subtypes, it may also have more than one supertype (parent) concept. The root (or top) of the hierarchy is the concept ‘anything’ from which all concepts are subtypes. The hierarchy consists of several branches or domains. In technology domains this would include for example various kinds of equipment, etc., but it would also cover e.g. biological domains, which would include e.g. plants, animals, cells, biochemicals and diseases. Each of the branches and domains is composed of a collection of related concepts. Each concept is at least related by one or more subtype-supertype relations with its direct supertype(s). This means that the concepts together are arranged in a taxonomy of concepts that are related by subtype-supertype relations. The concepts in the hierarchy are also related to each other by various relations of other kinds, as is illustrated in Figure 92 by the boxes that are related through an arrow. Those relations are classified by kinds of relations (relation types) that are defined in the Upper Ontology section of the dictionary.

Examples of collections of concepts are:

- *Kinds of physical objects* that defines the types of equipment and fluids and components of facilities. Furthermore, kinds of *geographic objects*, *organizations*, *organisms* as well as *documents* and *symbols* on drawings (“ink on paper”).

The collection of kinds of physical objects is sub-divided per discipline, which also form language communities that share terminology.

Examples of disciplines are: physics, business, biology, technology, etc. Such generic disciplines can be subdivided in more specialized disciplines, such as building technology, systems engineering, process technology, mechanical technology, which can be further subdivided in static equipment technology, rotating equipment technology and piping technology, control technology, electrical technology, civil technology, structural technology, transportation, information technology, measuring, etc.

- *Kinds of occurrences*, which includes processes, events and activities; basically all verbs.
- *Kinds of roles* that objects (can) play in relationships and in occurrences.
- *Kinds of aspects* of physical objects, for example: kinds of properties, qualities, states, process materials and materials of construction and kinds for ways of encoding information, such as languages and data formats.

- *Kinds of information and encoding* as recorded on any medium.
- *Kinds of relations* between things ('objects'). These kinds of relations (or kinds of relationships) determine the semantic expression capabilities of the Formal Language. They form the core concepts and their definition provides meaning that can be used to create expressions.

There are three main branches of subtypes of relation. They are kinds of relations that can be used to model:

- *Relations between individual things.* Such relations can be used to describe a real or an imaginary state of affairs.
 - *Relations between kinds* that describe knowledge that is common for the members of the kinds. For example, standard compositions and standard properties of kinds of objects. This includes among others specialization-generalization relations (also called subtype-supertype relations), (de)composition and connectability relations and possession of property relations between the kinds of physical objects and their kinds of aspects.
 - *Relations that relate individual things to kind of things.* The most important of this kind is the classification relation that expresses of which kinds individual things are.
- *Kinds of scales (Units of measure)* that can be used to relate properties to numeric values.

10.4 Domains of concepts (kinds)

As said above, the absolute supertype of all kinds, which is the top of the hierarchy, is the concept '*anything*'. Close to this top the following concepts (or kinds) are defined, each of which is further specialized as far as required for applications in a particular domain:

* *individual thing*

The concept 'individual thing' is the top concept (the supertype) of kinds that are intended to classify things as being individual things. The concept is used in the formal language definition for defining relations that allow for any individual thing as being related by a relation of a particular kind.

* *physical object*

The concept 'physical object' is the sub-top concept for kinds of things that satisfy the laws of physics. The subtypes include solid item, fluid, wave, space, geographic object, signal, organization, person, organism, etc. and their further subtypes.

Subtypes of physical object might not only belong to the category of physical things, but some of them may also belong to other categories, such as social things.

* *occurrence (activity)*

The concept 'occurrence' is the sub-top for all kinds of processes, activities, events and happenings that occur, but also of static occurrences such as support, storage, etc.

* *role*

The concept 'role' is the sub-top for kinds of roles of things, including roles of physical objects, roles of aspects, especially intrinsic aspects and roles of other things. Typically roles of such kinds are played in relations of particular kinds. For example, it includes kinds of roles of physical objects in occurrences (functions).

* *aspect*

The concept 'aspect' is the sub-top for *characteristic* and *property* (physical quantities), *quality*, *state*, *phase* and '*substance*' and all their detailed subtypes. Aspects are mainly intended to characterize physical objects and activities. Aspects are distinguished in *conceptual aspects* and *qualitative aspects* (which includes also quantitative aspects). Examples of conceptual aspects are color and pressure. Examples of qualitative aspects are red, green, blue, 5 bar and 3 psi.

Another subtype of aspect is *encoding aspect*. This is a sub-top of kinds patterns according to which information can be encoded. It covers for example languages, kinds of identifiers and all kind of electronic coding formats. It concerns the 'what' of the information, usually described as text, whereas the 'how' is recorded or presented as a physical object (often ink on paper or light on a screen).

* *information*

The concept 'information' represents 'meaning' irrespective of the way in which the meaning is expressed. For example, a piece of information is irrespective of the language, the encoding system and the format that is used to express the information. Thus the UID of such a piece of information is the same for various synonymous expressions.

Information is distinguished in conceptual information and qualitative information. Subtypes of conceptual information are e.g.: document (content), definition, instruction, memorandum, etc. Qualitative information consists of qualitative subtypes of those kinds, such as particular definitions, and other pieces of information. Typically qualitative information is written in natural language or in other coding systems (such as a programming language) and may not be in a formalized form. In the latter case it is not expected to be interpreted as a formal language. Note that names of things are treated separately.

* *relation* ('is related to')

The concept 'relation' is a sub-top concept for kinds of relations. It is further specialized into a large number of (standard) relation types that define the semantics of the formal expressions. Examples of subtypes of relation are: classification relation, assembly relation, specialization of concept, possession of aspect, etc., which are also denoted by phrases such as <is classified as>, <is a part of>, etc.

Figure 93 illustrates a selection of concepts from the top of the Gellish taxonomic dictionary. It illustrates the specialization hierarchy with examples of further subtypes of the above kinds (the actual formal hierarchy is slightly different).

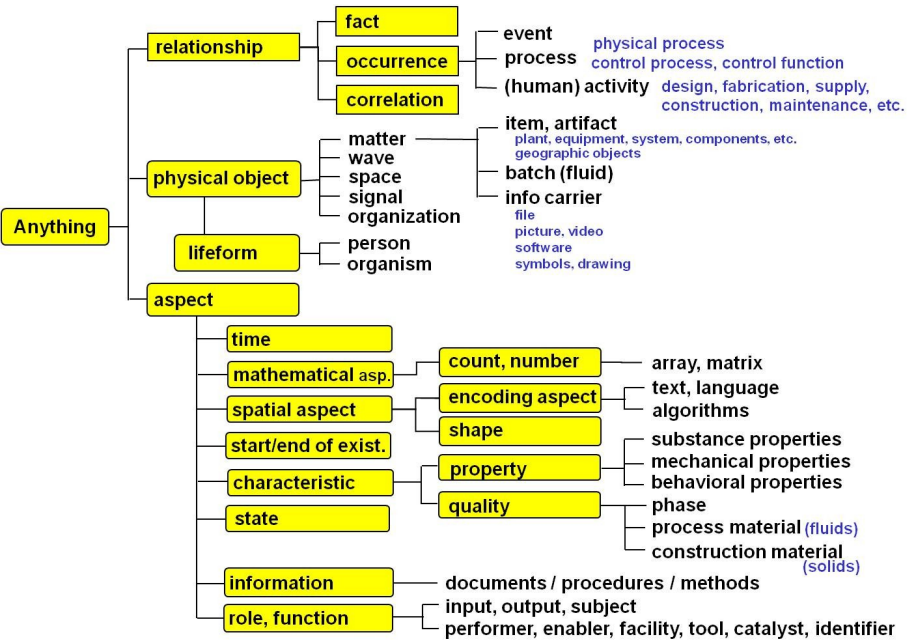


Figure 93, Top structure of a taxonomic dictionary

10.4.1 Collections of kinds

Formal languages are suitable to define not only generic concept, but also specialized product types as found in standards as well as manufacturer specific product type, often called manufacturer's models, as mention e.g. in product catalogues. Figure 94 illustrates how those concepts relate to each other.

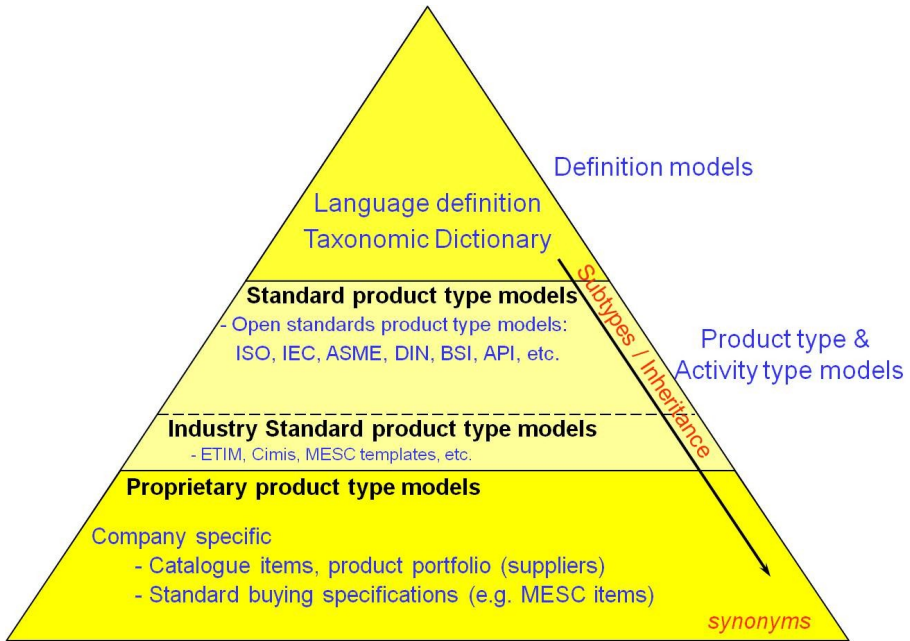


Figure 94, Categories of kinds

The top of Figure 94 illustrates that there is a collection of kinds that can be regarded as *base concepts* that are largely defined in ISO standards, whereas there are also collections of kinds below that which are (standardized) *product types*. The latter kinds have definition models (product models) that are models of standardized product types. Such product models are characterized by the fact that the product types are defined among others by their qualitative aspects (fixed aspect values). Furthermore there are proprietary kinds of things that are typically owned by companies and are known as company specific *catalogue items* or *proprietary manufacturer's models*, which may be defined by product type models, which typically include sizes and materials of construction.

The *base concepts* form the basic section of the dictionary. They are not only the above mentioned generic kinds, but also kinds with names such as pipe, pump, floor, etc., without a specification of their aspects or aspect values. For example, a pipe can have as aspect a diameter, but the concept is not defined by a particular diameter (aspect value) and not even by the fact that in general it will have some diameter. Base concepts includes also kinds which names are composed of more than one word (at least in English), such as centrifugal pump, long radius bend, etc. The base concepts also include the relation types, such as classification ('is classified as a'), composition, collection, connection and possession of aspect relations.

Base concept are usually not specialized on the basis of a material of construction, because all possible combinations between each kind of physical object and each possible kind of substance would lead to a combinatorial explosion of the number of kinds.

For example, 'stainless steel pipe' is not a recorded base concept, although it may be recorded as a standardized subtype (e.g. because there is an ASME standard about it).

Note: an individual item can always be classified as pipe together with a material of construction as follows:

P1	is classified as a	pipe
P1	is made of	stainless steel

Formal English has some special subtypes of 'kind', called '*conceptual kind*' and '*qualitative kind*'. Those concepts classify kinds and not individual things. A conceptual concept is a concept that has no defined extent or magnitude, so that it is not qualitatively or quantitatively characterized. A qualitative concept is a concept that has a defined extent or magnitude and thus it is qualitatively or quantitatively characterized. For example, the concepts 'color' and 'temperature' are unqualified conceptual kinds, whereas the concepts 'red' and '37 degree C' are qualitative kinds.

Another concept is called '*plural concept*' or '*collection of concepts*'. This concept and its subtypes are intended to classify collections. For example, 'ASME classes' is a particular collection of kinds that is not defined on the basis an aspect of individual items, nor of the kinds themselves, but only on the basis of the fact that the ASME organization has defined each class.

Collections of kinds are useful for administrative reasons to collect kinds on the basis of their origin or custodian or on the basis of a prime discipline that defines or primarily uses them.

10.4.2 **Standardized product types and catalogue items**

The category of *standardized product types* actually does not consist of just kinds, but they are extended with ‘structures’ (models) that consist of collections of related expressions that express facts about the kinds. This category is formed by the kinds with relations that describe the objects that are standardized by standardization bodies, such as ISO, IEC, ANSI, DIN, BSI, etc.

These standardized product type models are defined by using many more types of relationships such as the possession of specific property values, material of construction, fabrication standards, etc. Therefore, their definition requires a much deeper understanding of Formal English. The necessary information for that is not given in this guide, but should be obtained from the full Definition of Formal English, or from other documentation, such as the Parts 3 about knowledge modeling.

Typical examples of collections of such kinds for standardized items are defined and published by standards bodies. For example, standard pipes as defined by ANSI. Such collections are and can be included in domain specific dictionary extensions.

Note: an example of such a kind is a specialization of pipe and can be documented in Formal English through the explicit relationships that define its aspects, such as its nominal diameter, schedule, fabrication standard, etc. by their qualitative values.

For example, a pipe with a nominal diameter of 6 inch can be defined as follows:

101	6" pipe	is a specialization of	10077	pipe
101	6" pipe	has by definition as intrinsic aspect a	102	ND6pipe
102	ND6pipe	is by definition an intrinsic	551563	nominal diameter
102	ND6pipe	is by definition an intrinsic aspect of a	10077	pipe
102	ND6pipe	can be quantified on scale as equal to	920106	6 inch

In other words, the name ‘6" pipe’ is only a name for such a kind. Usually a computer cannot derive property values from such a name.

Company specific product types and their definition models, also when they are ‘standardized’ within a company, should be modeled according to the same methodology as the kinds defined by standardization bodies. However, company specific kinds are proprietary kinds and normally they will not be included in the standard Definition of Formal English. However it is possible to publish the product type UIDs, names and specialization relations, without necessarily publishing the detailed product type definition models. Those models can be developed, maintained and managed by companies, whereas the product types can be published as ‘proprietary extensions’ of the Gellish formal languages. Those kinds can then be used in data exchanges with business partners, such as in the context of projects.

Formal English is intended also to support the description of such proprietary product models. Data exchange would be simplified significantly when *standardized product types* and (*buyers and suppliers*) *catalogue items* would become available in Gellish Database form.

However, the definition of those product models is the responsibility of *standards bodies* and of individual companies.

10.4.3 Kinds of relations

Among the concepts in all categories there are not only specialization relations, but also many other kinds of relations. For example, for a kind of physical object there are different relation types that should be used to define what its properties are, what its normal composition is, which roles it can play, etc.

Those different kinds of relations are required for expressing the differences in meaning (semantics) of such relations, especially because the relation types determine the constraints that specify which kinds of things are allowed to play a role in relations of a particular kind.

The definitions of such constraints are inherited by subtype of the kinds of relations. Therefore, a formal dictionary includes a hierarchy of kinds of relations, also called relation types.

Kinds of relations include:

- * Relation types that relate kinds. They express knowledge about possible relations between individual objects. This includes the definition of the semantics of the formal language, as well as ‘product models’ that express social, business, technological and scientific knowledge.
- * Relation types that can be used to classify relations between individual objects. They express real or imaginary facts about individual objects.

- * Relation types that can be used to classify relations between individual objects and kinds. They express knowledge about individual objects. For example, the fact that an individual object belongs to a certain kind or the fact that it can be used for a kind of purpose.

The application of these standard relation types has as a result that the Gellish knowledge base contains a network of relations between concepts that documents knowledge about the concepts. Each concept acts as a node in the network of relations. A ‘cloud’ of relations around such a node in the network defines a “knowledge model” that specifies the aspects that are normally true for a ‘well formed’ object that is classified by the concept. Figure 95 illustrates such a knowledge model by specifying knowledge about a vessel.

520243 vessel	can be a part of a	160104 process unit
160177 material	can have as aspect a	550020 mass
550020 mass	can be expressed on scale	570039 kg
520243 vessel	can have as aspect a	550102 design pressure
550031 pressure	can be expressed on scale	570393 barga
520204 shell	can be a part of a	520243 vessel
520204 shell	can have as aspect a	550206 outside diameter
550188 diameter	can be expressed on scale	570423 mm

- blue text = Gellish standard relation type
- purple text = Gellish standard concept
- red text = Gellish standard unique identifiers

Figure 95, Example of a knowledge model with relations between concepts

The book ‘Semantic Information Modeling in Formal Languages [Ref. 2] contains an extensive discussion of definitions of the taxonomy of kinds of relations.

10.4.4 Composition and connection relations between members of kinds

The definitions of kinds of relations can be illustrated for example on composition relations and connection relations.

Members of kinds of things *can be composed of members of other kinds of things*. Such statements are conceptual composition relations between already defined concepts. Such statements do not define additional kinds.

For example, the statement that an impeller can be part of a centrifugal pump is the case for every member of the kind of centrifugal pump. Such possible composition relations between kinds are typically recorded in knowledge bases. Collections of such conceptual composition relations result in composition hierarchies that specify the knowledge how members of a concept can be or are normally composed. For example:

centrifugal pump	can have as part a	impeller
rotating equipment system	can have as part a	centrifugal pump

Normal compositions can be constrained by so called (simultaneous) cardinality constraints. A cardinality constraint specifies the minimum and maximum number of things of a particular kind that simultaneously can be related to one exemplar of the other kind. For example, the number of impellers that is simultaneously part of a centrifugal pump normally varies from 1 to n, whereas an impeller can be simultaneously part of only one centrifugal pump or an impeller can be not a part of any centrifugal pump at

all, for example when an impeller is in stock. Thus the pump has a cardinality range from 0 to 1 in this relation. When these cardinality constraints are made explicit, the definition becomes:

0,1	centrifugal pump	can have as part a	1,n	impeller
0,1	rotating equipment system	can have as part a	0,n	centrifugal pump

From the example relations it can be concluded that a centrifugal pump always has at least one impeller. If that is indeed the case, without exceptions in other contexts, then it can be concluded that ‘having an impeller’ is part of the definition of the concept ‘centrifugal pump’. This means that it is not only knowledge that it *can be* the case, but there is also true that it *is by definition* the case and forms part of the definition model of the concept ‘centrifugal pump’. This is expressed as follows:

0,1	centrifugal pump	has by definition as part a	1,n	impeller

Note that a rotating equipment system does not need to have a centrifugal pump as one of its parts. Thus it is not defined by such a relation. If in a particular context it holds that a rotating equipment system nevertheless always shall have a centrifugal pump as one of its parts, then that should be specified by using a relation of a kind that specifies a requirement. This can be done as follows:

0,1	centrifugal pump	shall have as part a	1,n	impeller (in context X)

The fact that a particular impeller I-1 is indeed a part of a particular centrifugal pump P-1 should be expressed by using yet another kind of composition relation, as follows:

I-1	is a part of	P-1

The reason that yet another kind of relation should be used is the following: A common constraint for the above kinds of composition relations is that they *relate kinds*, whereas for expressing that I-1 is a part of P-1 a relation is required of a kind that *relates individual things*.

Note that cardinality constraints are not applicable for relations between individual things.

Conceptual connection relations specify the things of particular kinds can be *connected to things of other kinds*. For example, a nozzle can be connected to a vessel. Conceptual connection relations are also relations between already defined kinds, similar to the relations that express possible compositions. Note that this example is not a defining relation, because if a vessel is not connected to a nozzle, it is still a vessel.

10.4.5 Kinds of physical objects

A physical object is a (real or imaginary) individual thing that consists of matter and/or energy that is distributed in time and space and that satisfies the laws of physics. All kinds of physical objects are subtypes of the concept 'physical object'. This concept is the top of the hierarchical network of specializations of kinds of physical objects. Man-made kinds of physical objects are subtypes of further subtypes of physical object, such as artifact, which is a subtype of solid item. Batches of fluids (liquids and vapors) are also physical objects. The materials, from which solid items or batches are 'made', such as steel and water, qualify aspects of physical objects and are themselves not physical objects.

To design facilities, systems and products in detail and to operate and maintain them, it is required to include concepts in a formal dictionary for all relevant kinds of physical objects that are used during the lifecycle of those facilities and their components.

Kinds of physical objects intent to define the commonalities in nature of things are, taking into account the physical aspects of the objects. For artifacts those aspects are the ones that are intended to make them suitable to fulfill their intended role.

Every kind of physical object should be defined on the basis of one or more aspects that the members of that kind have in common or on the basis of one or more kinds of components of which the kind of physical object is composed. A criterion for a proper kind of physical object is, that even if such an object would not be used for its intended role (e.g. when it is stored in a warehouse), then still it can be concluded from its aspects or components that it is a member of that kind.

For example, an eccentric reducer is a kind of physical object that is a specialization of the concept 'reducer', and is defined on the basis of a shape aspect.

The types of aspects that are most frequently used to distinguish kinds of physical objects are:

1. Intended role or function - e.g. heat exchanger

This means that the object is intended to be suitable for an intended activity or process and that implies consequences for the physical aspects of the object. Thus subtypes on this basis are not kinds of roles, but are (normally) physically shaped such that they can perform such roles. They are even members of the subtypes when they are off-spec products that do not (completely) satisfy the criteria for membership,

because they were intended to satisfy those criteria. For example a batch of off-spec petrol is still classified as petrol.

- 2. Qualitative characteristic - e.g. vertical submergible pump
- 3. Quantitative characteristic - e.g. 6 mm bolt
- 4. Material of construction - e.g. stainless steel pipe

However, it is recommended that physical objects specializations according to material of construction are not included in the dictionary (except for special cases such as 'glass fiber'). Kinds of material of construction or 'substance' are kinds of aspects that are defined in a separate hierarchy as specializations of 'stuff' or atomic and sub atomic structure.

- 5. Shape - e.g. sphere (spherical vessel), tree
- 6. Composition - e.g. shell and tube heat exchanger
- 7. Principle - e.g. pitot tube (which implies shape)

Examples of types of aspects that are used but that indicate a role rather than an intrinsic aspect are:

- 1. Location - e.g. driven end bearing
- 2. Condition - e.g. waste

The resulting kinds usually receive their names from one or more of those aspects. However, a concept usually implies the definition of more than only one aspect. A concept may derive its name from only one aspect, but its definition usually includes more than one aspect which makes it distinct from its parent class. Furthermore, it may derive its name from an intended role or function, but the physical aspect that really makes the distinction is the design consequence of that.

For example: the concept 'sphere' (in the meaning of 'spherical vessel') derives its name from its shape, although it also implies an intended activity of 'storage'.

Example 2: the concept 'heat exchanger' derives its name from its intended role in an activity; although to execute that activity certain shape and properties are required such as a surface area and a thermal conductivity.

Example 3: the concept 'shell and tube heat exchanger' derives its name from a combination of an intended role and a composition aspect.

The fact that a kind of physical object *derives its name* from a specific aspect has not necessarily an impact on what the concept **is**. The aspect from which a concept derives its name may even be language dependent. For example, a 'gear' in English is intended to gear, but it is called a 'tandwiel' in Dutch, which is a classification according to shape or composition (dented wheel).

Many kinds of physical objects are designed and/or fabricated suitable for a particular kind of role and therefore those kinds are named according to that *intended* role. In order to be suitable for that role it should be recognized that this intended role implies consequences for the physical aspects of the objects, such as their shape. For example, assemblies of equipment such as process units are usually dedicated designs for a specific role. This does not imply that the concept 'process unit' only defines activity or functional aspects, because it has shape consequences that makes the thing suitable to fulfill that role.

Kinds of physical objects often have names that are derived from their intended role. For example a pump derives its name from its intended role to pump (of made for the role of "pumper"). Nevertheless it is clear that a pump is a kind of physical object. The same holds for a blower, which is a type of compressor.

However, "pure" roles are independent of the type of physical object that performs the activity. If English knew the word "pumper" then it would have been a kind of role, because even a person could be in that role.

The problem in natural language is that we often use the same name for a kind of physical object and a kind of role. The term blower is a good example. It is not something that can blow (e.g. a person), but it is a special type of machine that is made to blow.

The general rule is in those cases that we define the kind of physical object "blower" and we only define a separate kind of role "blower" if there are clear examples in industry that other types of equipment can play that role, without being named after that role.

Kinds of physical objects should not be defined on the basis of the role the object *incidentally* plays. Such a concept typically is a *kind of role* (see below). A handy check is to think of such an object being located in a warehouse, out of its normal context, where it is in a place or position that it cannot fulfill its normal role(s). If you are then still able to conclude about its classification by such a kind, then it is a kind of physical object, but if you cannot draw conclusions about a classification by such a kind in such a context, then the kind is probably a kind of role.

The following kinds often belong to this category:

- *Components*, i.e. kinds that normally have a name that ends on ‘component’. They are normally kinds of roles that indicate that an item normally is part of an assembly of a kind which name often forms the other part of the name of the role .

For example, equipment component, pump component, control system component, piping component, etc.

Exceptions are kinds where the term ‘component’ in its name can be replaced by ‘item’. For example, the term ‘pneumatic component’ denotes items that are pneumatic, irrespective of being parts. Therefore the term can be defined as a synonym of pneumatic item, which is a kind of physical object.

- *Material*, i.e. kinds that normally have a name that ends on ‘material’. For example, connection material, insulation material, welding material. If the term ‘material’ indicates that items of that kind can be used for a particular kind of purpose or for a kind of usage in an activity, then the kind probably is a kind of role. For example, the kind of role ‘connection material’ states that some material is intended to be used for connecting things, but it does not specify what makes such things suitable for that purpose.

Exceptions are kinds of which its members do share an aspect that makes the members suitable for that purpose. In those cases the concept is a kind of physical object that classifies things by that particular subtype of physical object.

The term material in the name of a concept may also indicate that the concept is intended to classify collections. In formal languages the elements in a collection can be classified by a singular term, because the relation type definition specifies whether a single or plural object is involved. Thus there is no need to define plurals of kinds for the classification of elements in a collection. For that reason, the term material in the name of a concept may be superfluous. For example, the term ‘plate material’, may denote a concept that classifies collections, each of which elements is classified as a plate. In such a case the term in plural (‘plate materials, with an s) may be justified. However, for unambiguous modeling it is sufficient when the collection is just classified as a collection or as a ‘collection of items’, because the classification of the elements in the collection already specifies their kind. Thus practically plate material is a synonym of plate.

The concept ‘material’ itself is also an exception, because it defines a kind of physical object whose members have a specific quantity of mass.

- *Equipment*, i.e. kinds that normally have a name that ends on equipment. If the term equipment is not preceded by an aspect of the equipment, but by a purpose, then the concept is often a kind of role.

For example, connection equipment, protection equipment, fire-fighting equipment.

The term ‘equipment’ typically has the connotation of being a collection of piece of equipment or equipment items. Therefore, the dictionary typically uses as name a phrase that ends with ‘equipment item’.

Examples of proper kinds of physical objects which are defined on the basis of a common aspect of its members are therefore: rotating equipment item, electrical equipment item.

The number of roles that physical objects *can* play may be unbounded. For example, a screw driver can play a role as hammer. These ‘possible roles’ of objects (in activities) should be distinguished from the ‘fit for purpose’ classification which defines what a thing *is*.

A proper kind of physical object classifies objects that have *a common intended role which has design consequences*, i.e. what a concept *is*, not objects with common incidental possible roles or actual roles.

The following list indicates domains of kinds of physical objects that are included in the formal dictionary:

1. Plants, Systems and generic items, including spaces
2. Rotating equipment and their components
3. Heat generation and heat transfer equipment and their components
4. Static equipment, especially vessels and their components, including columns, reactors and storage items
5. Instrumentation and control systems and their components, including IT
6. Electrical systems and their components
7. Piping systems and theirs components
8. Valves and their components, both in-line and control valves
9. Protection systems and materials
10. Connection materials
11. Transport and movement systems and their components, including loading product components
12. Infrastructure, civil and steel structures, buildings and furniture
13. Heating, ventilation and air conditioning (HVAC) systems and their components
14. Solids handling systems and their components
15. Geographic items, such as oil fields, wells and their components, areas
16. Organisms, persons and organizations
17. Computers and telecommunication items

18. Waves, such as nuclear radiation, radio waves, light, sound, etc.
19. Signals, such as control signals, digital signals, etc.
20. Document files, such as drawings, forms, reports, messages and manuals.
21. Software.
22. Symbols and their components (graphical elements).

10.4.6 Kinds of roles

Some terms denote kinds, which people may intuitively think of being kinds of physical objects, although they are kinds of roles of physical objects. For example, a 'driven end bearing' is not a kind of bearing, because it is just an ordinary bearing, which is classified according to its location. But that location is not a permanent or intrinsic aspect of the bearing, so that it is a 'non-permanent' aspect. When the bearing is taken out of its usage context, for example when it is located in a warehouse, then it is not possible to determine this type of concept on basis of the aspects of the bearing. Conversely a "thrust bearing" is a valid kind of physical object because it is specifically designed to be able to withstand axial loads as well as radial loads.

Thus, there are kinds that may seem to be kinds of physical objects, but actually they are kinds of roles. This also holds for aspects. For example, the concept bottom temperature does not denote a special kind of temperature, but it denotes an ordinary temperature, which is measured at the bottom of some item. Thus bottom temperature is a role of a temperature.

Roles of things are usually temporal or location dependent. Something can be classified by its kind of role, but if it does not have such a role at a certain moment, then it is not correct anymore to maintain its classification as a member of such a kind.

The following test can help to determine whether a kind that needs to be defined is really a kind of physical object or whether it is a kind of *usage* or *role* of a physical object. Imagine that an individual thing of the kind needs to be classified while it is located in a warehouse, where it does not fulfill its intended role. If it is then possible to determine that it is classified by the kind that need to be defined, then the kind is a kind of physical object. If that is not possible, then the kind is a kind of role.

Kinds of roles have definition models that differ from definition models of physical objects. Thus kinds of physical objects must be distinguished from kinds of roles and roles of particular kinds can be or are by definition played by things of particular kinds. There may or may not be objects of specific

kinds that are especially designed or fabricated for fulfilling roles of particular kinds.

A kind of role for physical objects can often be defined by an <is by definition a role of a> relation with a kind of physical object and the kind of role is defined to be a subtype (specialization) of 'role' or e.g. 'usage' or 'located physical object'.

For example,

- driven end bearing is by definition a role of a bearing
- driven end bearing is a specialization of role
(being a role of a bearing when located at a driven end of a shaft).

Note 1: There are tricky cases. For example, it might be expected that 'fire fighting pump' is a role only. It is interesting to note that this 'intended use' appears in practice to have mechanical consequences and thus this concept name indicates functional aspects as well as mechanical aspects and therefore it is a special kind of (subtype of) pump and thus a kind of physical object.

Note 2: There are exceptional cases where a kind of physical object should be distinguished from a kind of role with identical names in natural language. For example, a temperature transmitter is a specialization of transmitter (an instrument), because there are specifically designed transmitters for the role of transmitting temperature. But there are also general purpose transmitters which can play a role as temperature transmitter. This means that we need two distinct concepts: temperature transmitter (item) and temperature transmitter (role).

The relations between them can be expressed as follows:

- temperature transmitter is a specialization of transmitter
- temperature transmitter (role) can be a role of a temperature transmitter
- temperature transmitter (role) can be a role of a general purpose transmitter
- general purpose transmitter is a specialization of transmitter

It seems trivial that a material can play the role for which it was designed. However, it can be valuable information to specify for which kind of activity or process some kind of thing is designed. Therefore it is usually not recorded that something of a particular kind is designed for a role of a particular kind, but it may be valuable to express that it is designed for performing a particular function, being a kind of activity or process. For example, it will have little value to express that a temperature transmitter (item) can have a role as a temperature transmitter (role), but it is valuable information to express that it is intended to transmit signals, because such

information can be used to support a design process or to verify the consistency of a model about transmission. A defining relation thus might become:

transmitter has by definition as intended function transmitting a signal

Thus, kinds of roles include also:

- kinds of functions, being roles of activities (occurrences).
- kinds of roles in relations, including also roles in correlations.

10.4.7 Identification of new kinds and their hierarchy

The easiest way to determine relevant kinds is to decompose a plant, system, piece of equipment or other physical object into all types (kinds) of components.

For example: a pump can be decomposed into a *bearing*, a *shaft*, etc. This identifies the kinds *bearing*, *shaft*, etc. and it identifies common composition relations between the whole (class) and its parts kinds, which can be recorded as:

bearing *can be part of a* pump.

The next step is to identify sub-types and supertypes of those components. For example, the sub-types *ball bearing*, *roller bearing*, etc. and the supertype *rotating equipment component*. These result in specialization relations which can be recorded as:

<i>ball bearing</i>	is a specialization of	<i>bearing</i>
<i>roller bearing</i>	is a specialization of	<i>bearing</i>
<i>bearing</i>	is a specialization of	<i>artifact</i>

So, objects can be classified and object *kinds* can be themselves defined as being subtypes of higher level kinds. (See figure 1, entity: 'specialization of a kind of object').

For example the object concept 'axial centrifugal pump' is a concept (type) of centrifugal pump, which on its turn is a specialization of pump, which is a specialization of equipment item, etc.

Such specialization of kinds result in *specialization hierarchies*. Such specialization hierarchies are required in order to distinguish between data that is valid for all types of pumps from data that is valid only for a specific type of pumps, e.g. only for axial centrifugal pumps.

Specialization hierarchies are important among others for selection and comparison of performance characteristics of similar equipment and equipment components.

Formal English allows that any concept can be specialized in an unlimited number of more specialized kinds as and when required.

10.4.8 Kinds of properties, qualities and encoding aspects

The dictionary also contains ‘kinds of properties’, ‘kinds of quality’, ‘kinds of encoded information’ and allowed units of measure. The knowledge base also contains relations between those kinds and the kinds of objects.

Properties are quantifiable characteristics, such as pressure and diameter. *Qualities* are characteristics that are qualifiable by qualitative values, but generally not quantifiable. Examples of qualities are: openness, corrosiveness, etc. Qualities are typically qualified by qualitative values. Examples of qualitative values are: open, closed, corrosive, non-corrosive, etc. Qualitative values are ‘qualitative subtypes’ of (conceptual) qualities. Qualities are usually qualified in user interfaces by textual values that are selected from pick lists or they are represented by yes or no in answers on yes/no questions.

Encoding aspects are the information bearing aspect of physical text, sound that is stored as a bit pattern in a computer, which is encoded according to some coding rules, such as a natural language, a coding standard such as ASCII, Unicode or HTML as used by the Internet or an application specific encoding, such as .DOC as used by MS-Word.

An individual property can be classified by a conceptual property or qualified by a qualitative property. Thus qualitative properties are property values that can be used to qualify individual properties. Qualitative properties are ‘qualitative subtypes’ of conceptual properties. Qualitative values are sometimes called domain values. Qualitative properties should not be confused with numbers on a scale. This is illustrated by the following example.

For an individual possessed property D1 we may state:

D1	is classified as a	diameter
D1	has on scale a value equal to	300 mm

In this example the concept ‘300’ is a qualitative value, which is a qualitative subtype of number. Instead of qualifying D1 on a scale we could qualify it by a qualitative property value, using the concept ‘300 mm’. In that case the second expression would not be a quantification by a number on a scale, but it would be a qualification as follows:

D1	has on scale a value equal to	300 mm

In the latter case the value '300 mm' is a qualitative subtype of distance (and not of number on a scale), whereas distance is a subtype of property. The relation between the qualitative value '300 mm' and the number '300' is that '300 mm' is defined as having a magnitude of 300 on a scale of mm.

The formal dictionary includes numbers as well as qualitative properties (properties with values) for kinds of properties, qualitative qualities and standard kinds of encoded information.

The taxonomic dictionary contains relations between kinds of properties, qualities and encoded information and the kinds of objects for which they are recognized to exist or for which they are definitional.

For example, the taxonomic dictionary contains a relation that says 'diameter' is a common kind of property for 'vessel'. These are defined as relations between a kind of property and a kind of physical object.

So, kinds of properties and their relations to kinds of objects include:

- mechanical characteristics, such as diameter, hardness,
- shapes, such as cylindrical shape, etc.
- fluid properties, such as density, viscosity, etc.,
- duty characteristics, such as capacity,
- design limits, such as design pressure,
- units of measure,
- states, such as phases (liquid, vapor, etc.), open, closed, toxic, corroded, etc.

Kinds of encoding aspects and their relations to kinds of objects include:

- languages, standard data formats
- procedures, methods and algorithms, etc.

10.4.9 Kinds of aspects of physical objects

Kinds of physical objects are always defined on the basis of one or more aspects that their members have in common. For example, a centrifugal pump is a specialization of a pump on basis of the fact that all its members apply the centrifugal principle. So 'centrifugal' is an aspect that qualifies centrifugal pumps.

Specialized kinds of physical objects can often be defined completely by the fact that they possess one or more aspects that are the basis for inclusion or exclusion of members in the class. This may mean that they are defined sufficiently by the definition of those aspects, without the need for an additional textual definition. So, specialized kinds should be defined by the explicit definition of their possessed aspects. They might not even need a name, but only an identifier that is unique in a specific context. This is especially the case when the name of the concept can be derived from a concatenation of its aspects.

For example, a special type of flange (= a kind of flange) defined by a specific buying description might be identified only by a company specific code, such as a KKS code. It does not need a name and it can be selected from a search on one or more of its aspects, including its parent kinds.

To select a 'standardized object' an engineer typically starts with a few *properties* of the specific object that he is designing, based on the requirements of the process or activity that should be performed. Then a 'standardized object' (= a specialized kind of physical object) is selected which has *properties* that fulfill the required properties sufficiently. Then the specific object can be classified to be of one of that standard class.

However, it should be realized that classification of an object may result in a situation that one or more of the properties of the specific object don't get a value assigned to it, but classification means that its properties are within the constraints on the property of the class.

The generic kinds will not include kinds that are defined by values of their properties. For example, a 6 inch valve is not a generic kind. There are a few exceptions on this rule: whenever kinds are generally recognized in practice, then they should be included in the general dictionary, even though they might be defined completely by their explicit aspects and even if their textual definition might be left away.

On the other hand, the definition of such kinds should be limited to kinds that are already in use in normal engineering practice, because theoretically it is possible to think of an unlimited number of those kinds.

For example: 'single stage compressor' is a relevant subtype of compressor, because it is generally recognized in industry, even though a specific physical object can be described also without that kind, such as by classification as 'compressor' with a number of stages = 1.

The aspects of physical objects also include quantification of their magnitude and qualification of their shape. Therefore kinds of mathematical and geometric items are included, such as:

- geometric models and their components
- mathematical models.

10.4.10 Kinds of substances - process materials & materials of construction

Physical objects are made of types of matter. The atomic and subatomic structure of matter is often referred to as its 'type of material' or its substance. Kinds of these define the types of atoms and molecules and the way in which they relate in a mixture.

For example, an impeller is made of stainless steel. Then the kind 'stainless steel' only refers to such a structural aspect. This fact is to be distinguished from the statement that an impeller is made of a specific batch of material. (The structural aspect of) the batch can also be classified as stainless steel.

Thus 'stainless steel' is an example of a kind of substance (usually in the role of *material of construction* for a physical object). This means that objects that are made of some kind of substance will have a relation with that kind of substance.

For example: physical objects *can be made of* stainless steel

Therefore, 'material of construction' is not defined as a characteristic of an object, but as a classification relation between an aspect of an object and a 'kind of substance'.

Process material like construction material, is also a role of a material. So, oil, water, butane, etc. are other examples of instances of 'kind of substance'. They have their own hierarchy:

crude oil	is a specialization of	oil
stainless steel	is a specialization of	steel

Process materials can be classified also according to their state or phase. For example: material can be in a fluid state, which state can be specialized further into liquid, vapor, solid, plasma, etc. or its state can be crystalline, polymorph, etc.

So, kinds of substance include:

- solid, such as steel, plastics, concrete etc. mainly used as material of construction

- fluids, such as oil, chemicals, water, air, etc. mainly used as process material
- physical elements and their constituents, such as H, He, etc., electrons, nuclides, etc.

10.4.11 Kinds of signals, waves and energy

Signals and waves are subtypes of physical object. Energy is defined as a *property*. They have a classification hierarchy which starts as follows:

signal	is a specialization of physical object
electrical signal	is a specialization of signal
radio signal	is a specialization of signal
wave	is a specialization of physical object
light	is a specialization of wave
energy	is a specialization of property
etc.	

Signals are modulated streams of physical objects, from which the modulation properties are representing the carried information.

Waves are physical objects, which can move in matter or can be present in the form of radiation. Waves represent or ‘imply’ *energy*, such as heat energy, that can be contained by a material. Energy is considered to be an aspect (a property) of a material. Energy can be extracted from a material and transported through another material.

For example, *heat* can be extracted from a fluid by transportation through the walls of pipes in a heat exchanger and transferred into another fluid.

Electro-magnetic waves are physical objects that can move in vacuum, but can also move through another physical object, for example in the form of *electricity* which moves through a conductor. Electrical energy can also be possessed by a physical object, such as electrical charge which is stored in an accumulator (such energy is also seen as a property of that physical object). It can also be extracted from the physical object and transported through a conductor such as in a wire.

For example, electricity and light waves are used to carry a physical representation of *encoded information* (e.g. a message) for control purposes, but they can also be used to transport energy.

10.4.12 Kinds of occurrences

Kinds of activities and processes include:

- human activities and business processes

- industrial processes and unit operations, such as distillation, heat transfer, etc.
- natural processes, such as corrosion
- control processes (functions) and electrical processes, such as conversion of signals
- events

10.4.12.1 Activities and Events

Occurrences are things that take time and may change over time. Activities are things that happen or occur. So an activity is a subtype of occurrence. In language terms, kinds of occurrences are expressed by the verbs in the language. In the formal language the active form, the passive form and the noun form are seen as synonyms. For example, to act, acting and action are all three names of one kind of activity.

The dictionary contains kinds of activities, irrespective of the question whether they are expected to be executed by people (human activities) or whether they are physical and/or (bio)chemical processes. So the kinds of activities cover the whole life cycle of process plants, from research to demolition, including the activities of the organization and the physical phenomena that take place in the plant during that time.

Kinds of activities do in principle not include the subject of an activity, nor the performer or enabler. For example, 'design', 'maintain' and 'repair' are kinds of activities, but 'design a pump' and 'repair a pump' are can be recorded as subtypes when required. If this would be done in general it would lead to a combinatorial explosion of the number of kinds because of all possible combinations.

A specific activity can be described as follows:

activity-1	is classified as	repair
physical object-2	is classified as	pump
physical object-2	is involved as subject in	in activity-1

10.4.12.2 Kinds of Processes, Process control and Functions

Anything happening over time is an occurrence. Therefore *processes* such as *distillation* or *fabrication*, but also biological processes such as *breathing* and *growing* and business processes, such as *selling* and *accounting* are defined as subtypes of occurrence. Also *process control* processes, such as *control*, *measure* or *actuate* are subtypes of occurrence. These processes are therefore included in the taxonomic dictionary in the same way as kinds of activities.

Examples of physical processes included as kinds of occurrences are: distillation, reaction, catalytic cracking, heat transfer, transport, etc.

Also 'passive' processes and events are included as kinds of occurrence. For example, 'contain', 'happen', 'trip', 'crack', etc.

The term *function* is used in practice in several ways. Sometimes a function is an algorithm or mathematical equation which is a relation between the parameters or inputs and outputs. Examples of such functions are the VDI/VDE 3696 'control functions'. In other cases a function is a role, such as the role (function) of a person. Finally, the term 'function' can also be used to indicate an activity or process (to function). For example a functional decomposition usually means a decomposition of the activities that should be performed. Therefore kinds of functions should be recorded as either subtypes of occurrence, role, or relation (mathematical function).

10.4.13 Kinds of organisms, biological & biochemical objects

Large organisms, such as plants, animals and persons and their components such as hart, leg and kidney, up to cells, proteins, etc. but also bacteria and viruses are included in this domain. The processes in these organisms are covered by the processes and occurrences domain. Their 'material of construction' has a boundary with the substances domain.

10.4.14 Templates and product models (forms and data sheets)

Information is in general expressed as a collection of aspects possessed by objects. Such information is in general required for, or resulting from an activity. It can be recorded in a file or in a document, such as a data sheet, a bill of material, material balance, etc. Those collections of aspects possessed by objects contain a selection of relevant *facts* collected from a specific perspective, called a context. It is always only a sub-set of all the facts that exist about an object. The combination of a complete set of facts about an object is called a *product model*. Such a required or resulting set of facts can also be defined in the form of a *query*.

It is also possible to define a 'fill in the blanks' *template* for the creation of new product models for either an individual thing or a kind of individual thing. A template can be compared with an '*empty form*' with pre-defined pieces of text that define the meaning of the empty fields.

For example, in the terminology of ISO 15926 a centrifugal pump data/requisition sheet defines a set of atomic facts about a pump unit. Each atomic fact consists of a relation between objects, such as pump components,

properties, numeric values, etc., including classification relations with kinds of properties and kinds of pump components, fluids, activities, etc.

On the other hand such a template implicitly defines, so to say, the *query* (e.g. on a database) that selects the data required to order a pump and thus to print such a data sheet.

Application systems need the definition of those templates in order to operate properly.

A Formal English knowledge base may contain the definition of kinds of properties that are common for kinds of physical objects according to those templates. For example, a knowledge base can contains facts such as:

container can have as property a internal volume

This relation is inherited by the subtypes of that physical object, so that a 'pressure vessel' and a 'tank' also can have an internal volume.

The definition of templates can make use of those facts.

An *exchange scenario* defines a set of data that can or should be included in a data exchange, which is part of a standardized kind of activity.

For example, 'order a pump' can be done in a standard way, whereas that standard way can be defined by the definition of a kind of activity. Such an activity requires a specific set of data that is sufficient to specify a required pump. The content of such a specification may be more or less detailed, depending on the company that defines the activity. Therefore, if this is standardized, the standard can and will define only a minimum set. The maximum set includes all characteristics that can be associated with a pump and its components.

10.4.15 Scales and currencies

Scales and units of measure are required to quantify physical properties, coordinates, monetary values and time zones. Therefore this domain dictionary includes:

- Scales, including Units of Measure
- Currencies

10.4.16 Qualitative concepts

This domain dictionary includes:

- Qualitative aspects or values and qualitative ranges for properties and qualities. They are typically used for pick lists and allowed values or domain values.
- Qualitative information
- Standardized types of components

10.4.17 Standard individual things

Apart from kinds of things, there are also '*individual things*'. Those individual things can be used for example as reference objects. They can be distinguished according to their role in *specific objects* and *typical objects*.

Examples of specific objects are countries and organizations.

Examples of typical objects that are included in the dictionary are standard symbols and typical designs, etc., intended for derivation of specific individual objects.

Typical objects usually appear in object libraries.

The dictionary includes a number of well known individual things, such as:

- individual geographic items, such as countries, area's and oceans
- individual organizations.

10.5 Formal dictionary development

The remainder of this chapter describes how to develop and extent a formal dictionary. To some extent this description also covers how to develop knowledge bases and requirements specifications on top of a dictionary by adding expression of knowledge and requirements about the concepts, although the latter is described in following chapters. This chapter also describes how to extent the semantic expression capabilities of a formalized language by defining new kinds of relations and it describes how proposals for extension of a formal language variant can be formulated and submitted.

To include new concepts in a formal dictionary, it is required to first define those concepts according to the rules for a proper definition. Such rules are described in this chapter. They are based among others on ISO 1087 and ISO 16354. A prime rule is that new concepts should be arranged at the proper place in the specialization hierarchy, as a subtype of one or more existing direct supertype concepts. In addition to that, for each new concept it should be defined which aspects and/or components distinguish the defined subtype from the concept that is its direct supertype and from the concepts that are

also a subtype of that supertype (being its neighbors). Those distinguishing aspects and components are called the distinguishers.

It is important to make a clear distinction between relations that are part of a definition model and relations that belong to knowledge models and requirements models, because it may be that people who use the dictionary don't want to be confronted with possible facts that are not by definition the case. Knowledge about concepts belongs to an encyclopedia and not to a dictionary. Requirements about kinds of things belong to standards or are organization context dependent. The creation of knowledge models and requirements models is described in the next chapters.

Relations that determine the definition of a kind of thing are always context-independent and are always valid. Those defining relations are therefore not company specific (except for proprietary inventions) and not dependent on design choices.

On the other hand, facts that express specifications or requirements are by definition only in the context in which they are specified and are thus context sensitive. For example, the fact that a bridge may have pillars does not belong to the definition of a bridge in general, because there are bridges without pillars.

Private extensions to the definition of a formalized language are recommended for proprietary concepts. However, it is recommended to feedback any proposed extensions of concepts that belong to the public domain knowledge.

This implies that company specific concepts can be defined. The definition of such private extensions can also be exchanged in formal language messages and those concepts can subsequently be used in other expressions. However, those private extensions should also be defined according to the rules of the formal languages. Otherwise they will not form a consistent set and do not inherit any information from higher level concepts which would prevent semantic verification.

This extension capability is especially of value for supplier specific catalogues which may contain inventions and proprietary company specific concepts. However, it is recommended to minimize the definition of private concepts in order to minimize the diversity of formal language dialects. If private extensions are nevertheless defined, then they should be made subtypes of existing standard concepts in the formal dictionary.

For example, the concept 'centrifugal pump' is part of the formal dictionary. Then the catalogue item 'Sundyne pump model 311' should be defined as follows:

- Sundyne pump model 311 is a model of Sundyne pump
- Sundyne pump is a specialization of centrifugal pump

So, when a company has exchanged the company specific concept 'Sundyne pump' and its subtyping relation 'Sundyne pump' is a specialization of 'centrifugal pump' together with its defining properties and possibly its options, then the receiving party can classify a specific pump as a 'Sundyne pump' even without referring to a specific model.

10.6 Patterns for definition models

Definitions normally follow a particular pattern.

The simplest form of definition model consists of a single specialization relation between a defined concept and its direct supertype, together with its contextual facts. This is the minimum that is required for the definition of every concept. It specifies a UID, a name, a supertype concept and a textual definition, together with contextual facts such as author, dates and status. The following paragraphs describe more extensive patterns of definition models of various kinds of concepts. This includes also definition models of (standardized) types of physical objects and of manufacturer's models such as catalogue items.

10.6.1 Pattern for the definition of physical objects

The general pattern of relations that form a definition model of a physical object is presented in Figure 96.

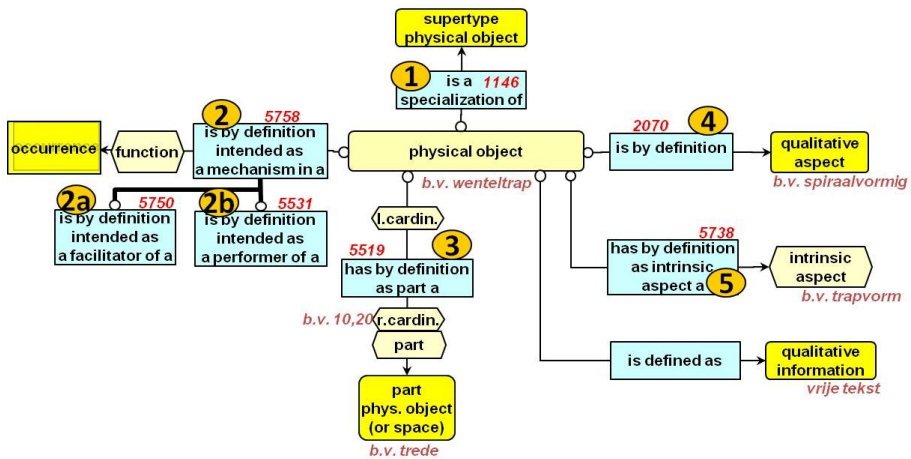


Figure 96, Pattern for definition models of physical objects

The figure shows how a physical object can be defined by various kinds of relations with various other concepts. For example, a concept (a kind of physical object) with UID 123 and name, such as ‘oval room’ can have various defining relations (1, 2, 3, 4 and 5) with other concepts. The possible related concepts are:

1. A kind of physical object that is its supertype
2. A kind of occurrence in which it is intended to be involved
3. A kind of physical object that is by definition a part
4. A qualitative aspect that defines the value of an aspect of the object
5. An intrinsic aspect that may be further defined by a specification of a value or one or more options for aspects of the object
6. A piece of qualitative information (text) that provides a textual definition of the kind of physical object.

This means that for the creation of a definition model for a physical object, the following steps should be executed:

1. Reserve a unique identifier (UID) to represent the kind of physical object.
2. Allocate one or more names, aliases, synonyms, codes and/or abbreviations and symbols to denote the kind of physical object.
3. Define the concept as being a specialization (subtype) of one or more other defined concepts that are its supertype.

4. Specify for which intended function (kind of occurrence) the kind is designed.
5. Specify from which kinds of components the concept is by definition assembled (if applicable).¹⁰
6. Specify which qualitative aspects, such as values for characteristics, properties, qualities or functions, are by definition possessed by the concept.
7. Specify which intrinsic aspect the kind by definition possesses.
8. Provide a textual definition.

10.7 Definition model development procedure

The processes of developing electronic definition models for concepts generally follow a generic procedure. That procedure consists of the following steps:

1. Allocate a unique identifier to the concept.
2. Specify a preferred name of the concept.
3. Specify a direct supertype of the concept.
4. Provide a textual definition of the concept.
5. Specify synonyms, codes, abbreviated terms and translations for the same concept.

If the concept is a subtype of physical object, then the following additional steps can be applicable:

6. Specify a distinguishing aspect of a physical object.
7. Provide an intended function of the concept.
8. Specify defining parts of the concept.
9. Specify pictures and drawings about the concept.

If the concept is a subtype of role, then the following additional steps can be applicable:

10. Specify the kind of thing that is by definition its role player.

¹⁰ Formally speaking: provide a specification of kinds of components when individual things of this kind are by definition composed of individual components of such kinds.

If the concept is an intrinsic aspect, then the following additional steps can be applicable:

- 11. Specify the kind of aspect that is by definition its role player.
- 12. Specify the kind of physical object that is by definition its possessor.

For example, a definition model of a kind of physical object shall include some or all of the expressions in a pattern that is illustrated in Figure 97.

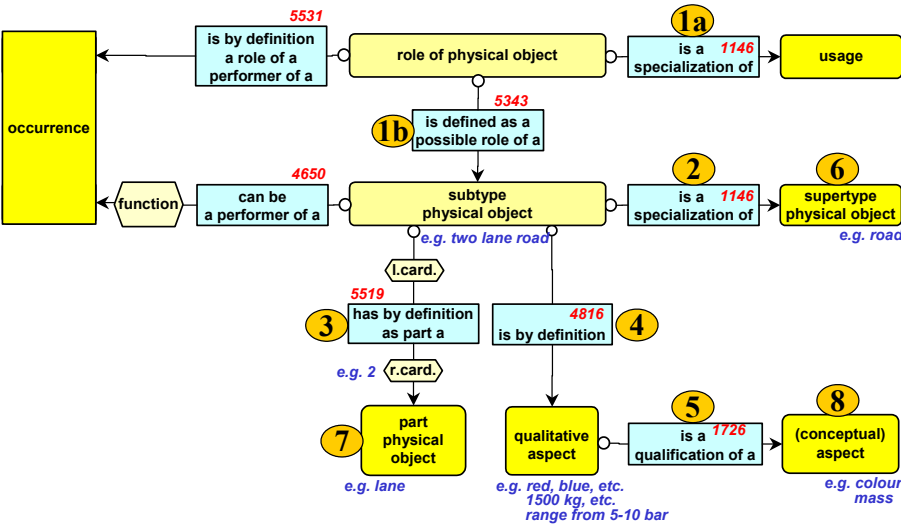


Figure 97, Definition models of a kind of physical object (subtype) and of a kind of role of a physical object

Different patterns are applicable for different kinds of things. The patterns can be translated in templates in user interfaces, so that each pattern can be applied by filling-in a particular template, from which software can (automatically) generate a number of expressions in a formal language.

Table 15 presents an example of a template for a definition model for a kind of physical object.

Definition of a new kind of physical object	
UID of concept:	520.243
Language:	English
Language community:	static engineering
Preferred name:	vessel
Synonym name:	
Abbreviated name:	

Code:	V
...	
Supertype concept UID:	130.108
Supertype concept name:	container
Textual definition:	is a container that is intended for storage of fluid.
Distinguishing function UID:	?
Distinguishing function name:	storage of fluid
Distinguishing aspect UID:	
Distinguishing aspect name:	
Distinguishing aspect value UID:	
Distinguishing aspect value name/number:	
Distinguishing aspect value Unit of Measure:	
...	
Defining part UID:	10161
Defining part name:	inlet / outlet
...	

Table 15, Template for the definition of a new kind of physical object

If a kind is a kind of role, then its definition model shall include that it is a subtype of usage (or role) and that it is defined as a possible role of a kind of physical object. It may also be that is by definition intended to be a performer of a particular kind of process or activity.

For example, the concept ‘water pump’ is a kind of usage of a pump and not a kind of pump. Therefore it shall be defined as follows:

water pump	is a specialization of	usage
water pump	is by definition a role of a	pump
water pump	is by definition a performer of a	pumping of water

10.7.1 Stage 1: Allocate a UID

Each new concept or fact shall be represented in a formal language by a unique identifier (UID). Such a UID shall bear no encoded meaning apart from representing a concept. Therefore is basically an arbitrary code. For processing efficiency a whole number is preferred.

To enable the integration of data from different sources it is essential to ensure that UIDs of concepts and facts are language wide unique. The formal language definition manager therefore provides a service of allocating reserved ranges to organizations, while applying the rules of par. 3.3. This

should guarantee that UUIDs in such ranges will not overlap with other UUIDs in the formal language definition. This enables organizations to allocate UUIDs within their own range to their own concepts and individual things. Organizations may also allocated UUID from the free range (above 10 billion), whereas in that range it is their own responsibility to avoid overlap with other parties. UUIDs in that range may also be used by other parties to represent different things.

Note that translation of existing concept names and expressions of facts do not need new UUIDs as the concepts as well as the facts are language independent and thus can be used for the same concept or fact in various languages.

It is recommended to propose new concepts as extensions of the public domain formal dictionary.

On request, the formal language manager can allocate UUIDs for new concepts and facts within the above-mentioned ranges and thus the concept will become available for general use in the public domain formal dictionary.

In order to avoid overlap between things in a proprietary domain dictionary and the public formal dictionary it is important to ensure that the allocation of UUIDs is a managed process as is described in chapter 16.

10.7.2 Stage 2: Specify names, synonyms and homonyms

The specification of a new concept or kind of thing that is represented in the formal language by a UUID shall have at least one character string that has the role of being a name of the concept. A name consists of an alphanumeric string of characters that expresses how the concept is called in a particular language and language community (such as a discipline or a company or a standard). In other words a name is a *term* or *phrase* that denotes a concept in a particular naming context. Definitions of concepts may partly be provided in graphical form instead of in data and textual form. Stage 13, describes how pictures (symbols) and drawings about concept are related.

Each *individual thing* (represented by a UUID) usually also has a name, although Formal English allows that individual things do not have a name and optionally may be denoted as ‘nameless’, whereas they are denoted via other related objects. Names etc. are related to the named things by a naming relation¹¹.

¹¹ In the Gellish Expression Format the relation type <is called> etc. is implied as a relation between a column with a UUID and an accompanying column with a name.

Concepts as well as individual things may have multiple alias names (synonyms), codes, abbreviations, etc. as well as translations. Note that synonyms shall be really interchangeable and denote the same concept (UID). Therefore, in the expression of such facts the UID of the left hand objects and the UID of the right hand object are the same.

Some examples of subtype relations of the alias relation are illustrated in Table 16.

Language	UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
English	670,171	bike	217	1,981	is a synonym of	670,171	bicycle
Dutch	670,171	fiets	218	4,691	is a translation of	670,171	bicycle
International	430,266	H2O	219	1,983	is a code for	430,266	water
English	430,465	AC	220	1,982	is an abbreviation of	430,465	alternating current

Table 16, Specification of aliases and translations

There are several other relation types available to specify other kinds of strings that are used as names of things, such as <is a ... long name, serial number, title, location code, identifier of>, etc. They all are subtypes of the alias relation type 1980.

For the clarity of visualizations of hierarchies of concepts, it is recommended not to use random synonym names in visualizations of the hierarchy, but to use only ‘preferred terms’ from one or more explicit language communities. (Note that the term ‘preferred term’ means preferred within some language and language community, because preferences are language community context dependent).

It is required to explicitly specify to which language each name belongs. Such a specification of the language adds the term to the vocabulary of that (formalized) language. Thus via the language the whole vocabulary of the formal language can be retrieved. The language also specifies in which language the textual definition is expressed. Furthermore, it shall be specified from which language community (or discipline) within that language the name originates. A language community specifies a local collection of terms and phrases that are the preferred denotations for the concepts that got a name in that community. Within a language community context in a language a name shall be a unique reference to a concept or individual thing. Thus there may be another concept (with a different UID) that is also called by the

same name, but that second name must have its basis in another language community.

For example, a new concept has UID 520243. The concept will be called in English <vessel> and the language community where the name and definition originates is <static equipment technology>, with its own UID 193264 that is selected from the dictionary. Note that this language and language community are sufficient to distinguish this concept from another concept that is also called vessel in English, but in the discipline of transportation and which is a kind of ship.

In Formal English this is expressed as follows:

Language	UID of language community (discipline)	Name of language community (discipline)	UID of thing	Name of kind of relation	Name of thing
English	193,264	static equipment technology	520,243	is called	vessel

Table 17, Allocation of a name to a concept

Note that the concept in the table is called the left hand object, because there will also be a right hand object as will be explained below.

Within the English language the usage of the same term for multiple things are called homonyms. This means that within a language community there are no homonyms, whereas in the formal language as a whole homonyms are allowed, because homonyms are distinguished by different UIDs as well as by different combinations of language community and name. Together, the language and the language community form the naming context that can be used later in case homonyms require to judge which concept (UID) is meant.

The Gellish Expression Format enables that a prime name for a *concept* is allocated together with a definition of that concept by a specialization relation (see next paragraph). The prime name for an *individual thing* can be allocated in the Gellish Expression Format together with a classification relation that defines the individual thing. For example, the name bank is allocated to concept 990152 and the name of the Westminster bank is allocated to its UID as follows in the Gellish Expression Format:

Language	Name of language community	UID of left hand object	Name of left hand object	Name of relation type	UID of right hand object	Name of right hand object
English	business	990152	bank	is a kind of	990008	organization
English	business	1234567	Westminste	is classified as a	990152	bank

			r bank			

Codes, abbreviations, translations, etc. are treated in the same way as names. They also originate from a language and language community context. Exceptions to this rule are international terms such as codes (‘symbols’) for units of measure, numbers, etc. The language context is then called International. The following table presents some examples to allocate names, synonyms, abbreviations, codes and translated names to concepts.

Language	Language community	UID of left hand object	Name of left hand object	Name of relation type	UID of right hand object	Name of right hand objects
English	materials engineering	990152	water	is a kind of	430143	inorganic substance
International	chemical engineering	990152	H2O	is a code for	990152	water
German	materials engineering	990152	Wasser	is a translation of	990152	water
English	electrical engineering	430465	alternating current	is a synonym of	430465	alternating electric current
English	electrical engineering	430465	AC	is an abbreviation of	430465	alternating current

Table 18, Allocation of aliases

10.7.2.1 Denotations by graphical objects

In addition to the allocation of textual names, etc. it is possible to denote concepts by means of graphical objects, such as symbols, drawings or pictograms. Those graphical objects are information objects that are presented as (copies of typical individual) physical objects (“ink on paper”) that may be modeled, or that may be referenced, for example as an electronic file in some native format.

The concept definitions may also be illustrated by examples or typical exemplars. Such examples or typical exemplars are individual objects that can be used as illustrations of instances of the concept or as a template to copy them for the creation of individual exemplars that comply with the definition of the concept. However, objects that differ from those examples may still belong to the kind that is represented by the concept, because the examples and typical objects may not show various alternatives and therefore they are not normative.

This means that a formal dictionary may contain normative drawings and illustrative drawings and pictures. Normative drawings are drawings that specify aspects that are by definition the case for the concept. Illustrative

drawings and pictures present examples or typical examples of instances that nevertheless are related to the concept (kind of thing) for which it is an illustration.

A picture or drawing is an individual physical object. For example, some ink on a piece of paper or bits arranged in a pattern on an electronic or optical device. In a formal dictionary we only deal with electronic pictures and drawings such as files in various formats. For example files in jpg, tiff, pdf, dwg, etc. formats.

Such an individual picture or drawing file can be defined by a classification relation as an electronic data file and is specified to be an element of a directory with as name a URL address. Individual physical drawings (e.g. three drawings: one in pdf format, one in dwg format and one on paper) shall be explicitly distinguished from the (common) qualitative information that is presented on each of those drawings. The files are presentations of the same qualitative information (being the drawing or picture content). Thus, such qualitative information is a qualitative concept that can classify the contents of several individual physical drawings or pictures.

Assume that “picture-1 of a rose” is a representative picture that illustrates what a rose is and assume we have (among others) a rose.jpg file with that picture. Then, altogether the picture is documented in the formal language as a representative picture of the concept ‘rose’ as follows:

UID of left hand object	Name of left hand object	UID of Fact	UID of Relation type	Name of relation type	UID of right hand object	Name of right hand object
106	rose.jpg	208	1,225	is classified as a	490,533	electronic data file
106	rose.jpg	209	1,227	is an element of	107	http://
107	http://	210	1,225	is classified as a	970,274	URL address
108	picture-1 of a rose	211	4,996	is presented on	106	rose.jpg
108	picture-1 of a rose	212	1,726	is defined as a	490,120	picture
108	picture-1 of a rose	213	5,879	is an illustration of a	101	rose

Table 19, Specification of drawings and pictures

Note: Relation type 2072 expresses that the concept ‘rose’ is represented by a graphical element (a line or area or an assembly of lines and/or area’s). It may be that the graphical element is a part of a bigger drawing, whereas the total drawing

is presented on a file. In that case the relation between the picture and the total drawing is as follows:

picture-1 is a part of picture-2

A graphical element is defined as an arbitrary figure and the definition does not constrain it to symbols. Thus graphical representation is the same as visualization.

10.7.2.2 Rules for names of concepts

Names of concepts in a formal dictionary should be consistent to facilitate searching. Therefore they are recommended to satisfy the following rules; although exceptions can always be added as synonyms.

1. *Artificial names* shall be avoided, but natural language and normal engineering names should be used instead.
For example, 'piping_segment_cmpnt' is not used in normal engineering language and thus it should be avoided.
2. *Names shall be in single*, not plural.
For example, use *vessel* and not *vessels* or *fitting* and not *fittings*.
3. *Abbreviations* and *underscores* shall not be used, unless they are explicitly denoted as such by using an alias relation. For example: PC a <is an abbreviation of> personal computer.
4. Names and definitions shall be in *mixed case*, which means generally *lowercase* (in English, nouns starting with an upper case character in German, etc.), with exceptions for generally accepted abbreviations that are in uppercase.
For example: *centrifugal pump* and not *centrif. pump*. Widely used abbreviations and/or uppercase may be included, especially in *synonyms* (see rule 6 below). For example: PC.
Note, this rule holds for English, whereas in other languages different rules may apply. For example, in German all nouns start with an uppercase character.
5. *Multi word names* shall be in the *normal word sequence* for the applied language.
For example, 'centrifugal pump' in is the normal word sequence in English, whereas 'pump centrifugal' may be useful for alphabetic searching, but superfluous in computer supported searching.

6. *Synonyms* are allowed and may be accompanied by a context in which the synonym is valid.

A synonym is an 'alternative' which should be associated with the 'reference name' (via an <is a synonym of> relation. The reference name should be used in all other relations.

For example, *ship* might be seen as a synonym of *sea going vessel*. Synonyms within a language context are e.g. synonyms in French, or German.

7. *Concatenated names* denote kinds that have multiple supertypes. They are valid kinds that shall explicitly be defined as subtypes of each of their supertypes, conform the constitutive components that are mentioned in their name.

For example, the concept *single stage centrifugal compressor* is a correct concept, which should be defined as a subtype of *single stage compressor* and as a subtype of *centrifugal compressor*.

12. *Usage of a dash or a space or no-space?*

For example, should there be a dash ('-') in methylethylketon as follows methyl-ethyl-keton and should we write, slip-on flange, electro-magnetic and soft-start spool?

In the dictionary the 'normal' natural language conventions shall be used. The normal convention for the example is to concatenate the entire word, as in methylethylketon.

In case of doubt it is possible to define a synonym with another spelling for the same concept.

13. *Usage of a dash or a slash?*

For example should a dash or a slash be used in AC-DC or AC/DC?

In a formal language the rule is to minimize abbreviations and codes. The 'normal' natural language conventions in this case is that the dash (-) means 'to' (AC to DC) and the slash (/) means 'or' (AC or DC). Therefore spelling out the preferred term is recommended. So use 'to' and 'or' in full text, but add strings with a dash or a slash as synonyms.

14. *Usage of an s and a z?*

A formal language should follow the rules of natural language. So for UK English this means in most cases an s is used. However, it is valuable to add a synonym with a 'z', because in British English, there are two conventions, one of which uses a z for words like 'organize', 'prioritize' etc. and the other convention uses an s for these same words.

In American (US English) many words take a z, which in British English take an s.

10.7.3 Stage 3: Specify a direct supertype

Each concept (kinds) that is added to a (domain) dictionary shall be defined as a specialization (subtype) of a more general supertype concept that is available (or should also be included) in the dictionary.

This formal definitional relation has as consequence that all kinds inherit the aspects of their parents in their subtype-supertype hierarchy. They inherit definitional aspects as well as knowledge about the aspects.

For example all equipment and equipment components are a *subtype of 'solid item'* and thus they inherit the relations which a solid item has. A solid item is defined as any physical object that consists of solid material and thus has a particular shape. From the properties of physical objects and solid items it can be concluded that all things that are classified by any subtype of solid item have for example the properties *mass, volume, number of items, cost, economic value, marginal value, shape, etc.* If they have such properties by definition, then that does imply that their properties have *some* value, but it does not imply that each product model of an individual exemplar *shall* include an explicitly expressed qualification relation with a qualitative value for those aspects!

So concepts inherit the aspects of all their supertype concepts. Thus the aspects that are inherited needs not be specified for every concept again.

For example, a *bearing* is a solid item and therefore it inherits the fact that it can have a mass, a volume, etc.

All kinds of physical objects also inherit all possible relations that may be made to physical objects as defined in the data model. For example, all kinds can have a name (tag), a description, a material of construction, a shape (geometry), etc.

Subtypes of the concept 'collection' are not required, because all kinds can be used for classifying a collection indirectly by classifying all elements in *collections* by one relation between the collection and the kind. Therefore, no separate kinds are required for collections.

For example: the concept centrifugal pump can be used to classify P-101 or can be used to classify each of the elements in a collection of pumps C-1 in a warehouse as follows.

C-1 is a collection of which each element is classified as a pump

The size of the collection or its 'number of items' is an aspect of the collection that can vary over time.

A specification of a specialization relation means that the defined concept is a further specialization (subtype) of the earlier defined supertype concept. Such

a supertype concept shall either exist already in the dictionary or one of its further defined supertypes shall exist already in the dictionary. Specialization relations require the expression of subtype-supertype relations (1146) that are denoted by the phrase <is a specialization of> or by one of its synonyms <is a kind of> or <is a subtype of>.

Relations of this kind classify relations between a concept that has a role as subtype and another concept that has a role as supertype in the relation. Examples of specialization relations are:

two lane road	is a specialization of	road
centrifugal pump	is a specialization of	pump
pump	is a specialization of	equipment item

Note that this relation can also be interpreted in the inverse direction. Then it defines that the supertype concept is a generalization of the subtype concept. For example: equipment item <has as subtype> pump, or in other words:

road is a generalization of two lane road.

Kinds of roles are also defined by subtype-supertype relations. However, there are many kinds of roles of which holds that roles of such a kind are by definition played by physical objects of a particular kind. For example: fire-fighting pump can be a role of many kinds of pumps. The concept itself is a subtype of usage, which is a subtype of role of physical object. But in addition to a specialization relation with its supertype (usage), it can also be stated as part of its definition model that it is by definition a role of a pump. This is expressed as follows:

fire-fighting pump *is a kind of* usage
fire-fighting pump *is by definition a role of a* pump

The inverse phrase that denotes the latter relation type expresses that by definition a pump (but not necessarily every pump) can play a role as a fire-fighting pump. This is expressed as follows:

pump can by definition possibly play a role as a fire-fighting pump

A specialization relation is a statement that is represented by unique fact id (the UID of the fact). In an Expression Table the specification of the name of a concept (as described in the previous paragraph) can be combined on one line with this specialization relation. Then the expression that a vessel is a kind of container becomes:

Language	Language community (discipline)	UID of left hand object	Name of left hand object	UID of Fact	UID of Relation type	Name of relation type	UID of right hand object	Name of right hand object
English	static equipment technology	520,243	vessel	111	1,446	is a kind of	130,108	container

Table 20, Specification of a subtype-supertype relation

Note 1: In this table, the UID of the right hand object indicates the supertype concept, whereas the name (or one of the names) of that concept is added in the table just for human readability and can be used for consistency verification.

Note 2: The character string (name) that is used in the expression of the fact that specifies the subtype-supertype relation will be the preferred name of the concept in the indicated language community.

The fact that the subtype-supertype relation will be specified for each concept means that all the concepts together will be arranged in one subtype-supertype hierarchy, which is also called a taxonomy. The relation between a defined concept and its supertype concept forms a branch in this hierarchy of concepts.

Thus each concept in a domain taxonomic dictionary should be a direct or indirect subtype of other concepts in the total hierarchy. When a domain taxonomic dictionary is developed it is usually convenient to begin with a pre-selection of a number of concepts from the higher level basic dictionary that together will form the top-concepts of the domain taxonomic dictionary. This will simplify the search for proper supertype concepts, because the pre-selection will reduce the search to a limited number of concepts that are already in the domain of the domain taxonomic dictionary. This work method requires that during the preparation for the development of the domain taxonomic dictionary those selected domain top-concepts shall be defined as subtypes of higher level concepts in the dictionary. It implies that the domain taxonomic dictionary becomes a truncated pyramid under the basic dictionary concepts.

Note that a defined concept can have a supertype concept that is either a concept from the list of domain top-concepts or it is a not yet defined supertype concept. Such a not yet defined supertype concept is itself a new concept that shall be defined in the same way as the defined concept.

During the development of a domain taxonomic dictionary the hierarchy will be subject to changes that may influence other parts of the hierarchy. A regular check on the consistency of the whole hierarchy is therefore

important to achieve a high quality taxonomy. The use of tools to regenerate the hierarchy is therefore recommended.

The following rule may be helpful to judge whether a concept is a subtype of another concept:

- If the potential subtype concept has the same function and is defined by the same aspects as the potential supertype concept, then it might well be a true subtype concept.

A subtype concept inherits all aspects that the supertype concept has, so that the definition of the supertype concept does not need to be repeated for its subtype concepts. The only things that need to be defined for the subtype concept is its differences from the supertype concept and the differences from its neighboring concepts that are also subtype of the same supertype concept.

The hierarchy in Figure 5 illustrates a number of specialization relations (subtype-supertype relations) that form a taxonomy of defined objects, such as motorway, road, civil item, physical object, etc.

Because all concepts are subtypes of more generic concepts, they are all arranged in a hierarchical taxonomic structure that gives insight in how they all relate to each other. Each more detailed branch in the hierarchical structure includes more specialized concepts. This is illustrated on the vertical axis of Figure 98: the concepts that are located lower on the vertical axis are more specialized.

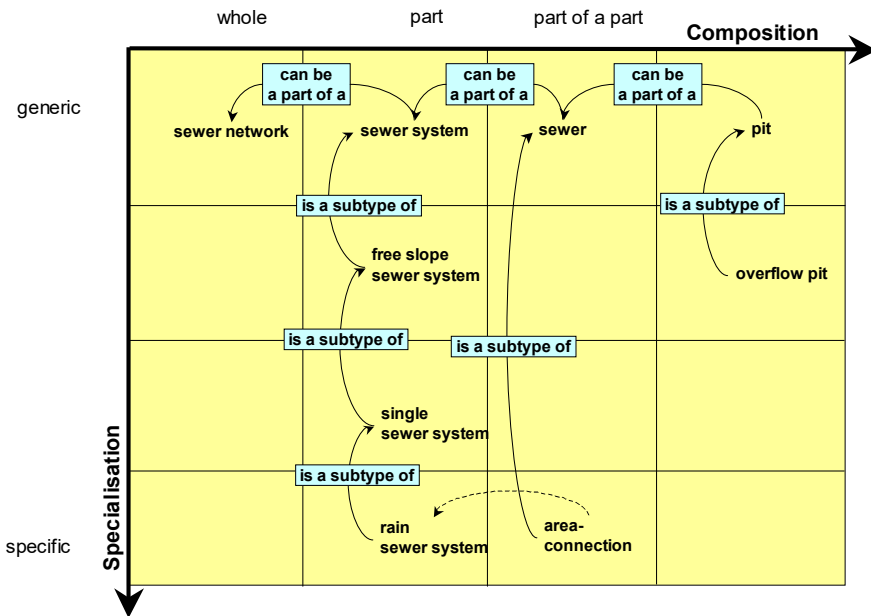


Figure 98, Basic structure for object definitions: specialization and composition

As is mentioned before, the hierarchical structure of subtypes of concepts in a dictionary is called a specialization hierarchy or taxonomy. One concept may have more than one supertype concept and a supertype usually will have more than one subtype concept. This means that the total hierarchy will not have a true tree-structure, but it will be a hierarchical network structure. The concept at the absolute top of the hierarchy is called 'anything'.

To facilitate that all the objects that are used in projects can be classified by the concepts in a formal dictionary it is required that sufficient concepts in various application domains are added to the domain dictionaries. This means that projects use a dictionary with unambiguous definitions that is common for all parties that participate in the project. Furthermore those concepts can be used to find all objects of the same kind and they can be used to attach and find knowledge about those concepts.

If a subtype is a qualitative (or quantitative) aspect or a (standardized) type, then the relation type that shall be used to qualify its relation with its supertype concept is a qualitative subtype relation, denoted by the phrase <is a qualitative subtype of>. This relation type is a subtype of the specialization relation that is denoted by the phrase <is a specialization of> or <is a kind of>.

If the subtype is a further detailed subtype that is a standard manufacturer's model, typically a model that includes a defined size and possibly other qualitative specifications, then the relation type to be used is <is a model of>.

Multiple subtypes of the same supertype should be mutually exclusive when they are based on the same criterion for subtyping. This will ensure that knowledge or requirements that are expressed for things of a kind are indeed applicable to all members of that kind. Strictly speaking they are not necessarily mutually exclusive, but overlap in definitions may cause confusion about the applicability of rules and constraints. Furthermore, collections of subtypes may or may not be complete, which means that there may be individual things that can be classified by a concept, whereas it may be impossible to classify them by one of its subtypes. For example, assume we have a collection of individual pumps. If the subtypes of 'pump' form a complete collection, then all pumps can be classified by at least one of the elements of the collection of subtype concepts. However, it does not need to be a complete collection and even if it is a complete collection, then users are still free to classify an individual thing only by the supertype. It should be avoided to create concepts such as 'other pump' as a way out to make incomplete collections of subtypes complete. As the definition of kinds such as 'other pump' are dependent on the availability of other subtypes, their definition is ambiguous. Therefore, in case of incomplete collections of subtypes, things that cannot be classified by a subtype should be classified by their supertype.

The inheritance principle

A specialization relation between concepts implies that everything that is the case for the supertype concept is also the case for the subtype concept. For example, the fact that a car is a kind of a vehicle means that everything that is the case for the concept vehicle is also the case for the concept car. This implies that a more specialized (subtype) concept "inherits" all knowledge about its supertype concept that is located at the higher level in the specialization hierarchy. Because that is the case for each concept in the hierarchy, this means that a concept inherits all aspects of all concepts that are arranged higher in the specialization hierarchy. In other words, it inherits everything from all its supertypes (all its generalizations).

Note that this inheritance mechanism only operates via the specialization relations. Knowledge is not inherited via composition relations. So, if a component is a part of an assembly, then what is true for an assembly is not inherited to the component.

For example, a path is a subtype of road, therefore a path inherits knowledge that is available about the concept road. Further specializations of path are foot path and bike path. This means that both inherit knowledge from path as well as from road.

So, a concept inherits always via the specialization relations. In other words, it inherits via the <is a kind of> relations.

Rule:

- Multiple specializations and multiple generalizations are allowed for any concept. This means that a concept may have several subtypes and also several supertypes.

Specialization of aspects by qualitative aspects

An aspect is phenomenon that describes something about a physical object or activity but is not the object itself. For example, a length, pressure, color, material of construction, duration, etc. Aspects are defined largely in the same way as physical objects. They also have one or more names and their kinds are defined as subtypes of higher level more generic kinds of aspects. However they don't have parts and don't possess other aspects, although aspects may be correlated.

Kinds of aspects can be distinguished in conceptual aspects and qualitative aspects. A qualitative aspect is a numeric or textually defined value of an aspect. The values: 3, '3 mm', red, high, stainless steel, 5 seconds, are examples of qualitative aspects that are qualitative subtypes of the conceptual aspects, length, color, etc.

Qualitative aspects are defined by a special kind of specialization relation, called a qualitative subtype relation. So, instead of an <is a kind of> relation a qualitative aspect is defined for example as follows:

red is a qualitative subtype of color

10.7.4 Stage 4: Add a textual description

Each concept shall be related to a textual description (or definition), although theoretically such a definition can be composed from a definition model. A textual description shall consist of a character string (text, consisting of one or more sentences) in the natural language that is indicated as the context for the preferred name of the concept.

Definitions of kinds shall build on the definition of their direct supertype. Therefore, textual description shall begin with a reference to its direct

supertype concept(s) and shall specify in which respect the defined subtype concept is distinguished from its direct supertype(s) and in which respect (aspect value(s)) it is distinguished from its neighboring concepts that are also subtype of the same supertype.

For example:

ball bearing is a specialization of *bearing which contains balls*.

As a result a definition may sometimes seem to be simplistic. However that is not the case. You should be aware that the name of the direct supertype in fact stands for the definition of that direct supertype which has a direct supertype again with a definition, etc. etc. until the top of the classification hierarchy is reached. This chain delivers a very rich total definition.

The distinguisher that defines the kind of physical object can be one of three possibilities:

- a. A distinguishing aspect value (a qualitative aspect that may be quantitative)
- b. An essential part (a component that itself is also is a physical object)
- c. An occurrence (an activity, a process or an event) that specifies the function for which the kind of physical object is designed as performer.

Therefore, after mentioning a supertype, a textual definition shall continue with a description of the aspect that is used for determining the classification of such things and its value, or of the kind of component that shall by definition a part, or of the function (the activity or process) for which such things are intended to be performers or enablers.

For example, an additional textual definition might start with 'that is intended for', when a concept is used for classification according to an intended activity; or it might start with 'which consists of', when the concept is used for classification according to composition, etc.

The concept UID shall be related to the description by a description relation.

A textual definition builds on the definitions of its direct and indirect supertypes, and after mentioning its supertype concept it preferably starts with a phrase such as 'that consists of' or with 'that is intended to' (which means: 'that is intended to be suitable to') or 'of which its' (followed by the distinguisher).

This means that the textual definition should preferably be read as follows:

... is a kind of ... that has a *[kind of aspect]* that is *[value]*, *[etc.]* or
... is a kind of ... that consists of a *[kind of physical object]* or
... is a kind of ... that is intended for *[kind of activity or process]*.

When the definition is given this way the ‘full sentence definition’ can be generated automatically by the concatenation of the relation with the direct supertype and an additional (partial) textual definition.

A textual definition may be extended with a description of what is typically but not necessarily the case. Such a description shall be expressed in a separate sentence and shall begin with the word ‘Typically’. It may also be extended with examples.

For example:

vessel is a container that is intended for storage and/or processing of fluids. Typically it is closed. For example, a steam drum.

The Gellish Expression Format enables to add a textual definition on the same line as the specification of its name and the specification of its supertype concept. Thus the naming relation and the description relation remain implicit in that format.

10.7.5 Stage 5: Specify defining components

Often a concept, such as a kind of physical object or activity is defined by the fact that all of its well-formed members have by definition one or more particular kinds of other objects as their component(s). Typically such components enable the capabilities of the assembly to perform its intended function. Such components are distinguishers that define the whole assembly. Therefore we call such kinds of components, *defining components*. In such cases the definition model of the defined concept should include composition relations with its kinds of components as far as they are by definition components of the defined kind of whole.

Many kinds of things *can* have or *shall* have a particular variety of components. This can be modeled by possible and required composition relations, which express how things of such kinds can be or shall be composed of things of other kinds. whereas those possible and required compositions describe options and requirements for options, whereas those options do not define the concept. This means that such composition relations do not belong to definition models, but should be included in knowledge models and requirements models respectively. Various knowledge bases contain large numbers of such possible assemblies.

For example, a car is defined among others by being driven by an engine and may have a turbo. Thus a well-formed car is an assembly with an engine, but a car without an engine is still a car, although not well formed, because it may be in repair. Thus, an engine is a defining part for a (well-formed) car. A turbo is an option for a car, thus the possible composition relation that expresses that option is not part of a definition model of the concept car, but could be included in a knowledge model.

Thus, for composition relations we distinguish between kinds of relations for composites that are *by definition* composed of components of particular kinds and kinds of relations for components that are *optional* or *required*. Optional composition relations are expressed in the form of (conceptual) composition relations between concepts (denoted by the phrase <can have as component a>) or as (conceptual) required composition relations that express requirements of particular organizations (denoted by the phrase <shall have as component a>). If a concept has by definition (always) a particular kind of object as part, then that is a defining fact. Such a fact is expressed by a defining composition relation (denoted by the phrase <has by definition as component a> and that fact then belongs to the definition model of that concept. Thus a definition model should only include defining relations.

Because of the difference between assemblies and collections we also make a distinction between *assembly relations* between parts and whole assemblies and *collection relations* between elements and whole collections.

Assembly relations

Defining assembly relations can expressed by the phrase <has by definition as part a>.

A defining assembly relation describes a 'part-whole' relation, defining that members of one concept (being the composite or assembly) by definition has as part one or more members of another concept, being the part(s). For example, a 'shell and tube heat exchanger' has by definition each of the following concepts as part: a shell, a tube, a nozzle and a head. (for a specification of the number of parts, see par. 14.32 'cardinality constraints'. Defining compositions can be expressed as in the following example:

shell and tube heat exchanger	has by definition as part a	shell
shell and tube heat exchanger	has by definition as part a	tube
shell and tube heat exchanger	has by definition as part a	nozzle
shell and tube heat exchanger	has by definition as part a	head

Notes:

The fact that there are minima and maxima on the number of parts of the same kind can be defined explicitly by specifying cardinality constraints (see par. 14.32, cardinality constraints)

This relation type can be denoted by the above phrase, but also by an inverse phrase, expressing the fact in an inverse order as the fact that a component is by definition a possible component of a whole assembly of this kind. For example:

shell *is by definition a possible part of a* shell and tube heat
exchanger

The inverse relations specifies a *possible* part, because not every shell has to be a part of a shell and tube heat exchanger. This difference between the ‘cardinalities’ of the related things can also be expressed explicitly by cardinality constraints.

When there are subtypes defined for a kind of part, then the various parts of different members of the whole kind can be classified by different subtypes. Those subtypes of the parts can be the cause that also subtypes of the kind of assembly are defined. For example, there are different types of heads for heat exchangers: fixed heads and floating heads. So a heat exchanger has options for the kinds of heads, but if we define that a fixed head exchanger has by definition as part a fixed head, then for fixed head exchanger there is no option for its head type.

Rule:

- an item may be part of more than one assembly at the same time.

Collection relations

Collection relations can be expressed by the phrase ‘*has by definition as element a*’.

Apart from single things, we also recognize *collections of things*, which may be collections of things of the same kind or type, such as manufacturer’s models or collections of things of different kinds with a common supertype concept. For example, a collection of pumps (of various types) in a warehouse.

Collections are plural things. A collection is something in their own right (just as an assembly). This means that catalogues, product libraries, bills of material, etc. are objects themselves, that can have collection relations with their constitutive components.

A special kind of relation enables to express that each element in a collection

is classified by a particular concept, whereas the whole collection is recommended to be classified only on a high abstraction level. For example, 'my stamp collection' is classified as a collection AND for my stamp collection holds the relation that <each of which elements is classified as a> stamp. This has as advantage that the language and dictionary does not need to contain separate concepts and names for all kinds of collections, but that a specific collection is classified as collection AND that each element is classified. To support this the dictionary contains generic kinds, such as 'collection', 'catalogue', 'bill of material', etc.

For example, there is no need for a concept called 'pumps' next to pump, because a particular individual pump collection can be classified as 'collection' (or one of its subtypes) and can have a separate defining relation that expresses that 'each of which elements is classified as a' pump.

A conceptual collection relation expresses a relation between an element of a concept and a collection of a concept. This means that members of one concept '*can be an element of*' members of another concept, being a collection.

For example:

- a schematic symbol 'can be an element of a' symbol library
- a piping component 'can be an element of a' piping dictionary
- a spare part 'can be an element of a' stock in a warehouse
- a material 'can be an element of a' catalogue.

The inverse phrase expresses the possible decomposition of a collection of such a concept into individual elements of the indicated kind.

For example: a piping class '*can include a*' valve and a fitting.

Cardinalities

The example above that specifies that "a car has by definition as part a wheel" illustrates that it is required to specify also the minimum and maximum number of parts of the same kind that are by definition (at the same time) part of the assembly. These numbers are called the minimum and maximum (simultaneous) cardinalities. If the cardinalities are not specified then the allowed number of parts is greater than zero, but further undefined. In this example, a car shall have at least three wheels, but the maximum number is unspecified. This is specified as a sequence of two numbers, such as (3, n).

Specialization relations versus composition relations

When modeling knowledge it is not as easy as it may seem at first sight to make the proper distinction between specialization relations and composition relations (stage 2 and stage 3).

It is important to understand the procedure for modeling knowledge in a logical and systematic way. Starting from the definition of an assembly concept at a generic level, specifying its subtypes as well as its kinds of parts and the kinds of parts of its subtypes, up to the specification of the parts of its parts and the subtypes of its parts. In other words, the route from generic to specific and from assemblies to parts.

Examples of the composition relations are expressed as follows:

sewer network	has by definition as part a	(1,n) sewer system
sewer system	can have as part a	(1,n) sewer subsystem
sewer subsystem	can have as part a	(1,n) sewer subsystem component

On each level in the composition hierarchy, such as sewer network, sewer system, sewer subsystem and sewer subsystem component, it is possible to define specializations (subtypes) of the type of systems or components. The framework for knowledge is therefore created through the alternating specification of specialization relations and composition relations. This should be done as follows:

1. Start with a *kind of thing* at the highest assembly (compositions) level and define which subtype concepts exist. (In the example, we don't have subtypes of sewer network).
2. Specify from which kinds of *components* such a kind of assembly can be composed.
3. Define for each of those kinds of components the kinds of *subtypes* that exist.
4. Specify for each kind of subtype of which kind of *components* it can be composed.
5. Etcetera.

10.7.6 Stage 6: Specify defining intrinsic aspects

The fact that a concept is a subtype of its supertype concept implies that there is at least one difference between the definition of the subtype concept and the definition of its direct supertype concept(s). For a concept that is not a kind of aspect, that difference can be a defined composition, but usually the

subtype has by definition a particular aspect value as distinguishing factor. Thus, the supertype concept has freedom (or more freedom) for the values of the aspect than the subtype, whereas the subtype concept is defined by a particular value or a narrower value range for that aspect. This stage therefore describes the specification of the defining aspects, by which concepts, especially kinds of physical objects, are further defined.

In a similar way as with the components we make a distinction between defining aspects and optional aspects. Defining aspects of a kind of thing are qualitative aspects that define a kind of thing by always being (intrinsically) possessed by any thing of that kind. In other words, things of that kind always have a value that is qualified (or quantified) by the defining qualitative value. Thus such aspects are intrinsic for objects of such kinds.

For example, a 'red rose' <is by definition> 'red'. However, 'red' is not by definition only an aspect of red roses, but also of other red things. Nevertheless, we can express the inverse relation as 'red' <is a defining qualitative aspect of a> 'red rose'.

A definition model of a concept includes relations with defining aspects only and not with optional aspects. Those defining aspects are also called distinguishing aspects or distinguishers, because they are the aspects that help to distinguish between the kinds of things that they define. The definition model of a concept can therefore be extended with expressions of the possession of such defining aspects.

This is illustrated with relation 4 in Figure 96.

Actually, a defining aspect for a kind of thing is always a qualitative aspect (also called an 'aspect value'¹²). For example, each pump has an operating principle, but the concept pump is not defined by a particular operating principle. Nevertheless, it could be specified that the operating principle of a pump can be one of a collection of allowed values. However, subtypes of the concept pump, such as 'centrifugal pump', are *defined* by the fact that their operating principle is by definition a particular value, such as 'centrifugal'.

This means that for the supertype concept pump it can be specified that it has subtypes that have as distinguishing aspect their 'operating principle'. This illustrated in Figure 99.

¹² The term aspect value can be confusing, because it may refer either to an individual aspect value, or to a generic qualitative aspect value (a quality) or a quantitative aspect value (a number) that qualifies that individual aspect. Therefore, we use the term 'qualitative aspect' for a generic qualitative or quantitative concept that is 'universal', independent of any individual situation.

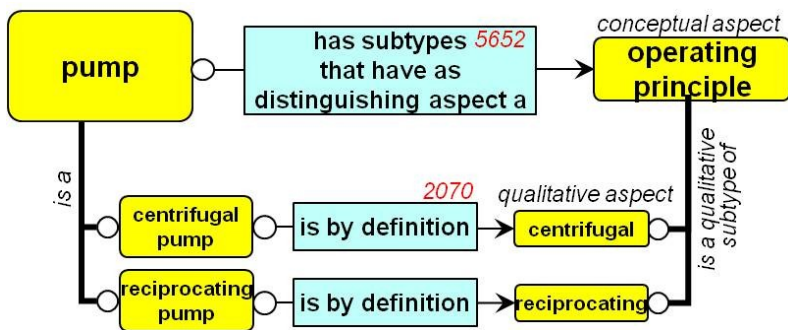


Figure 99, Defining qualitative aspects for subtypes of pump

The facts from Figure 99 are expressed in Table 21.

Name of left hand object	UID of relation type	Name of relation type	Name of right hand object
pump	5652	has subtypes that have as distinguishing aspect a	operating principle
centrifugal pump	1146	is a specialization of	pump
centrifugal	1726	is a qualitative subtype of	operating principle
centrifugal pump	5283	is by definition	centrifugal
pump	5652	has subtypes that have as distinguishing aspect a	orientation
horizontal pump	1146	is a specialization of	pump
horizontal	1726	is a qualitative subtype of	orientation
horizontal pump	5283	is by definition	horizontal

Table 21, Definition of subtypes by distinguishing aspects

Note: The value of the distinguishing aspect is the real distinguisher that defines the subtype concept.

Thus the concept ‘operating principle’ is qualitatively specialized as ‘centrifugal’, ‘reciprocating’, etc. This implies the inverse relations that express for example that the qualitative aspect (a qualitative concept) ‘centrifugal’ is defined as a qualitative subtype of ‘operating principle’, as is expressed on the third line in Table 21.

Different qualitative subtype aspects will usually be mutually exclusive. However, they do not necessarily exclude each other.

A collection of all defined subtypes of a concept which specializations are based on qualitative values of the same supertype kind of aspect may be a complete collection, but not necessarily it is a complete collection. In other words there may be other subtype concepts, because there may be other

currently undefined subtype qualitative aspects. Thus often the option should be open to add new subtypes.

A concept can have multiple collections of subtypes that are defined on the basis of different distinguishing aspect. For example, the concept pump it typically specialized on the basis of operating principle, but also on the basis of the orientation of its shaft. That definition and the definition of for example its subtype ‘horizontal pump’ is defined in the second half of Table 21.

Subtypes that are based on different conceptual aspects, such as operating principle and orientation will usually overlap and thus are usually not mutually exclusive.

A concept that has by definition multiple qualitative aspects (values) from different distinguishing aspects will be a further sub-subtype. For example, the concept called ‘horizontal centrifugal pump’, which will be a subtype of two supertype concepts. Thus that concept is an example of a concept that has two parent concepts. This is expressed as follows:

Name of left hand object	UID of relation type	Name of relation type	Name of right hand object
horizontal centrifugal pump	1146	is a specialization of	horizontal pump
horizontal centrifugal pump	1146	is a specialization of	centrifugal pump

Such multi-parent concepts inherits multiple distinguishing aspects from their parents, whereas they do not necessarily have an additional distinguishing aspect, because they are sufficiently distinct by inheriting multiple qualitative aspects.

Multi parent concept may have textual and modeled definitions that just consist of the specialization relations with their supertype concepts.

Generation of textual definitions

The above relations specify pieces of a definition models. Such definition models can be used by software for automatically generating textual definitions.

For the example of a centrifugal pump the following textual definition can be generated automatically from the above expressions:

centrifugal pump is
a kind of pump of which its operating principle is centrifugal

and an automatically generated textual definition for the concept ‘horizontal pump’ would be:

horizontal pump is
a kind of pump of which its orientation is horizontal.

In case of a multi-parent concept it is possible to derive a part of a textual definition from each of the defining expressions of relations. For example:

- a horizontal centrifugal pump is defined as:
a horizontal pump of which its operating principle is centrifugal
- a horizontal centrifugal pump is defined as:
a centrifugal pump of which its orientation is horizontal

Such multiple definitions can be combined into one definition that refers to the further supertype (grandparent) concept as follows:

- a horizontal centrifugal pump is defined as:
a *pump* of which its operating principle is centrifugal AND of which its orientation is horizontal.

10.7.7 Stage 7: Define the nature of intrinsic aspects

The concept ‘centrifugal’ plays an ‘intrinsic aspect’ role as ‘operating principle of a centrifugal pump’ on the fourth line of Table 21. The relation between the possessor and that intrinsic aspect role can also be specified explicitly as follows:

centrifugal pump	5848	has by definition as intrinsic aspect a	operating principle of a centrifugal pump

Such an intrinsic aspect is defined by three relations. A relation with its supertype ‘intrinsic aspect’, a relation with its possessor (the inverse of the above relation), a relation with the aspect of which it is a role.

operating principle of a centrifugal pump	1146	is a kind of	intrinsic aspect
operating principle of a centrifugal pump	5848	is by definition an intrinsic aspect of a	centrifugal pump
operating principle of a centrifugal pump	5817	is by definition an intrinsic	operating principle

Finally the definitions can be made complete by relating the role to that qualitative aspect as follows:

operating principle of a centrifugal pump	5283	is by definition qualified as	centrifugal

A distinguishing (defining) kind of intrinsic aspect of a concept shall be defined by specifying what its nature is. Each distinguishing kind of intrinsic aspect such as a ‘pipe diameter’ is a subtype of intrinsic aspect (which is a role of an aspect) and shall therefore be defined as follows:

pipe diameter

is (by definition) a subtype of

intrinsic aspect

pipe diameter

is by definition an intrinsic

diameter

pipe diameter

is by definition an intrinsic aspect of a

pipe

10.7.8 Stage 8: Define values of intrinsic aspects

The *qualitative aspects*, such as centrifugal and horizontal in the above example, are concepts that need to be defined and included in the formal dictionary so that they can be selected from the dictionary. A definition of such a concept value is done in a similar way as the definition of other concepts, except that the a concept value is defined by a qualitative subtyping relation (<is a qualitative subtype of>), whereas qualitative aspects have no distinguishing aspects. The qualitative subtype relation is a subtype of a specialization relation, that specifies that the subtype is qualitative.

Sometimes the distinguishing aspect is quantified by a number on a scale, indicated by a unit of measure. The relation type that indicates the quantification on a scale can also indicate whether the value is equal, greater than or less than the number on the scale. For example:

UID of left hand object	Name of left hand object	UID of Fact	UID of Relation type	Name of relation type	UID of right hand object	Name of right hand object	UID of UoM	UoM
104	M6 bolt	205	5,317	has by definition as possible intrinsic aspect a	105	diameter of a shaft of an M6 bolt		
105	diameter of a shaft of an M6 bolt	206	5,737	is by definition quantified on scale as equal to	2,000,000,006	6	570,423	mm

Table 22, Specification of a distinguishing quantitative aspect (value)

The other relation types are:

- 6049 is by definition quantified on scale as greater than
- 5978 is by definition quantified on scale as greater than or equal to
- 6052 is by definition quantified on scale as less than
- 5979 is by definition quantified on scale as less than or equal to

Typical kinds of aspects (subtypes of aspect) that need to be considered to determine distinguishing aspects are:

- Aspect
 - technology (operating principle)
 - shape
 - size
 - topology (structure)
 - material of construction

A distinguisher is not necessarily an aspect, it can also be one (or more) of the following kinds of discriminators:

- Function (role in an activity or process)
 - application (purpose)
- Composition
 - obligatory parts
 - configuration (arrangement)
- Organization (owner, custodian, etc.)

The specification of such other distinguishers is discussed in the next paragraphs.

The value of a distinguishing aspect shall also be defined. This is done by specifying that the intrinsic aspect is by definition qualified by a qualitative aspect.

A qualitative aspects is an aspect that is qualified by a value. For example '3 meter' is a qualitative aspect with a value 3 on a meter scale. Such a qualitative aspect is a qualitative subtype of the general aspect length. Length is therefore also called an example of a conceptual aspect.

Another example is the color red. Red is a qualitative aspect of the conceptual aspect color. So qualitative aspects may be quantifications on a scale or may be qualified by textual 'values'. A number of qualitative aspects that are qualifications of the same conceptual aspect are often collected in

collections of allowed values, such as in pick lists in user interfaces of software systems.

The formal dictionary already contains a large number of qualitative aspects, but nevertheless it will be frequently required to define additional qualitative aspects.

So, the definition of a qualitative aspect implies that the aspect value shall be defined as a qualitative subtype of a more general aspect. Examples of such definitions are:

reciprocating	is a qualitative subtype of	operation principle
3 m	is a qualitative subtype of	distance

Example of a definition of the concept ‘two lane road’

Figure 100 is an illustration of an example of a definition model for the concept ‘two lane road’ with as UID 123.

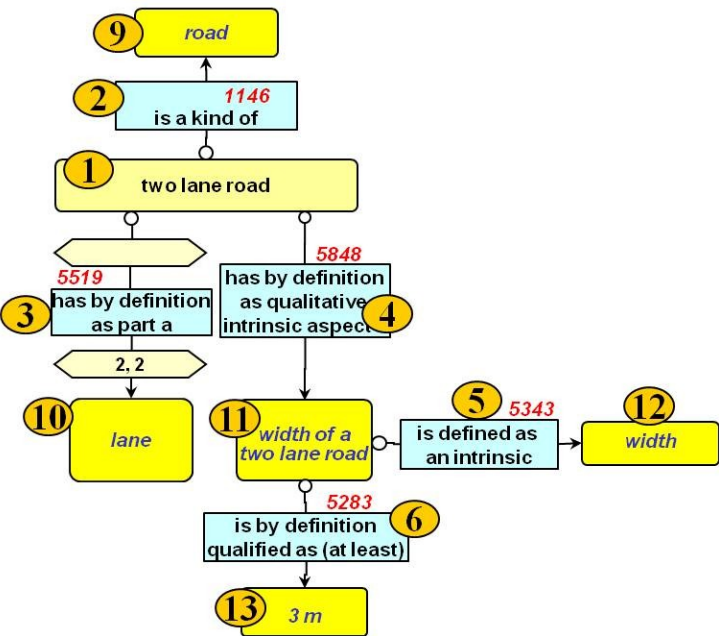


Figure 100, Definition model of the concept two lane road

The concept two lane road in Figure 100 is defined trough the following steps:

1. the concept is given a name: two lane road (fact 1)

2. it is defined that the concept is a kind of the concept road (fact 2)
3. it is defined that this kind of road has by definition two lanes (fact 3)
4. it is defined that this kind of road has by definition a qualitative intrinsic aspect called ‘width of a two lane road’ (fact 4)
5. for that intrinsic aspect it is defined that its nature is that it is by definition an intrinsic width (fact 5)
6. for that intrinsic aspect it is defined that it is by definition qualified as at least 3 m (fact 6)

Note that it is assumed that the related concepts already exist in the formal dictionary. If that is not the case, then they shall be defined by a similar procedure.

The recording of the above facts in a Formal English expression table is the way in which the concept two lane road is included in the domain dictionary.

10.7.9 Stage 9, Specify functions and define occurrences

A concept can be defined by a distinguisher that is its *intended* function. Such a function is a role for which objects of such a defined kind are designed. It means that a well-formed object of such a kind is intended to be suitable to perform or enable an occurrence, such as an *activity or process*, to take place. The kind of object therefore will have intrinsic aspects that enable to perform such a function, although such aspects might not be specified explicitly. It does not imply that an object of such a kind actually performs such a function, but it is intended to have the capability to do so.

Apart from the function for which such a kind of object is made suitable, it is possible that objects of such a kind may also have other functions, including intended as well as actual functions.

A relation between a kind of physical object and its intended function can be specified by relations of the kind <has by definition as intended function>. For example, compressors are made for an intended process called ‘compression’. That can be expressed as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
130,069	compressor	207	5,536	has by definition as intended function	191,936	compression

Note that the right hand object is an occurrence (activity, process or event), which is the occurrence that is intended to be performed by the fulfiller of the function (the player of the role).

A function that is to be fulfilled is an occurrence that is wanted to take place¹³. Such an occurrence is called a function, because the occurrence shall be performed or enabled by some performer or enabler although we may not yet know or specify the performer or enabler. As a consequence, defining a kind of function is the same as defining a kind of occurrence (activity, process or event) that is meant by the function.

An occurrence, being an activity, a process or event, is an interaction between a number of interacting objects that take place over time. Events are occurrences of short duration and processes are typically occurrences of longer duration. When only the interaction between a performer and a subject is modeled, then the interaction can be modeled as a binary relation.

An occurrence shall be defined as being a subtype of its direct supertype occurrence.

Typical knowledge about kinds of occurrences are facts that specify the kinds of things that are able to perform those occurrences or that can be subject in or that can be enablers of those occurrences. The definition of a kind of occurrence and the knowledge about it shall be modeled by specifying facts such as those that are described below and that are indicated in Figure 101 (note that non relevant names of roles are not shown in the figure).

¹³ A formal definition of a function is: a function is a role of an occurrence of being intended to be performed. In other words, if an occurrence is wanted to be performed by a physical object, then that occurrence has a role as 'function' in a relation with its (potential) performer.

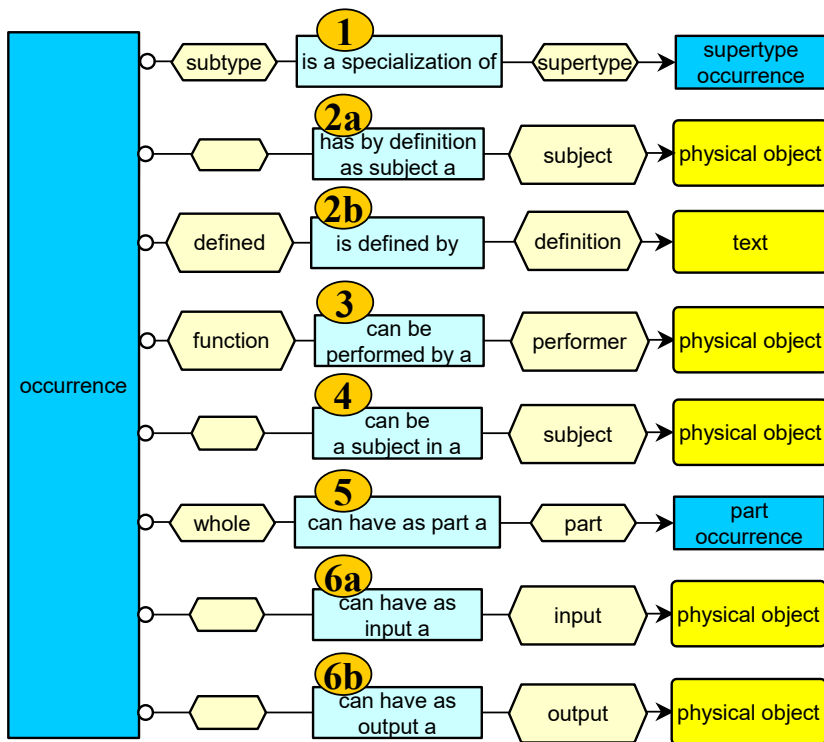


Figure 101, Definition of a function and specification of knowledge about it

Step 1: Define the direct supertype(s)

Define the type of occurrence by specifying it as a subtype of a supertype occurrence. This is normally done by using a specialization relation ('is a specialization of'). However, functions are often qualified by the material that is subject in the process. Therefore, in those cases a function is defined as a qualification of a more generic occurrence.

For example as follows:

transport of waste water	is a qualitative subtype of	transport
pumping of waste water	is a qualitative subtype of	pumping

Step 2: Add a textual definition

Add a textual definition of the occurrence or define the distinguisher that distinguishes the subtype from its neighbor subtypes in an explicit way. The distinguisher that defines the type of occurrence can be a distinguishing aspect (e.g. a frequency), an essential part occurrence or an obligatory subject or performer of the type of occurrence.

For example as follows:

pumping of waste water	has by definition as subject a	waste water stream
transport of waste water	has by definition as subject a	waste water stream

Note, each kind of thing that is introduced, either as left hand object or as right hand object, needs to be defined in the same way as the occurrence is defined. So, the example above requires that waste water stream is defined by a specialization relation and a textual definition or a distinguisher.

10.7.10 Stage 10: Geometric representation of objects

We can distinguish between a kind of 'graphical element' and a concept that it can represent. For example, a valve symbol can represent a valve.

Furthermore, we can distinguish between symbols which are assemblies of lines, circles, etc. and the geometric aspect of symbols (the shape of the symbol and the shape of its parts).

In the formal dictionary there are therefore kinds of symbols (and possibly typical individual symbols) and relations between the kinds of symbols and the concepts that they can represent.

The shape of kinds of symbols in a symbol library, is often company specific, although various standards institutes have published catalogues with 'standard symbols'.

Such catalogue items with the geometry (shape) of symbols can be expressed in Formal English, but the dictionary is not intended to contain the precise dimensions of such standard symbols.

For example, the concept 'centrifugal pump symbol' and 'globe valve symbol' as well as their relations with the concepts that they by definition represent will be part of the dictionary, but the exact geometry of those symbols is not defined in the dictionary.

10.7.11 Stage 11: Other kinds of relations

Optionally one or more relations with a specific property value, algorithm or other concept. These additional relations provide an explicit definition of the thing by qualifying an aspect. Such an explicit definition should be specified as follows:

... is by definition ...(property value-1)

For example: plastic vessel is by definition made of plastic.

10.7.12 Stage 12: Add contextual facts

To complete a definition model, each main fact shall be accompanied by a number of contextual facts. For example, a status, date of creation, author, source, etc. A full list of contextual facts are defined in the definition in the document 'Gellish Syntax and Contextual Facts'.

10.7.13 Stage 13: Information about a concept

Strictly speaking information about concepts does not belong to definition models. However, one might want to capture informative text or references for clarification. This can be done in two ways:

By referencing a document

By inclusion of a piece of text in the model.

10.7.13.1 Referencing a document

The definition of a document file is similar to the definition of a drawing or picture as described in the previous paragraph. Relation type 1911 expresses that the concept is described by the information, which is the document content that is contained in one or more physical files. This is illustrated by the example below.

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
110	API 617.pdf	220	1,225	is classified as a	490,533	electronic data file
110	API 617.pdf	221	1,227	is an element of	107	http://
107	http://	210	1,225	is classified as a	970,274	URL address
111	API 617	223	1,726	is a qualification of	910,137	standard specification
111	API 617	224	4,996	is presented on	110	API 617.pdf
130,069	compressor	225	1,911	is a concept that is described by	108	API 617

Table 23, Referencing a document

10.7.13.2 Inclusion of text in the model

Instead of a reference to a document it is also possible to include the text directly in the model. This is done by inclusion of the text in the ‘full description’ column on the line where the information is defined by a qualification as information or as a subtype of information (in the example below the text is a qualification of ‘requirement’).

UID of left hand object	Name of left hand object	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	Full description
130,069	compressor	5,298	shall be compliant with	5	API 617	
5	API 617	1,726	is a qualitative subtype of	970,002	information	
130,069	compressor	5,298	shall be compliant with	6	Par. 5.3.1	
6	Par. 5.3.1	1,726	is a qualitative subtype of	970,007	requirement	The purchaser and the vendor shall mutually determine the measures that must be taken to comply with any governmental codes, regulations, ordinances, or rules that are applicable to the equipment.

Table 24, Inclusion of text in a model

10.7.14 Stage 14: Documenting definition models in a standard format

The above information, expressing main facts and their contextual facts shall be recorded in a standard format. This can be for example an RDF/XML format or the Gellish Expression Format.

An example of a definition is a cooling system, which is defined as a system that is intended to be suitable to cool something. The concept ‘cooling system’ with this definition can be added to the dictionary by specification of a line in an Expression Table of which the core is as follows:

UID of LH	Left hand object name	UID of fact	UID of Relation type	Name of relation type	UID of RH	Right hand object name	Description
101	cooling system	102	1146	is a specialization of	103	system	intended to be suitable to cool something.

A kind of thing (a concept) should only be included in a formal dictionary when it defines that a collection of things have common characteristics and/or have a common behavior. Generally known individual things, such as geographic and geopolitical objects, such as rivers, mountains, countries and cities are also proper candidates for the general dictionary.

10.8 Definitions through usage of templates

Definitions and knowledge can be expressed directly in Formal English, but there is also a simplified method available that can be used without in-depth knowledge of the Formal English language. The latter method is illustrated in Figure 102.

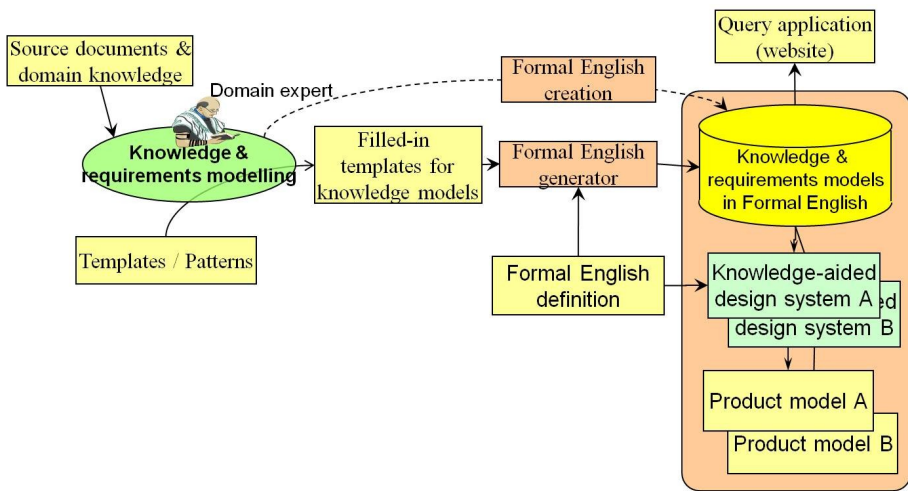


Figure 102, Creating and using definition models and knowledge models

The simplified method provides the discipline specialist with a template with fill-in-the-blanks options as an aid to model his discipline domain knowledge. Such a template is always the same, it is independent of the subject that is modeled, whether it is domain knowledge about a road, a bridge, a sewage or process equipment: the method to document definitions and knowledge is always the same.

Each of the earlier described stages in the modeling process results in models that are characteristic for that stage. Therefore, each stage requires its own templates. This means that there are definition templates and various kinds of knowledge templates, as well as requirements templates. When a domain expert has completed filling-in some templates, then a Formal English generator is used to create a database with knowledge models. That generator is a computer program that is able to interpret the template and that uses Formal English definition to create a correct Formal English database.

The stages in the Gellish Modeling Methodology with the accompanying templates guide the discipline specialist through the various parts of the architecture while modeling his or her knowledge about the kinds of objects.

In this way it is relatively simple to develop usable knowledge models about kinds of objects that belong to the knowledge domain of the specialist, without the need for the specialist to become a specialist in a formal language.

A template has the form of a table with predefined columns, as is illustrated in Figure 103.

10.9 Implementation of taxonomic dictionaries

Taxonomic dictionaries that are created according to the Gellish Modeling Methodology can be stored in a system independent way. For example in a triple store or in a Gellish database that uses a standard Gellish Expression Format [Ref. 4]. This has as consequence that the dictionary definitions and identifiers can be imported in any system that is able to read such neutral files. That is the reason why this document does not discuss any particular software system, but only deals with their data and document content.

10.10 Quality rules for taxonomic dictionaries

This paragraph specifies quality rules for formal expressions (main facts with contextual facts). It assumes that the expressions are presented in one or more Gellish Expression Format tables or an agreed equivalent format that is a representation thereof.

Rule 1: Only use of properly defined concepts and relation types.

In each relation between concepts (kinds) the concepts shall be:

- Either selected from the formal dictionary,
- or
- Properly defined as a subtype of a concept in the formal dictionary, whereas the used concepts are related by relations that are classified by kinds of relations (relation types) that are also selected from the formal dictionary.

How to verify rule 1:

- 1 Verify for each used concept whether the concept (UID) has a sequence of specialization relations (including also subtypes of specialization relations) with more general concepts, which sequence, when following the hierarchy, terminates at the concept “anything”. If the top concept is not “anything”, then an error shall be generated including a report about the concepts that actually terminates the sequence.

This can be verified using the built-in verification rules of software that imported a formal language defining taxonomic dictionary-ontology.

- 2 Verify for each used relation type whether the relation type exists in the formal dictionary (upper ontology section). Proprietary extensions of relation types shall be verified separately.

This can be verified using built-in verification rules in software that imported a formal language defining taxonomic dictionary-ontology.

Rule 2: Proper definition.

A subtype of a concept is properly defined if it is described by a textual description that is a text string that expresses that the subtype is a specialization of its direct supertype concept and by which aspects and aspect values it is distinguished from its supertype and from its ‘sister’ subtypes. Instead of such an explicit textual definition it is also allowed (and in particular business contexts required) to specify modeled definitions. In a modeled definition the supertype is specified to have subtypes that are defined by a particular kind of aspect, whereas the value of that aspect is specified for each subtype. An ideal modeled definition can be used to generate a textual definition.

How to verify rule 2:

Verify the text in the full description column by manual inspection.

Rule 3: Grammatical correctness.

Each relations between things is classified by a kind of relation (relation type) as specified in column 60 and 3 of an Expression Formal table. Such a kind of relation defines the kinds of roles that are played by the related things in a relation of such a kind. Each role player shall be able to play the role that it is required to play according to the kind of relation. The (required or specified) nature of a role player shall not be in conflict with the nature required by other relation types in which the role player plays a role.

The formal dictionary includes “facts about conceptual relations” that define the relation types of the formal language. These definitions also specify the required kinds of roles and the kinds of things that can play those kinds of roles (which implies that also their subtypes can play those roles).

Example 1:

The expression C is a specialization of D is only correct if C is a concept. This conclusion follows from the formal definition of the <is a specialization of> relation, which requires as first role a subtype, whereas a subtype role can only be played by a concept (or by a subtype of concept). The relation type requires as second role a supertype, which can also only be played by a concept. So, D must also be a concept.

Example 2:

The expression “A is by definition a performer of a B” is only correct if A is a subtype of physical object, because the definitional statements in the formal dictionary specify that an <is by definition a performer of a> relation requires as first role a performer and it also specifies that a performer role can only be played by (a subtype of) physical object. The relation type requires as second role a ‘performed’, which is a role that can only be played by (a subtype of) occurrence. So B must be (a subtype of) occurrence. These conclusions about the nature of A and B shall not be in conflict with the nature required by other relation types in which A or B plays a role.

How to verify rule 3:

Verify the consistency of the object type of the left hand and of the right hand object according to various relations in which those objects are involved. This can be done as follows: When something is related to another thing by a relation of a particular kind, then it can be derived from the definition of the relation type in the formal dictionary what kind of thing it must be. This can be determined as follows: according to the upper ontology section of the formal dictionary the relations of a particular type each require a particular first kind of role and a second kind of role. This is either directly specified for the relation type or it is inherited from its supertype relation type. The formal dictionary also specifies which kind of thing can play such a kind of role. This can also be specified either directly or by inheritance from its supertype kind of role. When the object is a left hand object and a standard phrase for the relation type is a normal phrase (or both the opposite), then the thing shall be of the kind of thing that can play the first kind of role. When it is a right hand object or an inverse phrase then it shall be of the kind of thing that can play the second kind of role, etc.

Conclusions about the kind of thing that are derived from various relations for something and its explicit classification relations or specialization relations (or their subtypes) shall not be in conflict with each other. There is no conflict if the conclusion about the kind of thing drawn from the classification or specialization relation(s) is the same or a subtype of the kind that is required by a relation type.

This should be verified by the built-in rules in software.

If the specialization hierarchy of concepts is incomplete (rule 1 is not satisfied), then this verification is impossible.

Rule 4: Semantic consistency.

The relation types that classify the relations in formal expressions determine the nature of the things that play the roles in the relation. Those natures shall be consistent with the natures that are required by other relations in which the same thing plays a role.

Example 1: Inconsistency detection.

From the expression C is a specialization of D it can be concluded from the formal definition of the specialization relation that C is a concept. If there is another expression that states that C is a part of E, then from that relation it can be concluded from the formal definition of the composition relation that C is an individual thing. As C cannot be a concept and at the same time be an individual thing. Therefore, it can be concluded that these two expressions are inconsistent.

Example 2: Inconsistency detection.

From the expression A is a performer of B it can be concluded from the formal definition of the ‘being performer’ relation that B is an occurrence. If there exists another expression that states that B is a performer of F then the two expressions are inconsistent, because B cannot be an occurrence as well as a physical object.

How to verify rule 4:

There is no conflict between a conclusion about the kind of thing drawn from one relation type and a conclusion drawing from another relation type for the same thing, when the required kind of thing is the same or is a subtype or supertype of the kind that is required by the other relation type. In other words they are in conflict when the conclusions about the kinds of things belong to different branches of the specialization hierarchy.

This can be verified by the built-in rules in software.

Rule 5: Inheritance of constraints.

All constraints (facts that are by definition the case) that are valid for members of a supertype concept shall also be valid constraints for any of the members of its subtype concepts.

Example 1: inheritance of defined components

If a car has by definition a motor, then there may not be subtypes of car that do not have by definition a motor and there may not be individual (well-formed) cars without a motor.

How to verify rule 5:

Whether a relation is a constraint is determined by the relation type that classifies the relation.

If a constraint (or more generally a fact) is redefined for a subtype, without or with identical cardinality constraints (see rule 7), then the redefinition shall be reported and its status shall be changed into ‘inherited’. This shall imply that the fact is ignored, because the constraint shall be inherited from its supertype concept. However, the software may verify whether facts with the status ‘inherited’ are indeed inherited from a specification for a supertype, as facts about the supertypes might have been changed so that the inheritance is not the case anymore.

Rule 6: Inheritance of cardinality constraints.

The cardinality constraints of a subtype concept shall be equal or narrower than the cardinality constraints of its supertype concept.

How to verify rule 6:

Software shall verify whether minimum and maximum cardinality constraints that are defined for a concept are narrower than possible definitions of cardinality constraints for the same relation type for its supertypes. It is incorrect if a cardinality constraint of a subtype is wider than the cardinality constraint of its supertype.

Rule 7: Completeness of contextual facts.

A data exchange message that is compliant with the Gellish Expression Format either in tabular or equivalent form shall include data that is consistent with the column definitions according to one of the subsets that are defined in the document “Gellish Syntax and Contextual Facts - Definition of Universal Semantic Databases and Data Exchange Messages” [Ref. 4]. Each field that corresponds with a field on a row in such a table shall have a value that complies with the rules defined in that document.

How to verify rule 7:

Software shall verify whether a value in a field is present when that is obligatory and whether the value satisfies the format constraints for the field.

10.11 Summary of relation types for definition models

The following relation types are used to create a core of definition models as described in this section. Additional relation types will be needed to add

other kinds of requirements in the form of data or when the content of documents is modeled.

Note that each relation type has an inverse expression. When the inverse expression is used the left hand object and right hand object shall be exchanged.

UID of Relation type	Name of relation type	Description
1,446	is a specialization of	is a relation between two concepts that specifies that the first concept is a subtype of the related second concept (the supertype). This means: what is true about the supertype concept is also true (by inheritance) for the subtype concept. The subtype concept is distinguished from the supertype concept by additional constraints, which are also called distinguishers. Example: motorway <is a specialization of> road
5,652	has subtypes that have as distinguishing aspect a	is a relation between a concept and a conceptual aspect that specifies that the concept has subtypes, which subtypes are distinguished from the concept because they have by definition a particular value for the conceptual aspect. Example: instrument <has subtypes that have as distinguishing aspect a> measured property
5,317	is by definition a possessor of a	is a relation between a concept and a kind of aspect that specifies that the concept is defined by having that kind of aspect (by definition). Example: pipe <is by definition possessor of a> pipe diameter
5,283	is by definition qualified as	is a relation between a possessed aspect concept and a qualitative aspect that specifies that the possessed aspect is qualified by the qualitative aspect (by definition). Example: construction material of a copper bolt <is by definition qualified as> copper
1726	is a qualification of	is a relation between a conceptual aspect and a qualitative aspect that specifies that the qualitative aspect has the conceptual aspect as its nature. Example 1: red <is a qualification of> color Example 2: this text <is a qualification of> information
5,492	shall be quantified on scale as	is a relation between a possessed aspect concept and a numeric value that specifies that the aspect has a value that is equal to the numeric value when being quantified on a particular scale (which scale is specified separately) Example: diameter of an M6 bolt <can be quantified on scale as> 30 mm
5,493	shall be quantified as greater than	is a relation between an aspect and a numeric value that specifies that the aspect has a value that is greater than the numeric value when being quantified on a particular scale (which scale is specified separately)
5,494	shall be quantified as less than	is a relation between an aspect and a numeric value that specifies that the aspect has a value that is less than the numeric value when being quantified on a particular scale (which scale is specified separately)
4,717	has as an intended function a	is a relation between a concept and a kind of activity or process that specifies that the concept is defined so that instances of that kind are intended to be suitable to be performer of activities or processes of the specified kind. Example: compressor <has as an intended function a> compression
5,519	has by definition as part a	is a relation between two concepts that specifies that a member of the first concept has by definition one or more parts that are member of the second concept. Example: car <has by definition as part a> engine

1,981	is a synonym of	is a relation between two text strings (names) that both denote the same concept, that specifies that the first text string is a synonym of the second text string, whereas that first text string originates in the context for the first text string.
4,691	is a translation of	is a relation between two text strings (names) that both denote same concept, that specifies that the first text string is a translation of the second text string, whereas that first text string originates in the language (and context) for the first text string.
1,983	is a code for	is a relation between two text strings that both denote the same concept, that specifies that the first text string is a code for the second text string, whereas that code originates in the context for the code.
1,982	is an abbreviation of	is a relation between two text strings that both denote the same concept, that specifies that the first text string is an abbreviation of the second text string, whereas that abbreviation originates in the context for the abbreviation.
1,225	is classified as a	is a relation between an individual thing and a kind of thing (concept) that specifies that the individual thing is a member of the kind. Example: file-1.doc <is classified as a> electronic file
1,227	is an element of	is a relation between an individual thing and a collection of individual things that specifies that the individual thing is an element of the collection. Example: file-1 <is an element of> directory-1
4996	is presented on	is a relation between information and an individual physical object that specifies that the information is expressed on the physical object. It is expressed as an aspect of the physical object. For example as a spatial (shape and pattern) aspect of ink on paper. Example: pictogram-1 <is presented on>file-1.dwg
1,911	is a concept that is described by	is a relation between a concept and qualitative information that specifies that the concept is described by the qualitative information. Note that the qualitative information can be graphical or textual. Example: motorway <is a concept that is described by> pictogram-1

11 Knowledge modeling

11.1 Introduction to knowledge modeling

There are large benefits to be gained when knowledge and requirement specifications become available in a computer interpretable way. Those benefits arise when documents are converted to data that can be read and interpreted by computers. It means that large volumes of textual specification are replaced by databases that enable that software can search for and present knowledge and requirements from different sources in a consistent and integrated way at the right moment in the design process. It also means that such specifications can be directly imported in design systems and that it can be reused during computer-aided verification of designs. This will lead to a faster design process and an increased quality of the designs.

This section provides guidance on the modeling of knowledge and requirements in a computer interpretable way for application and reuse of that knowledge and those requirements.

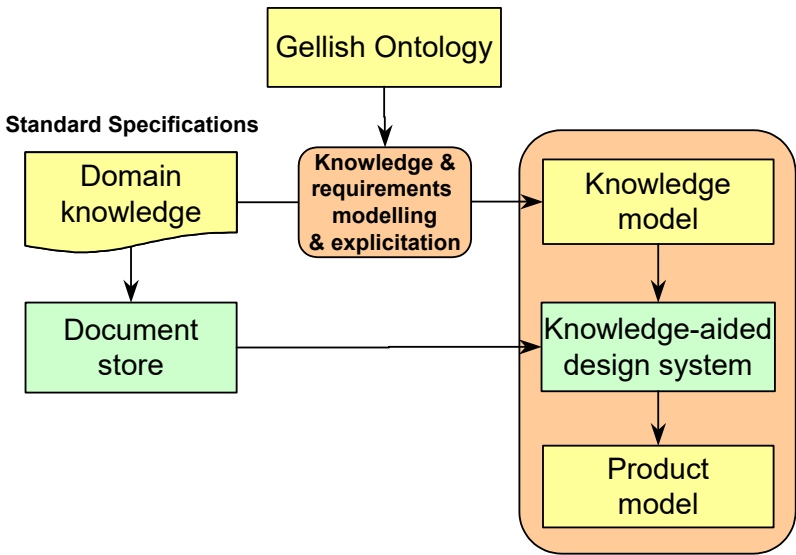


Figure 104, Knowledge modeling for knowledge-aided design

This section provides guidance on the modeling of knowledge and requirements in a computer interpretable way for application and reuse of that knowledge and those requirements

illustrates the knowledge modeling process in which domain knowledge is partly converted into knowledge models that are expressed in a formal language, using a formal language definition (taxonomic dictionary-ontology), and is stored in databases whereas other domain knowledge is maintained as computer accessible documents. Sources of knowledge are typically the standards documents that are used in organizations. In that conversion process knowledge about kinds of processes and functions and about kinds of facilities and equipment and their components is expressed in formal expressions. Knowledge modeling thus includes transforming knowledge and standard specifications into computer interpretable expressions of possibilities and requirements that are integrated in knowledge models and requirements models of kinds of things. A professional knowledge modeler may supervise this process and may manage the quality of the resulting knowledge models.

The figure also illustrates the next step in which that knowledge that may originate from various different sources is retrieved and imported in design systems where the modeled knowledge can be presented to its users and can be used to provide guidance on how to design facilities, equipment or processes.

11.2 Expression of knowledge

Knowledge expressions specify possibilities, which are facts that *can be the case* about a particular kind of thing. Such possibilities may be *typically* or *normally the case*, but they are not always the case for all things of that particular kind. For example, a kind of knowledge about things specifies how things of a particular kind are normally composed. Such knowledge results in a composition hierarchy or composition tree. Other examples of kinds of knowledge are knowledge about the intrinsic aspects of things of a particular kind, or the roles that such things can play, or the processes, events and activities (occurrences) in which they can be involved. All recorded knowledge about things of a particular kind is together called a knowledge model of the kind of thing or of the kind of occurrence.

The expression of knowledge in a formal language creates collections of expressions of possibilities that consist of relations between kinds of things. Those kinds of things or concepts should already be defined in a formal dictionary.

Each expression relates two concepts by a relation of a kind that is represented by a phrase that typically starts with <can have ...> or <can be ...>, such as

(an) A can be related to a B.

The phrase <can be related to a> is the name of a kind of relation that classifies the relation between A and B.

Such kinds of knowledge expressions have as meaning:

It is possible that an individual thing of the specified kind (A) has a relation of the specified kind with another individual thing of the related specified kind (B).

Table 25 presents some examples of knowledge that express of possibilities.

Intention	Name of left hand object	Name of relation type	Name of right hand object
statement	pump	can have as part a	shaft
statement	pump	can have as aspect a	capacity
statement	centrifugal pump	can have as part a	impeller

Table 25, Knowledge about a pump

The relations between the *individual things* that are possible according to such a statement are of a different kind than the kind of relation between the *concepts*, because the two kinds of relations have different meanings (a semantic difference). The definitions of concepts in the Gellish family of formal languages specify what kind of relations between individual things can be used for the realization of possibilities. For example, the upper ontology section contains specifications such as: a conceptual assembly relation <can be realized by a> assembly relation, etc. Or in Gellish Expression Format:

Intention	Name of left hand object	Name of relation type	Name of right hand object
statement	can have as part a	can be realized by a	has as part
statement	can have as aspect a	can be realized by a	has as aspect

Table 26, kinds of relations for the realization of possibilities

Each expression consists basically of a left hand concept, a relation type and a right hand concept. Such a fact can be graphically represented in its simplest form as indicated in Figure 105.



Figure 105, Graphical representation of an atomic possible fact about a kind of thing

The circle and the arrow in the figure indicate the direction in which the standard Formal English phrase that represents the relation type should be read. An arrow in the inverse direction can represent the same fact, provided that the inverse formal language phrase for the relation type is used.

Typically, but not necessarily, an expression is a statement. The fact that it is a statement is indicated by the intention of the expression. For example, the fact that a pump can have as part a shaft can be expressed in the simplest form of an Expression Table [Ref. 4] as follows:

Intention	Name of left hand object	Name of relation type	Name of right hand object
statement	pump	can have as part a	shaft

Table 27, Example expression of knowledge (a fact about a kind of thing)

Note that all three concepts <pump>, <can have as part a> and <shaft> shall be selected from a formal taxonomic dictionary or should have been defined and added as an extension of the formal dictionary.

11.2.1 Role of objects in relations

The left hand concept and the right hand concept each plays a role of a particular kind in the relation. Typically those roles are implicit, because the kinds of roles are defined as component of the definition model of the kind of relation. For example, a composition relation is defined as a relation that has by definition two roles players: one that plays a role as ‘part’ and the other

plays a role as ‘whole’. Sometimes the kinds of roles that are played by the left and/or right hand concepts are made explicit in the expression, as is illustrated in Figure 106.



Figure 106, Extended graphical representation of a possible fact

Such an extended expression can be presented in an extended version of the above table as follows:

Name of left hand object	Name of role of left hand object	Name of relation type	Name of role of right hand object	Name of right hand object
pump	(whole)	can have as part a	pump shaft	shaft

Table 28, Roles that are played by objects in a fact

The roles, such as the right hand role ‘pump shaft’ or ‘shaft of a pump’ (as synonym), are often important when constraints are specified that only apply to an object in such a particular role. For example, it often occurs that a particular constraint only applies to a part in the context of being part of a particular kind of whole. For example, a particular constraint might only apply to pump shafts and not to shafts in general. However, it may also be that a role, such as the left hand role in the table (whole), is not used elsewhere and thus such a role is normally left blank as long as there is no need to give that role an explicit name.

11.2.2 Cardinalities

It may be that there is a constraint on the number of individual things of the specified kind that are allowed to play the same kind of role at the same time in relation to one individual of the related kind. The minimum and maximum numbers are called the minimum and maximum simultaneous cardinalities. They may be constraints on the number of left hand objects or on the number of right hand objects. For example a well-formed centrifugal pump shall have at least one impeller, but can have maximally many impellers. So one instance of the concept ‘centrifugal pump’ can be related to 1 or more instances of ‘impeller’. The minimum and maximum simultaneous cardinalities are therefore 1 and many. If the centrifugal pump is represented as the left hand object, then these are the right hand cardinality constraints that are indicated as 1,n next to the right hand concept, impeller. This is illustrated in the following table:

Left hand cardinalities	Name of left hand object	Name of relation type	Right hand cardinalities	Name of right hand object
0,1	centrifugal pump	can have as part a	1,n	impeller

Table 29, Cardinalities of objects in a kind of fact

On the same line the left hand simultaneous cardinality constraints (0,1) specify that a particular individual impeller is either not part of a centrifugal pump or it is part of one centrifugal pump at a particular moment. This may be for example because it is a spare impeller in a warehouse, or it is an impeller for a centrifugal compressor. The cardinality constraints only constrain the number that occurs at the same time. They do not constrain occurrences in sequence. For example, the fact that an impeller that has been part of centrifugal pump may be part of another centrifugal pump at a later stage is not in conflict with the above cardinality constraint.

Subtype concepts inherit cardinality constraints from their supertype concepts. However it may also be that new cardinality constraints are specified for the subtype, which then shall be within the constraints of the supertype. For example, a single stage centrifugal pump has right hand cardinality constraints on the number of parts that are impellers of 1,1.

Cardinality constraints may only be specified for kinds of things. For individual things it may be specified that they have a number of parts of the same kind. There are two options to model those parts. 1. The individual parts are identified individually, each with its own UID and each related to the whole assembly by its own composition relation, or 2. It is specified that there exists a collection of individual things for which it shall be specified that its members are classified by the same kind and that collection is a part of the assembly. For example, the collection ‘impellers of P-101’ ‘each of which is classified as a’ impeller.

11.3 Contexts

Definitions are independent of a context or, in other words, they are applicable in a universal context. However any expression is expressed in a particular language. Therefore, the applied natural language shall be specified as the context for each expression. Language communities and disciplines usually have their own preferences for naming conventions, synonyms, abbreviations and codes. Therefore, next to the language, also a language community shall be mentioned as the context for the applicable of terminology of the name of left hand object. (The name of the right hand

object is free, as long as it occurs somewhere as left hand object name.) Facts that express general knowledge about well-formed items, such as possibilities for compositions or properties of items, are usually also applicable in a wide or universal context. However, specifications and especially requirements and constraints are only applicable under specific circumstances. Therefore, specifications and requirements shall be accompanied by a specific validity context. Furthermore, when knowledge is used it shall be specified which context(s) are applicable. For example, the expression in English that specifies the requirement that a compressor shall be compliant with a standard with the name ‘API 617’ is only applicable in the context of the company X who specified that requirement. Assuming that the term ‘compressor’ is a name given to a concept in the language community ‘rotating equipment engineering’, the expression becomes:

Language	Language community for left hand name	Context of validity of fact	Name of left hand object	Name of relation type	Name of right hand object
English	rotating equipment technology	company X	compressor	shall be compliant with	API 617

11.4 The knowledge models creation process

In principle, knowledge models consist of many kinds of facts. The most important types are facts about compositions and facts about the possession of aspects (characteristics, properties, qualities and performance). Others may concern with relations with activities in which they may be involved (such as testing or functions they may fulfill), fluids that they may contain or organizations that may supply them, etc.

This means that the creation of a knowledge model consists of the following three main steps:

1. Specify possible compositions
2. Specify possible aspects
3. Specify possible relations with other kinds of objects

11.4.1 Stage 1: Specify possible compositions

It may be that objects of a particular kind are by definition composed of a particular kind of component. In that case such a composition fact is a

defining composition that should already be included in the definition model of the concept.

Apart from these defining components there are kinds of objects that may or may not be a part of a whole assembly of a kind. These are the possible or optional components. The facts about those possible components are included in knowledge models.

A fact about a possible composition captures that a kind of object can have one or more objects of a particular other kind as parts.

For example: a bridge <can have as part a> pavement.

It is also possible that more knowledge about a composition of such a kind of object should be specified. For example, next to the fact that a bridge can have a pillar as a part, it may also be specified that a particular kind of bridge shall have a minimum number of pillars. For example: a bridge <can have as part a> 0,1 or more pillars.

Note that the specification of the composition of a kind of object (in other words: from which kinds of components a kind of object can be composed) is different from the specification of the subtypes of the kind of object. So the composition hierarchy is something else than the specialization hierarchy. This distinction is earlier described in paragraph .

11.4.2 Stage 2: Specify possible aspects

A knowledge model also contains facts about describing (optional) aspects that are not part of the definition.

This includes facts about aspects of the whole, but also aspects of the parts. For example: the fact that a road normally has a width¹⁴ is a fact about a whole road. Such a fact is expressed by a relation between the kind of object and the kind of aspect (characteristic). The example fact is expressed as follows:

road can have as aspect a width

When a design is made of an object of a kind, some of the aspects will be quantified by a numeric value, which is usually a value on a scale. That is the case for example for length, width, thickness, etc. Other characteristics are of a qualitative nature and will be qualified by a (standard) textual 'value'. For

¹⁴ In fact a road always has a width, but it may be the case that an information model about a road does not include that width or the width (value) of a designed road may not yet be known.

example, material of construction, such as a kind of steel, color, shape, operating principle, fabrication method, etcetera.

The formal dictionary includes definitions of such standard quantitative and qualitative values of kinds of aspects. Domain specific dictionary extensions are required wherever additional standard values are required.

It is possible to include a number of standard values in a collection of allowed values. This enables to specify in a knowledge model that for a particular kind of aspect (or for a kind of aspect in a particular role) only values that are selected from the collection are allowed.

It is possible that an organization specifies its own list of allowed values, as is described in the next chapter.

11.4.3 Stage 3: Specify knowledge about functions and occurrences

After the specification of facts that express knowledge that is implied in the definition of the function or occurrence it is possible to add expressions of knowledge about facts that are not by definition the case, but that is possibly the case. Such facts are expressed by phrases that start with “can be ...”.

Step 1: Specify performers and enablers

Specify which kinds of things (physical objects) are able to act as a performer (active) or enabler (passive) of occurrences of that type (if the performer or enabler is not yet specified as part of the definition of the activity, see step 2).

For example as follows:

pumping of waste water	can be performed by a	waste water pump system
transport of waste water	can be enabled by a	sewer system

Step 2: Specify subjects

Specify which kinds of things (physical objects) can act as a subject in the occurrence of that type (if the subject is not yet specified as part of the definition of the activity, see step 2).

For example as follows:

pumping of waste water	can have as subject a	waste water stream
transport of waste water	can have as subject a	waste water stream

Step 3: Specify parts

Specify the parts of the occurrence and/or of the performer (if required).

For example as follows:

transport of waste water	can have as part a	collection of waste water
waste water pump system	can have as part a	driver

Step 4: Specify the process steps

Specify the process steps that split, combine or treat streams by defining the streams that have a role as inputs and outputs of the occurrences (if required).

For example as follows:

cleaning of waste water	can have as input a	waste water stream
cleaning of waste water	can have as output a	clean water stream

Note that streams that are output of a kind of process can be specified as possible input to another kind of process. This specifies a topology of processes, such as typically presented on a process flow diagram or in a business activity model.

11.4.4 Stage 4: Specify other possible relations

There are a large number of other kinds of knowledge that can be specified. For example, possible connections between objects of particular kinds or relations between kinds of objects and kinds of documents. Other examples are kinds of objects that can have a particular role in a kind of activity, such as being subjected to a particular kind of test. These kinds of facts are not about aspects that are inherent to the kind of object, but they concern relations between a kind of object and a kind of activity or process.

In order to find a suitable kind of relation to express knowledge it is recommended to search for phrases that begin with <can be...> or <can have...> and that contain a kind of role that one of the involved things play. For example: <can... input> will deliver <can have as input a> as well as its inverse <can be an input for a>. Software should support such a search for kinds of relations, although it may also be helpful to execute an alphabetic

search in a formal dictionary of kinds of relations or a search for subtypes of a known supertype kind of relation.

11.5 Usage of knowledge for creation of design proposals

Knowledge in the form of knowledge models can be used for example for guiding design processes through generating various available options as design proposals. In that case, information about individual things is created on the basis of knowledge about a kind of thing. In other words facts about individual things are created on the bases of facts about concepts. Knowledge models include conceptual relations about particular kinds of things. Such statements support the design process as they provide suggestions on the kinds of relations that might be created. This is available because the formal language definition includes specifications of the kinds of relations that can be realizations of conceptual kinds of relations. For example, the formal language definition specifies that a <can have as part a> relation can be realized by a <has as part> relation. Thus the fact: a centrifugal pump <can have as part a> (1 to n) impeller(s) indicates that it can be realized by a number of <has as part> relations between an individual pump and its impeller(s). Thus the conceptual relations in knowledge models (as well as in requirements models) specify which kinds of relations can be realized for things of the related kinds.

The generation of a design proposal is illustrated by the following example. Assume that somebody wants to design a pump, which he calls P-1301 (say), whereas he already knows that it must be a centrifugal pump. To create a new individual pump he has to classify it. In this example that will be done by the following expression:

Intention	Name of left hand object	Name of relation type	Name of right hand object
statement	P-1301	is classified as a	centrifugal pump

Table 30, Creating new individual things by their classification

The design software can now search for knowledge (and definitions and requirements) in the database about a centrifugal pump. It will find the definition of the concept ‘centrifugal pump’, which at least specifies that it is a subtype of pump. Therefore the software will find also knowledge about the supertype ‘pump’ as well as other supertypes of centrifugal pump, which is inherited from those supertypes. Thus the software will find also knowledge such as expressed in Table 25. Together with the knowledge as expressed in

Table 26 on page 170. On that basis the software can then make proposals for statements about the individual P-1301 as follows:

Intention	Name of left hand object	Name of relation type	Name of right hand object	UoM
statement	P-1301	has as part	Shaft-1	
statement	Shaft-1	is classified as a	shaft	
statement	P-1301	has as part	Impeller-2	
statement	Impeller-2	is classified as a	impeller	
statement	P-1301	has as aspect	Capacity-3	
statement	Capacity-3	is classified as a	capacity	
statement	Capacity-3	has on scale a value equal to	?-1	m3/h

Table 31, Proposal for a design on the basis of a knowledge model

The proposed names of the new items can be generated from concatenating their classification and a sequence number. Instead of that the names can be generated conform the knowledge (and definitions and requirements) that was used. The resulting proposed specification then becomes for example as follows:

Intention	Name of left hand object	Name of relation type	Name of right hand object	UoM
statement	P-1301	has as part	P-1301-1	
statement	P-1301-1	is classified as a	shaft	
statement	P-1301	has as part	P-1301-2	
statement	P-1301-2	is classified as a	impeller	
statement	P-1301	has as aspect	P-1301-3	
statement	P-1301-3	is classified as a	capacity	
statement	P-1301-3	has on scale a value equal to	?-1	m3/h

Table 32, Statements about individual things

Thus a knowledge expression with the phrase <can have as part a> is realized by an expression about an individual thing with the phrase <has as part>, etc.

This process was implemented in the Gellish Communicator as is described in par. 6.2.8.

11.6 Knowledge-aided design proposals

Product design and the process of creating information models depends on the knowledge of the modelers about possibilities and requirements as well as on their ability to express designs in a formal language. To facilitate this process it is possible that software generates design proposals that are derived from knowledge that is stored in knowledge models and requirements models.

Facts that express knowledge and requirements (and definitions) about *kinds* of things can be interpreted as potential facts about individual things. For example, the fact that expresses the knowledge that a 'car' can have as part an 'air bag' means that there are potential facts about individual cars such as: 'car-123' has as part 'air bag-123'. The availability of such modeled knowledge implies that, when an individual thing is classified by a kind of thing, the software can determine potential facts about the individual thing.

The process to create proposals from knowledge is illustrated by Figure 107.

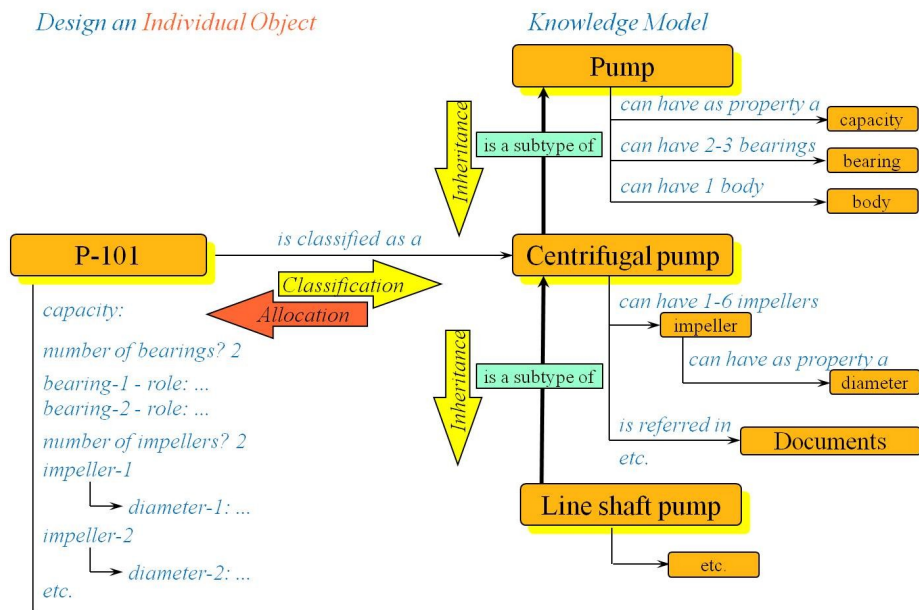


Figure 107, Example of allocation of knowledge to an individual thing

The right hand side of Figure 107 illustrates knowledge about pumps. For example, it states that a centrifugal pump is a subtype of a pump. Therefore, knowledge facts about a pump are inherited by the concept centrifugal pump. Additional knowledge facts are added that are specifically valid for

centrifugal pumps, such as the fact that such a pump can have 1-6 impellers with their properties. When an individual object, such as P-101 is classified as a centrifugal pump (is related to that concept by a classification relation), then the knowledge facts about that concept can be used to generate proposals for individual facts about P-101.

With another example we will illustrate how the facts are generated and expressed in a formal language. Assume that a standard query for knowledge about 'sliding door' results in the following facts that express (partly inherited) knowledge and requirements:

door	shall have as aspect a	width
door	can have as aspect a	material of construction
sliding door	can have as part a	2, 4 wheel
sliding door	is a specialization of	door

Furthermore, assume that D-101 is classified as a sliding door. Thus the first proposed relation is the following classification relation that follows from the knowledge:

D-101	is classified as a	sliding door

Other proposals for facts can be generated on the basis of the above facts that specify knowledge and requirements about a door, in combination with facts in the extended upper ontology section of the formal dictionary. For many relation types that can be used to express knowledge about a kind of thing, the extended upper ontology section states which relation type should be used to express a fact about an individual thing of such a kind. For example, the extended upper ontology section contains the following expressions:

can have as aspect a	can be realized by a	has as aspect
can have as part a	can be realized by a	has as part

In other words: a <has as aspect> relation type is a realization of a <can have as aspect> relation type.

If for a relation type there is no counterparty specified in the extended upper ontology section, then usually there is a counterparty specified for its supertype or higher level supertype.

Thus from the above knowledge and requirements the following proposals for facts about D-101 can be generated:

D-101	is classified as a	sliding door
D-101	has as aspect	width of D-101
width of D-101	is classified as a	width
width of D-101	has on scale a value equal to	? mm

D-101	has as aspect	MOC of D-101
MOC of D-101	is classified as a	material of construction
MOC of D-101	is qualified as	?
D-101	has as part	wheel of D-101
wheel of D-101	is classified as a	wheel

Proposals can be further enhanced in various ways:

Proposals should also deliver the possible subtypes of the counterpart relation types as alternative proposals. For example, the fourth relation type might also be <has on scale a value greater than>.

Proposals may present options for the values (indicated by question marks (?)). How options are specified is discussed later.

The options for unit of measure may be derived from the fact that width is a subtype of distance for which a ‘length scale’ is specified that is qualified by a list of scales. Furthermore a default unit of measure may be derived from a fact about the preferred scale for a unit (in a particular context).

It should be verified whether potential new objects and new facts exist already and whether they are in conflict with other existing facts (for example conflicting classifications). When they do not exist yet, then they should get proposed new UUIDs. For that reason it is required that ranges are specified from which the new UUIDs should be selected after verification of their non-existence.

The user of an application should have the option to accept, reject or modify the proposals.

12 Requirements modeling

12.1 Objectives and scope of requirements models

The objective of this section is:

To provide guidance for the unambiguous and computer interpretable expression of standard requirements about kinds of things in such a way that those requirements can be used to aid designing and verification.

From this objective, the following sub-objectives can be derived:

- To increase the quality of specifications of requirements about kinds of things and the ability to exchange, integrate and use such requirements by computer applications.
- To increase the ability for computer supported requirements aided design and to support the creation of Information Models.
- To increase the ability for computer aided verification of completeness and compliance of delivered information (documents and data) about a product.

In scope of this section are:

Requirements that are expressed for kinds of things, whereas those requirements are indirectly applicable for all individual things that can be classified by those kinds.

Requirements for kinds of things are requirements that are generally applicable for any individual that is of that kind. Requirements for kinds of things are typically expressed in standards, regulations and best practice guides, either as tabular data or as textual requirements. Usually a design for an individual thing includes various references to requirements for kinds of things. Those requirements include requirements for many standard kinds of components and their fabrication and testing as are described in national and international standards or in company specific standards. The modeling of all such kinds of requirements is described in this section.

The requirements that are applicable for kinds of things may be generally applicable, irrespective of their context, but usually they are only applicable in a particular context, such as when an authority or company specifies that a particular standard with requirements is applicable.

Examples of requirements that are covered by this part are:

- *Requirements for Information Delivery.* These are requirements that are expressed by a company that particular kinds of information (data and documents) are required to be delivered when certain kinds of components are delivered to that company. For example, a specification of the document types and types of properties per type of equipment that shall be delivered when such a piece of equipment is delivered.
- *Standard Requirements for Kinds of things or Product types* and their quality. These are requirements that are expressed in international standards about particular kinds of physical objects, materials of construction or processes. Delivered products shall be compliant with those requirements when the standard is adopted by being declared applicable in the project specifications. For example, the API 617 standard specification with requirements for compressors or requirements for heat treating of steel pipes.

In the Gellish Modeling Methodology, requirements are arranged in requirements models. Requirements models can be applicable for large or small assemblies or for components. For example, they can include requirements for a complete kind of process plant, a kind of building, a kind of road, a kind of ship or a kind of airplane, and they can be requirements for a kind of system, a kind of compressor, or for a kind of heat exchanger, or even for a kind of bearing, a kind of impeller, nut or bolt or any other kind of component. The requirements may also be about kinds of processes or kinds of activities in which the kinds of objects are involved. For example, they might specify how a particular type of test should be performed or how a commissioning or start-up procedure should be executed.

Requirements about a kind of thing are indirectly applicable for all individual things that are classified by that kind of thing or by one of its subtypes. Therefore such requirement models shall be suitable for use in computer-aided design of individual things and the requirement models shall be suitable also for use in computer-aided verification of designs. For example, a requirements model for centrifugal compressors expresses requirements that any centrifugal compressor shall satisfy when that requirements model is (contractually) applicable.

Requirements models may include a specification of the kind of information that shall be delivered as a description of a delivered product and its operation and maintenance.

This document specifies the procedure to create requirements models. It also discusses how those requirements models can be used to support the creation of designs of individual objects and it discusses how those requirements models can be used to verify delivered designs of individual objects. The latter implies that requirements models can be used to verify the completeness and quality of a Product Information Model.

Out of scope of this section (and in the scope of the section about information models) are:

- Requirements that are designs, which are specifications of individual things.

Requirements about individual things are requirements that are specific for one particular object (and possibly for some copies of it), for example requirements for a new product. Such requirements are typically expressed in a gradually increasing level of detail during the design stages. Early in a project they are expressed in the form of functional specifications or a front-end design, whereas at a later stage the requirements are expressed in the form of a detailed design. Such a design describes an imaginary individual object and it's functioning. A design is called an expression of requirements, because it expresses the requirements for a real individual object that shall be fabricated and constructed such that it reflects the imaginary one. Guidelines for the modeling of such requirements for individual things are given in the section about Information Models.

12.2 Inheritance of requirements

Requirements that are expressed for a kind of thing are inherited by all the subtypes of that kind of thing. For example, the requirements expressed for a vessel are inherited by a horizontal vessel and by a vertical vessel. This means that requirements that are specific for only one subtype, such as requirements that are specific for horizontal vessels, only need to be specified for that subtype, whereas requirements that are applicable for all subtypes can be specified for the supertype and don't need to be specified again for each subtype.

This mechanism can be applied in requirements models, because in a formal dictionary the concepts are arranged in taxonomy hierarchy, which means that it is obligatory for a formal dictionary that each concept is defined as a subtype of a more general concept. For example, the formal dictionary contains the following fact:

horizontal vessel is a specialization of vessel

This illustrates the importance of the maintenance and use of a good quality formal dictionary.

Note: System implementers should be aware that the above described inheritance is also applicable for relation types. For example, a requirement is a specialization of a possibility. Therefore, the relation type <shall have as aspect a> is a subtype of the relation type <can have as aspect a>. This implies that a realization relation might be specified only for a supertype of the required relation type. For example, the formal language might contain the following fact:

<can have as aspect a> <can be realized by a> <has as aspect>

This means that to search for the relation type that should be used to fulfil a requirement it is necessary to look also at the supertypes of the relation type, and to the subtypes of the fulfilling relation type.

12.3 Context dependent requirements

Requirements can vary from very broad specifications to a very specific requirement. A very broad requirement is for example the requirement that an item or product shall be compliant with the information contained in a complete document. An example of a very specific requirement is that a particular type of component, for example a shaft, shall be made of stainless steel or that it shall be heat-treated. Such requirements can be expressed as follows:

Applicability context	Name of left hand object	Name of relation type	Name of right hand object
Company X	centrifugal compressor	shall be compliant with	API 617
Company X	shaft	shall be made of	stainless steel
Company X	shaft	shall be a subject of a	heat treatment

As described above: each requirement has its applicability context. Thus the requirements in the above table only apply for Company X. However, even within a specific context of a particular company, not every shaft shall be made of stainless steel and shall be heat-treated. So, apparently those requirements only apply for a (every) shaft of a centrifugal compressor. This means that the assembly (the compressor) acts as the applicability context for the constraint. This shall be made explicit by specifying that a ‘shaft of a

centrifugal compressor’ shall be made of stainless steel¹⁵. The above table then becomes:

Name of left hand object	Name of relation type	Name of role of right hand object	Name of right hand object
compressor	shall have as part a	shaft of a compressor	shaft
shaft of a compressor	shall be made of		stainless steel
shaft of a compressor	shall be a subject of a		heat treatment

A requirement can also have the form of a constraint. An example of a constraint is the specification that a particular type of component shall have a diameter of at least 20 mm. This requirement can be expressed as follows:

foundation bolt of a compressor	shall have as aspect a	diameter
diameter	shall be at least	20 mm

Here the same issue occurs: not all diameters shall be at least 20 mm. Furthermore, the right hand side of this constraint is a property that is a qualification of distance. In practice it is preferred to separate the number from the unit of measure (scale). This is done by using another relation type. The result is as follows:

Name of left hand object	Name of relation type	Name of role of right hand object	Name of right hand object	Unit of measure
foundation bolt of a compressor	shall have as aspect a	diameter of a foundation bolt of a compressor	diameter	
diameter of a foundation bolt of a compressor	shall have on scale a value greater than		20	mm

In this way requirements can be constrained to things in particular kinds of roles.

¹⁵ In principle it is also possible to mention the assembly as the applicability context. However, in practice this leads to many applicability contexts that are difficult to manage. Therefore it is recommended to apply the rule that the role of the part is mentioned explicitly as is described in this guide.

The contexts within which requirements are valid can also be specified via a collection of facts (collection of requirements) so that all the elements in the collection are collectively declared applicable in a particular context.

A Gellish Expression Table contains a column (with UID and one for the name of the collection) in which the collection can be specified of which the fact is an element. It can be specified for the whole collection that it is united in a bigger collection, whereas each collection forms a data set that can be declared to be applicable in a particular context.

For example, a collection of requirements called ‘SGSI equipment data requirements for handover to operations’ can be declared to be applicable for a particular project.

12.4 Project requirements

Each project should specify which and how information shall be delivered at the end of the project and what the requirements for that delivered information are.

One of the recommended requirements for projects could be that the produced information shall be delivered in such a way that it is integrated in a (semantic) information model, possibly including also requirements models and knowledge models to enable the verification against requirements. That requirement implies that the scope of an information model should cover all components of the Information Model Architecture as was presented in Figure 10, Architecture section 6: Models of individual things.

This means that the delivery should include a Product and Process Information Model, integrated with a dictionary with definition models, a documents repository with document models, and possibly a collection of requirements models and knowledge models. The requirement also implies that the project shall deliver an information model that satisfies the requirements that are expressed in the requirements models. One of those requirements might be that the information model shall include relations that indicate which fact in the product and process information model satisfies which requirement in a requirements model.

If the requirements models about the components of a complete product are or shall be integrated in an information model, then this means that an information model shall be created and delivered.

12.5 Implementation of requirements models

This document does not discuss any particular software system, but only deals with their data and document content in a system independent way. This has as consequence that the requirements can be imported in any system that is able to read the Gellish Expression Format.

Kinds of systems in which requirements models will most likely be implemented are document-oriented systems, such as Electronic Document Management Systems (EDMS's), amended with data management capabilities, Content Management Systems (CMS systems) or Enterprise Content Management Systems (ECM systems). Other kinds of systems in which Information Models may be implemented are more data oriented systems, such as Product Data Management Systems (PDM systems) and Product Lifecycle Management Systems (PLM systems). The data storage capabilities of each of those systems may be extended with one or more files to enable storage of data that otherwise could not be stored.

12.6 Verification of delivered information

A formal (computer interpretable) specification of requirements enables to verify whether such requirements are satisfied when the deliverables are also expressed in a consistent formal way. In other words, when requirements are expressed in a formal language and the delivered information that intends to satisfy those requirements is also delivered according to the same language, then that enables computer-aided verification of the completeness and other quality aspects of the delivered information.

For example, it may be a requirement that when an object of a particular kind is delivered that then a particular kind of document shall be delivered as well and that the object shall have a particular kind of part for which spare parts shall be delivered. The formal language enables that such requirements are automatically verified, provided that the information about the delivered documents, parts and spare parts is also delivered in the formal language.

12.7 Conditional requirements

The above requirements seem to be unconditional, except that they are based on the content of an authoritative document, typically a regulation or law or standard, whereas that content shall be declared applicable for a particular organization, or a particular situation or project. However, semantically they

are quasi-unconditional, because they imply an implicit condition as is illustrated in Figure 108.

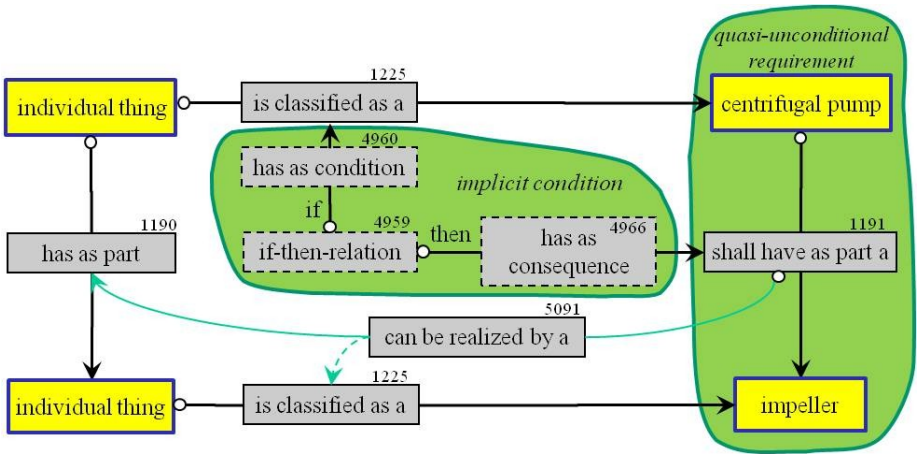


Figure 108, Condition and consequences of a quasi-unconditional requirement

The right hand side of Figure 108 illustrates the general requirement that a road <shall have as part a> lane. Such a general requirement is typically expressed by an authority and described in a standard. Therefore the requirement is only valid (applicable) in cases where the authority rules. This ‘validity context’ can be specified as an contextual fact in a separate column in an Expression Table. So, within the specified context the requirement holds for any road, which means that it is only applicable for an individual thing under the condition that the individual thing <is classified as a> road. This condition and its consequence are illustrated by the implied (dashed) if-then-else relation in the middle of the figure. When it is concluded that the requirement is applicable, then a <can be realized by a> relation points to a kind of relation that shall be found or shall be created in order to satisfy the requirement. The realization relations are specified in the extended upper ontology part of the formal language taxonomic dictionary . In this example the requirement is satisfied by a <has as part> relation between the individual road and a part, whereas the part shall have and <is classified as a> relation with the concept ‘lane’. For humans this is rather trivial, but for computers it is precisely the process that shall be followed by software in order to determine which requirements are applicable for which objects and what kind of facts shall be found or created.

There are also conditional requirements that have one or more explicit conditions. One or more consequences shall follow when the condition(s) are

satisfied, whereas other consequences shall possibly follow when the condition is not satisfied. These explicit conditional requirements are modeled by so called (qualitative) if-then-else relations. The general model of such a conditional requirement is given in Figure 109.

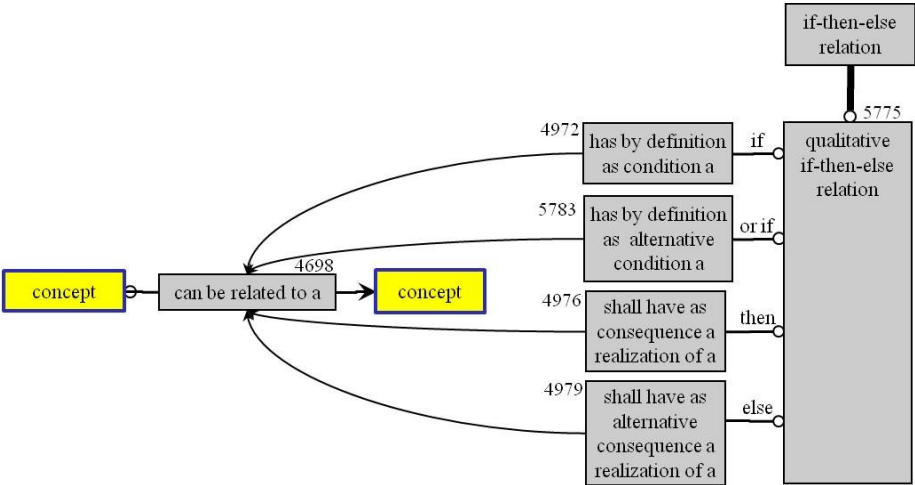


Figure 109, General model of a conditional requirement

Figure 109 illustrates that a general conditional requirement is modeled by a conceptual if-then-else relation that is defined by one or more conditional relations between the if-then-else relation and one or more conceptual relations (<can have ...> or <can be ...> relations). When the conditions are satisfied, then one or more consequence relations can point to the kinds of relations that shall be realized. Optionally there are also one or more consequence relations that shall be realized when the conditions are not satisfied. An example of a conditional requirement is given in paragraph 12.9.8, where the conversion of textual requirements to models is discussed. The process to verify whether the requirements are satisfied is similar to the above described process for ‘quasi-unconditional’ requirements.

Note that all the expressions of conditions and consequences that are involved in the same qualitative if-then-else relation belong together. In order to explicitly group them it is possible to group them together as elements in a ‘collection of statements’. The Gellish Expression Formal tables contains a separate column for recording the collection to which a statement belongs.

To interpret an if-then-else- relation correctly the following rule should be applied:

If a concept is appears more than once in the conditions and consequences of the same if-then-else relation, then in each case the same instance of the concept is meant.

For example, in the example of paragraph 12.9.8 there are three condition about aspects of a rotor system. The above rules states that these three conditions are by definition conditions about one and the same rotor system.

If this is not intended, then individual variables shall be created to distinguish the instances, whereas the conditions shall apply to the different (local) variables.

12.8 Integration of requirements models in Information Models

12.9 Requirements model creation procedure

The content of a requirements model will depend on the perspective of the party that creates the specifications. Also the complexity of the kind of thing and the number of options that are possible determine what will be included in the requirements models. A purchaser who regularly acquires the same type of objects may specify standard requirements for those kinds, and for some aspects he may only specify which kind of information he wants to be delivered together with the delivered object. A manufacturer who fabricates standardized components may specify those kinds of components in much more detail.

The process to create an electronic requirements model for a kind of assembly typically consists of the following steps:

1. *Define the concept (for which the requirements are applicable)*

This may include the definition of a subtype of a concept, in case the requirements are a specification of a standard subtype of a kind of thing. For example, define the concepts M6 bolt or Volvo S40 or Bus stop type 1, when a standard specification is made of requirements for an M6 bolt or for a Volvo S40, or for a Bus stop type 1 (e.g. according to handbook bus stops in the Netherlands).

2. *Create a requirements model*

This includes the definition of a collection of facts and its relation with the concept about which the collection contains requirements.

3. *Specify the obligatory components of an assembly.*

This includes a specification of the kinds of components of which assemblies of a kind shall be composed and the minimum and maximum number of those components that shall be used.

4. *Specify the required kinds of aspects of kinds of physical objects.*

This includes a specification of the kinds of aspects for which values shall be delivered and by which the aspects of all physical objects of that kind shall be specified.

5. *Specify the allowed values for the required kinds of aspects.*

This includes a specification of the values that the kinds of aspects (properties and qualities) of the kinds of physical object shall have or of the values that are allowed for aspects of those kinds.

6. *Specify the textual requirements*

This includes a specification of the textual requirements and constraints as well as the documents that contain additional requirements for a kind of assembly or for a kind of component.

1. *Relate documents to files at addresses*

Specify for each document that it is presented on a physical medium, either as an electronic data file or on paper, microfilm or any other medium.

2. *Model textual requirements when required.*

This means that textual requirements are converted into a computer interpretable model of the same requirements, using standard concepts and relation types. Such a model of requirements should be applicable for verification of the satisfaction of the requirements or for guidance on the creation of facts that satisfy the requirements.

7. *Specify requirements about kinds of documents*

8. *Specify requirements for kinds of processes and activities (functions).*

12.9.1 Step 1, Definition of a subtype concept

Standard Specifications are typically only valid for particular subtypes of concepts, also called standard design solutions. Therefore, those subtypes should first be defined in a way that is similar to the way in which any concept shall be defined.

In summary for example the definition of a bus stop type 1 (according to a Standard Specification document for bus stops in the Netherlands) shall have as a minimum definition the following facts:

UID of left hand object	Name of left hand object	UID of Fact	UID of Relation type	Name of relation type	UID of right hand object	Name of right hand object	Full description
1	bus stop	201	5,652	has subtypes that have as distinguishing aspect a	4	location	
105	bus stop type 1	202	1,726	is a qualification of	1	bus stop	is a bus stop that is located in a centre of a city
105	bus stop type 1	203	5,403	is by definition located at a	5	city centre	

Note that this definition relies on the fact that the concepts bus stop, location and city centre are already defined in the dictionary.

12.9.2 Step 2, Create a collection of requirements

The collection of facts that forms a particular requirements model shall be represented by a UID, it shall have a name and it shall be defined as a qualification of a collection of requirements. For example, a requirement model for a bus stop type can be expressed as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of Relation type	Name of relation type	UID of right hand object	Name of right hand object	Full description
100	requirements for bus stop type 1	204	1,726	is a qualification of	970,283	collection of requirements	is a collection of requirements for ... according to ...
100	requirements for bus stop type 1	205	5,658	are requirements for a	105	bus stop type 1	

Note that the relation type <are requirements for a> is used, which is the inverse expression for <shall be compliant with requirements> that was used in Figure 7.

The specification that a particular facts is an element of the collection is provided on the line where the fact is created by filling in the UID and name

of the collection in column ‘UID of collection of facts (column 50) and column for the name of collection of fact (column 68) respectively.

12.9.3 Step 3: Required components of an assembly

After defining kinds of physical objects and specifying knowledge about them additional requirements can be added by creating expressions that specify what shall be the case in a particular context but is not by definition the case.

Most kinds of physical objects are assemblies of a number of various kinds of components. Knowledge about their possible composition can be specified by defining the kind and number of components of which they can be composed. But organizations may also specify that in their context it is required that objects of specific kinds have components of specific kinds. Those kinds of components can be rather generic, but they may also be very specific, such as a specification of requirements for particular manufacturer’s models as components. The requirements that objects of a particular kinds shall have parts of particular kinds are expressed by <shall have as part a> relations. For example, the statement that a car *can* have zero, one or two airbags and the statement that cars for company ABC *shall* have two airbags can be expressed as follows:

Name of left hand object	LH card	Name of relation type	RH card	Name of right hand object	Validity context
car	0,1	can have as part a	0,2	airbag	
car	0,1	shall have as part a	2,2	airbag	Company ABC
pump	0,1	shall have as part a	2,2	bearing	Company ABC

Each kind of part may have further parts. The specification of all the options for parts and parts of parts forms a decomposition hierarchy that forms the core of a knowledge model of the assembly. Such a knowledge model about a kind of product can be used as a pick list or checklist for the design of individual products of such a kind of physical object.

Note that the kinds of parts themselves all need a definition in the dictionary. In many cases the part is specified through the specification of a role of a part. A <can/shall have as part a> relation may be used to indirectly specify a part through such a specification of its role. For example, instead of specifying that a pump shall have two bearings it is more precise to specify that a pump shall have a driven end bearing and a non-driven end bearing,

where the right hand object is not a physical object, but a role. This is done as follows:

pump	0,1	shall have as part a	1,1	driven end bearing
pump	0,1	shall have as part a	1,1	non-driven end bearing

It is required that those roles are defined in a way as is specified in par. 12.9.1.

Specify subtypes of kinds of parts

In many cases there are various options for kinds of parts. For example, a pump can have a number of bearings, but for each kind of bearing it is possible to choose from various subtypes of bearing. Two examples of definitions of subtypes are:

ball bearing		is a specialization of		bearing
needle bearing		is a specialization of		bearing

Each of those subtypes can be further decomposed and/or can have further subtypes.

Another example in which also UUIDs are shown is a particular type of bus stop:

105	bus stop type 1	22 0	1,72 6	is a type of	44,136	bus stop
105	bus stop type 1	22 1	4,98 9	shall have as part a	40,137	platform

It is also possible that there are soft requirements, or options. For example possible components that are unnecessary, are desired or that are required when certain conditions are satisfied. For the specification of such options the following relation types are available:

105	bus stop type 1	222	5,667	can conditionally have as part a	107	fence
105	bus stop type 1	223	5,665	can have as desired part a	8	bike garage
105	bus stop type 1	224	5,666	can have as unnecessary part a	9	toilet facility

The inheritance rules imply that these requirements also hold for possible subtypes of the concept for which the requirements are specified. For example, requirements for a centrifugal pump in general are automatically

(by inheritance) applicable for each horizontal centrifugal pump, vertical centrifugal pump, line shaft pump, high speed integrally geared centrifugal pump, etc. and for the subtypes of subtypes.

It may be that more than one component of the same kind is required. For example a multi-stage centrifugal pump requires more than one impeller. Such a requirement can be made explicit by specifying the minimum and maximum number of components of the same kind that shall be part of an object of a kind. This is called a specification of the minimum and maximum simultaneous cardinalities. Note that the precise requirement is that multiple components shall be part of the assembly *at the same time*, because due to replacement it is always allowed that several components of the same kind are part of the same assembly one after the other.

This can be specified as follows:

130,058	centrifugal pump	225	4,989	shall have as part a	130,144	1,n	impeller
130,180	multi stage centrifugal pump	226	4,989	shall have as part a	130,144	2, n	impeller

In these expressions 1 and 2 are the minimum number and n is the maximum allowed number of simultaneous parts of the same type, whereas n indicates that the maximum is unlimited.

Requirements for components in roles or at particular locations

Often the specification of requirements does not refer to a pure kind of component, but to a kind of object in a particular kind of role or to a kind of object in a particular location. For example, a driven-end bearing or a first stage impeller. The requirements about such kinds of roles of objects are specified in the same way as requirements for kinds of components themselves.

The advantage of the specification of kinds of roles of objects is that it enables to specify additional requirements that are only applicable for objects of a kind that are in such a particular kind of role.

Satisfaction of the requirement to have a part

A requirement for a part that is expressed by a <shall have as part a> relation is satisfied by a <has as part> relation, where the part is classified by the kind of part. For example:

P-1201

has as part

I1

provided that

P-1201	is classified as a	centrifugal pump
I1	is classified as a	impeller

The requirement may also be satisfied by an object that is classified by a subtype of centrifugal pump and by a part that is a subtype of impeller. For example:

P-1201	is classified as a	line shaft pump
I1	is classified as a	pump impeller

or even by a classification by a manufacturer's model, provided that that model is a subtype of centrifugal pump. For example as follows:

P-1201	is classified as a	Elliot M234

whereas the manufacturer's model is defined as a subtype (of a subtype) of centrifugal pump. For example:

Elliot M234	is a model of	horizontal centrifugal pump

Satisfaction of requirements to have components in particular roles

A requirement that an assembly shall have as part a kind of component in a particular kind of role can be satisfied by a fact that can be expressed in more than one way. The recommended way is to specify that an individual object has a part whereas that part is classified by that particular role. This done by using the precise relation type <is classified by role as a>. However, it is allowed to use the normal classification relation <is classified as a>, whereas the software will determine from the definition of the kind of thing that the kind of thing is in fact a kind of role of a kind of thing. For example, it will discover that there are not such things as front-end bearings, but only bearings located at the front end of a shaft.

For example, the requirement that a pump shall have a front-end bearing may be satisfied as follows:

P-1201	has as part	B1
B1	is classified by role as a	front-end bearing

Note that the individual role R1 of the bearing B1 is not made explicit in this model.

12.9.4 Step 4, Required kinds of aspects

The specification that a kind of physical object requires particular kinds of aspects means that for each object of that kind there shall be values given for each of the aspects. If the values of the aspects (properties) are quantitative (numeric), then the value shall be accompanied by a unit of measure (kind of scale). The specification of the requirement may also specify the required unit of measure.

Note that the kinds of physical objects, the kinds of aspects and the units of measure shall be selected from the project dictionary.

For example, for a vessel it is a requirement (in a particular context) that the length between its tangent lines shall be given in millimeters. This can be expressed in Formal English as:

vessel shall have as aspect a length between tangent lines mm

When we add the unique identifiers of the kinds of things from the formal dictionary and express the requirements in an Expression Table, then the core of a line with this example becomes as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of Relation type	Name of relation type	UID of right hand object	Name of right hand object	UID of UoM	UoM
520,243	vessel	219	4,956	shall have as aspect a	550,362	length between tangent lines	570,423	mm

How to satisfy a requirement for an aspect

The formal language specifies that a relation of the type <shall have as aspect a> is an expression of a requirement that can be satisfied by a relation of the type <has as aspect>. That specification can be found in the collection of facts about realization relations and is expressed as a relation between relation types as follows:

shall have as aspect a can be fulfilled by a has as aspect

In the formal dictionary this is expressed as:

UID of left hand object	Name of left hand object	UID of fact	UID of Relation type	Name of relation type	UID of right hand object	Name of right hand object
4,956	shall have as aspect a	139	5,165	can be fulfilled by a	1,727	has as aspect

With this knowledge it becomes possible that knowledge-aided design software assists in the design process as follows:

Assume that a vessel is designed, called V-1205, then from the above specification of a requirement for a vessel computer software can conclude that V-1205 shall have a length between tangent lines, expressed in millimeters. So the software can ask for that value.

Verify deliverables on requirements for aspects

This paragraph describes the process to verify the requirement that objects of a kind shall have a particular kind of aspect. The core of that process is the following. For every object that is classified by a kind of object for which a requirement is specified the computer should search for relations of a kind that satisfy the requirement. Figure 110 illustrates a combination of a requirement and a satisfaction of that requirement, integrated in one model.. The figure shows a relation between an expression of a requirement and the expression of a fact that satisfies the requirement, being the <can be fulfilled by a> relation between the <shall have as aspect a> relation and the <has as aspect> relation.

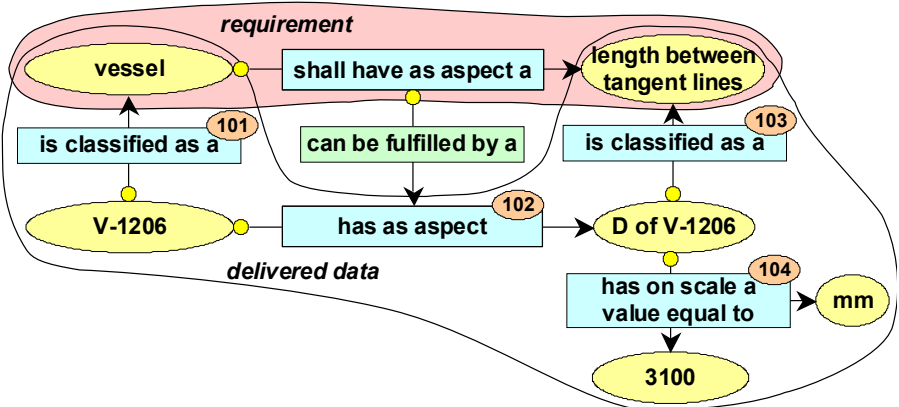


Figure 110, Relation between requirements and satisfying delivered data

The meaning of the requirement for a kind of aspect is not only that there shall be an aspect, but also that the aspect is qualified either in a qualitative way or in a quantitative way. This means that that there should be more than one fact delivered to satisfy one requirement. In this case an individual object shall have an aspect of the proper kind and that individual aspect shall be qualified by a qualitative aspect ('value'). That qualification may be either by a qualitative value or by a number on a scale ('dimensionless' aspects are quantified on the scale '-'). Examples of qualitative values are 'red' and

‘blue’ or ‘high’ and ‘low’, etc. The qualitative or quantitative aspect may be constrained by a list of allowed values. The allowed values should be defined and be part of the dictionary.

For example, assume that a piece of equipment, called V-1206, is delivered to a product owner who expressed the requirement that a vessel shall have a length between tangent lines, as illustrated in Figure 110 and that V-1206 is classified as a vessel. Then the database system of the receiving party can automatically ‘know’ from the above described specification of requirements that the delivered data is only complete when V-1205 has a length between tangent lines with a value. Furthermore, assume that the data about the equipment as illustrated in Figure 110 is delivered in table as follows (in which the numbers of the fact UIDs correspond with the numbers in the figure):

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object	UID of UoM	UoM
1	V-1206	101	1,225	is classified as a	520,243	vessel		
1	V-1206	102	1,727	has as aspect	2	length of V-1206		
2	length of V-1206	103	1,225	is classified as a	550,362	length between tangent lines		
2	length of V-1206	104	5,025	has on scale a value equal to	922,365	3100	570,423	mm

This enables that the software can verify the data automatically and can report that the requirement is satisfied. This enables to verify automatically the progress of the delivery of the data it also enable to determine an information quality Key Performance indicator for completeness of the data.

The requirement as well as the qualification can also be expressed by variants of the relation type, such as:

- 5,489 has on scale a value greater than or equal to
- 5,490 has on scale value less than
- 5,468 has on scale a value within numeric range
- 5,020 is qualified as

The specification of the requirement that the aspect values may only be selected from a list of standard qualitative aspects is discussed in paragraph 12.9.7.

Specify requirements for aspects of parts

Frequently requirements for aspects appear to be aspects of parts. For example, for a centrifugal pump it may be specified that it shall have as aspect an <impeller diameter>. An extended model should include two specifications: one that specifies that a centrifugal pump shall have as part an impeller and secondly that an impeller shall have as aspect a diameter. The direct relation between centrifugal pump and <impeller diameter> is in fact short-cut for the extended specification. When the same or another database also contains a separate part, classified as impeller, with a diameter, then it is usually difficult for a computer to determine that the <impeller diameter> of a pump is the same thing as the <diameter> of an impeller that is a part of that pump. This illustrates that the short-cut specification has a clear disadvantage.

Note: This is a main reason why many systems with data about the same kind of things have so many kinds of ‘properties’ (aspects) and why those properties are also different. They do not have only properties such diameter, but they also have impeller diameter, pipe diameter, etc. etc. Apart from the fact that those properties often have various different names, abbreviations and codes.

To enable computer application systems to link those different ways of specification, we need precise relation types in order to distinguish between a requirement for an aspect of an assembly and a short-cut relation for the specification of a requirement for a property of a part. In the above example, the formal specification becomes:

centrifugal pump shall have a part with as intrinsic aspect a impeller diameter

This is represented in a formal table as follows:

UID of left hand object	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
130,057	centrifugal pump	201	5,238	shall have a part with as intrinsic aspect a	850,373	impeller diameter

For a specification of required parts, see chapter 12.9.3.

The concept <impeller diameter> is an example of something that is not a pure aspect (or property). It is an aspect in a role of being possessed by definition by an impeller. In general all such aspects that have a kind of physical object in their name are by definition possessed by that particular kind of object. They shall be defined as a subtype of <intrinsic aspect>.

which is a role of an aspect. Those concepts and definitions shall be included in the formal dictionary.

12.9.5 Requirements for multiple relations of the same kind

Simultaneous cardinalities

Requirements may specify that an object of a particular kind shall have more than one part of a kind at the same time. (Due to replacements during its lifetime, an object nearly always may have parts of the same kind, one after the other.) This can be indicated by a specification of the minimum and maximum simultaneous cardinalities for the left hand object and for the right hand object in a formal expression. The minimum simultaneous cardinality specifies the minimum number of individual things of the kind that shall be related at the same time to one object of the other kind. The maximum simultaneous cardinality specifies the maximum number that may be related to an object of the other kind. For example, for a car it may be specified that one car shall have at least 4 and also at most 4 wheels. But one wheel may be part of none or of one car at the same time. This is then specified as follows:

car 0,1 shall have as part a 4,4 wheel

12.9.6 Requirements for units of measure

Each quantification of an aspect by a relation between a property and a number shall be accompanied by a scale (units of measure) that indicates the scale that need to be used to interpret the property value from the number.

When it is specified that an object of a kind can or shall have a property of a particular kind, then that main fact may be accompanied by an contextual fact that specifies a unit of measure that shall be used to quantify a property of that kind when being possessed by an object of the specified kind. In such a case the unit of measure shall be interpreted as the default unit of measure for the quantification relations of that kind. For example:

wheel (4956) shall have as aspect a diameter mm

The above expression specifies that diameters of wheels shall be quantified by numbers on a millimeter scale. This expression does not imply a specification for diameters of other kinds of things. If all diameters shall be quantified using the same scale, then that can be specified as follows:

diameter (5051) shall be quantified on scale mm

12.9.7 Step 5, Allowed values for required aspects

In many situations the values (numbers or literals) for kinds of aspects are limited to a discrete number of values in a list. Typically shown as pick lists of classes in a user interface. Such allowed values can be collected in a collection of qualitative aspects by stating that they are an element of the collection. This is specified for a collection of classes for example as follows:

Colors A	is a qualification of	coll. of qualitative aspects
red	is an element of collection of kinds	Colors A
green	is an element of collection of kinds	Colors A
blue	is an element of collection of kinds	Colors A

Usually the constraints do not apply to all individual things that are classified by the concept. For example, not all colors are constrained by red, green and blue. Because often constraints only apply to aspects of a kind that are possessed by objects of a particular kind. For example, it may be specified that the colors of a particular type of cable are limited to the colors in Collection A. This can be expressed as follows:

cable type A	shall have as possessed aspect a	cable type A color
cable type A color (4846)	shall be one of the	Colors A

12.9.8 Step 6, Textual requirements and rules

Rules and requirements are usually expressed as text in documents, such as in national, international, industry and company standards or in government regulations. The information in a requirements document can be extracted from the document in various forms:

- The separate requirements can be extracted from such documents as a collection of more or less independent and consistent rules.
- Each rule can be converted into an explicitly modeled requirement.

The explicitation of requirements in documents can vary in level of detail as is illustrated in Figure 111.

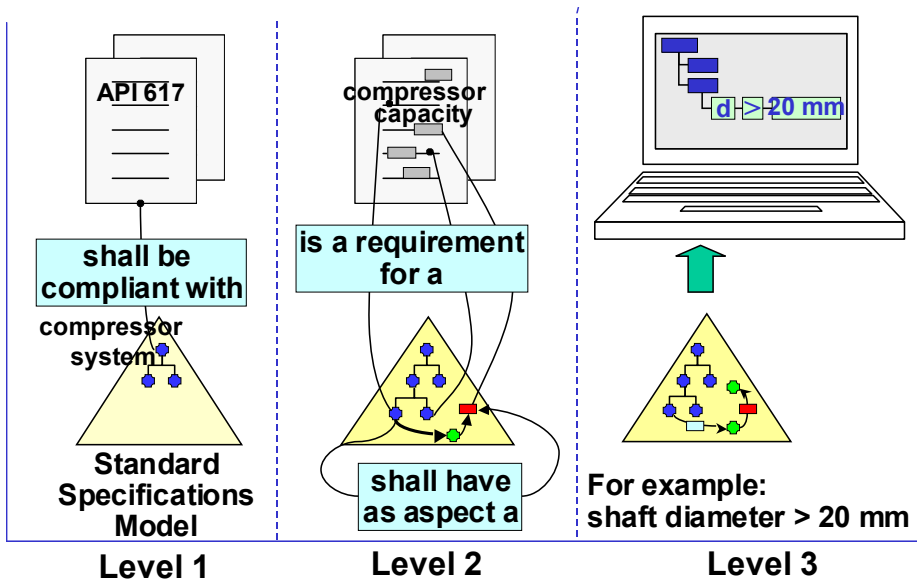


Figure 111, Levels of explicit modeling of document content

We distinguish three levels of explicit modeling:

1. Document level modeling. This implies that the documents as a whole are related to one or more kinds of things. This is described in the next chapter.
2. Partial explicit modeling. This implies that a document is split in 'pieces of text' that are related to a kind of thing, whereas that kind of thing is included in a model of the whole kind of thing about which the document provides requirements. Note that this might be done while maintaining a view on the requirements in the form of the whole document.
3. Full explicit modeling. This converts requirements from text to structured data as described in the other parts of this document.

An example of the first level is a requirement that "a centrifugal compressor shall be compliant with API 617". This requirement refers to a complete document, the API 617 standard, in which requirements are described. That document contains a number of requirements about centrifugal compressors, among others. However, the requirement does not express any 'explicitation' of the document and declares that the document as a whole is applicable.

An advantage of the second level of extraction of textual requirements is that each of the separate requirements, as expressed in sentences or paragraphs,

can be allocated to a particular kind of object or kind of activity in a knowledge and requirements model. This enables that all requirements from different documents about the same kind of component or activity can be combined and shown together. This makes it easier to verify the consistency of rules from different sources. It also enables that the design or verification of an item of a particular kind can apply all the different rules together. Another advantage is that it is much easier to verify whether all the requirements are satisfied or to record which fact satisfies which requirement.

Thus on the second level of explicitation the content of a document is decomposed in parts, such as paragraphs, each of which is a separate requirement. Such a requirement will typically consist of one or more sentences.

The methodology to model requirements and rules is illustrated by the example of a conditional requirement in the ISO 13709 standard with requirements about compressors. The example requirement is specified in the following sentence:

C.3.3.3 Rotor systems that have low inertia and are subject to accidental unloading should be equipped with a quick-acting brake to prevent damage from overspeed.

This whole requirement can be related to an explicit purpose and can be described and related to a rotor system in the way as described in the previous paragraphs. This will result in the following model:

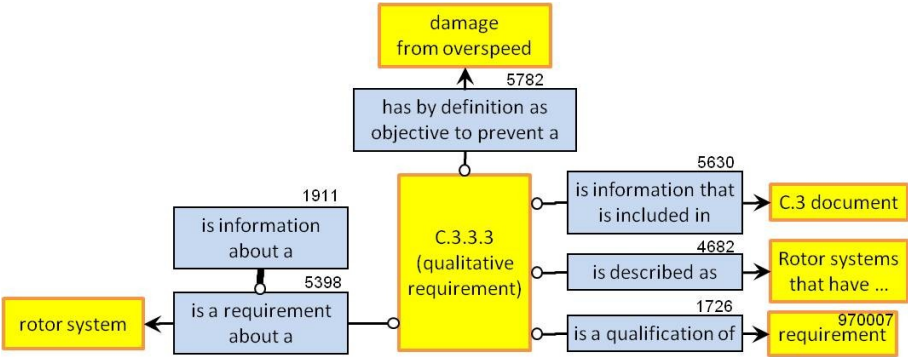


Figure 112, Modeling of a textual requirement at level 2: partially explicit

For example some sentences that appears in that API 617 standard specification document. These two examples of requirements can be expressed as follows:

UID of	Name of left	UID of	Name of	UID of	Name of right	Full description

left hand object	hand object	relation type	relation type	right hand object	hand object	
130,069	compressor	5,298	shall be compliant with	5	ISO 13709	
5	ISO 13709	1,726	is a qualification of	910,137	standard specification	
130,069	compressor	5,298	shall be compliant with	6	C.3.3.3	
6	C.3.3.3	1,726	is a qualification of	970,007	requirement	Rotor systems that have low inertia and are subject to accidental unloading should be equipped with a quick-acting brake to prevent damage from overspeed.
6	C.3.3.3	5630	is information that is included in	5	ISO 13709	

Note that the text that makes up the requirement is a separate object that has a name (in this example ‘C.3.3.3’), just as the information that makes up the full document. Both objects are defined as a qualification of information (or of a subtype of information, such as ‘requirement’). The full textual requirement is provided in the column ‘full description’ in the Gellish Expression Table on the line where the requirement is defined to be a requirement.

The above model includes the specification of objects that represent the sentences in a requirements document. In addition to that we could also model the structure of the requirements document itself, thus creating a ‘document model’. Such a model would specify for example the sequence of the sentences and where the chapters and paragraphs start and finish, etc.

The third level of detailed modeling of textual requirements is described in paragraph 12.9.10.

12.9.9 Relate documents to files and addresses

A full document text is usually not explicitly included as text in a requirements model, but is referenced via a relation to one or more external electronic data files on which the document is presented. For example, it can be stored in a pdf file as well as in a doc file. This is done as follows:

UID of left hand object	Name of left hand object	UID of Fact	UID of Relation type	Name of relation type	UID of right hand object	Name of right hand object
106	API 617.pdf	208	1,225	is classified as a	490,533	electronic data file
106	API 617.pdf	209	1,227	is an element of	107	http://
107	http://	210	1,225	is classified as a	970,274	URL address
108	API 617	211	4996	is presented on	106	API 617.pdf

12.9.10 Detailed modeling of textual requirements and rules

The full detailed modeling of textual requirements and rules enables automation of consistency verification and computer aided support for design, execution or verification.

A semantic analysis of the sentences the express requirements and rules result in a recognition of a number of “objects” in the sentences and a number of relations of various kinds. For the example requirement C.3.3.3 from ISO 13709 the objects are illustrated in the analysis schema of Figure 113.

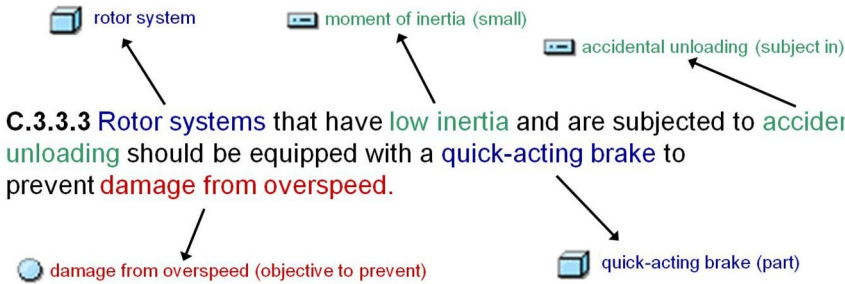


Figure 113, Textual requirement

This analysis shows that the requirement is built on knowledge about possibilities of aspects, parts and processes of rotor systems. Such knowledge can be modeled independent of the requirement. This results in an implied knowledge model, in this example about a rotor system. The example knowledge model is presented in Figure 114.

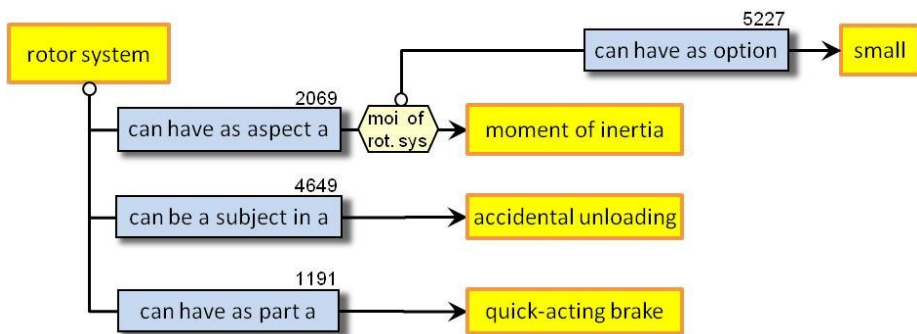


Figure 114, Implied knowledge model about a rotor system

The conditional requirement can now be modeled explicitly by a conceptual if-then-else relation that builds on the knowledge model. The requirement implies two conditions: that two possibilities in the knowledge model are the case for an individual rotor system. Thus, if an individual thing is classified as a rotor system or a subtype of it and if for that individual thing those conditions are satisfied, then the third possibility in the knowledge model shall be realized for that rotor system. Thus the rotor system shall have as part a component that is classified as a quick-acting brake.

Thus, the general structure of a conceptual if-then-else relation is:

Part of clause	Left hand object	Name of relation type	Right hand object
if	rule	has by definition as condition a	condition-1
and if	rule	has by definition as condition a	condition-2
or if	rule	has by definition as alternative condition a	condition-3
then	rule	shall have as consequence a	consequence-1
and then	rule	shall have as consequence a	consequence-2
else	rule	shall have as alternative consequence a	consequence-3

The conditions and consequences are shall be possible facts (states). The complete clause *is composed of* two or more partial clauses, which implies that if a kind of thing in any of those condition or consequence is mentioned more than once in one conditional requirements model, then it is a rule that in all cases the same instance is meant. In other words, if a condition regards something of a particular kind and another condition or one or more consequences regard something of the same kind, then the condition(s) and the consequence(s) regard the same thing of that kind.

Requirement Model of C.3.3.3								
2	101	1	3	74	75	15	201	65
UID of left hand object	Name of left hand object	UID of fact	Name of relation type	UID of role of right hand object	Name of role of right hand object	UID of right hand object	Name of right hand object	Partial definition
301	rotor system	201	can have as aspect a	302	moment of inertia of a rotor system	550,024	moment of inertia	acce
302	moment of inertia of a rotor system	204	can have as option			303	small	acce
301	rotor system	205	can be a subject in a			304	accidental unloading	acce
301	rotor system	206	can have as part a			305	quick-acting brake	acce
101	C.3.3.3	207	is a qualification of			970,007	requirement	Rotor systems that have low inertia and are subject to accidental unloading should be equipped with a quick-acting brake to prevent damage from overspeed.
101	C.3.3.3	208	is information that is included in			102	Document C	acce
101	C.3.3.3	209	is a requirement about a			301	rotor system	acce
101	C.3.3.3	210	has by definition as objective to prevent a			306	damage from overspeed	acce
101	C.3.3.3	211	is modelled as			103	C.3.3.3 model	acce
103	C.3.3.3 model	212	has by definition as condition a			204	Model of condition-1	acce
103	C.3.3.3 model	213	has by definition as condition a			205	Model of condition-2	acce
103	C.3.3.3 model	214	shall have as consequence a			206	Model of consequence-1	acce

Note that in the above table the UIDs of the relation types are not shown, nor the other contextual facts.

12.9.11 Step 7, Specify requirements about kinds of documents

There are several kinds of requirements that are related to documents. Those requirements specify:

- 1.General requirements that apply for documents of any document type.
- 2.Requirements that specify per delivery of pieces of hardware of a kind which kinds of documents shall be delivered simultaneously.
- 3.Requirements that specify the minimum kind of content that is required for kinds of documents.

Requirements for data about documents

These are requirements for data about documents of specified kinds. Sometimes such requirements are called requirements for meta data. However, documents are objects in their own right. Therefore the requirements about them can be included in information models just as all other objects. For example:

Name of left hand object	Name of relation type	Name of right hand object	Full description
document	shall have as aspect a	document title	
document title	shall be compliant with	Requirements for document titles	
document title	shall be compliant with	Requirement for old titles	
Requirement for old titles	is a qualification of a	requirement	The mapping to an old title shall be retained as a synonym with the status 'replaced'.
document	shall be classified by a subtype of	document	

Required types of documents and data sets per object type

Information that is delivered may be in the form of documents, but those documents may well be generated from the content of databases. This illustrates that the same information may be delivered in the form of data sets, as well as in the form of documents. The management of information therefore requires an integrated management of documents and data sets.

A data set is an expression of a collection of facts that are stored in a file or database according to an explicitly defined data structure. The generation of a document from such a file, for example by a report generator, in fact implies a conversion of the form of the data and an addition of a lay-out and possibly explanatory text to support the interpretation of the data.

For example, an Excel spreadsheet with a table that is an equipment list can be treated as a document, but its content is in fact a data set that can be loaded in a bigger database, to become part of an Information Model. However, a data set may also consist of a complete database, such as an MS-Access database with the same equipment list and possible other information in various tables.

The management of documents and data sets includes not only the management of the documents and data sets as a whole, but also the management of their content, especially because the content of data sets can be managed and verified on completeness, consistency and other quality criteria. This means that data should be managed by data set and that data sets are treated in the same way as documents¹⁶.

¹⁶ A data set is in fact an expression of information, not in a natural language, but in a data structure, which is basically an artificial formal language. This can be seen from

This implies that requirements can be expressed that for a kind of thing (facility, equipment, component or activity) not only particular kinds of documents shall be delivered, but also kinds of data sets.

Such kinds of requirements can be expressed in Formal English as follows:

UID of left hand object	Name of left hand object	UID of Fact	UID of relation type	Name of relation type	UID of right hand object	Name of right hand object
160,104	process unit	6,000,220	5,103	shall be described by means of a	490,186	process flow diagram
703,006	piece of equipment	6,000,221	5,103	shall be described by means of a	493,539	equipment list data set

Note that a specification that a piece of equipment shall be described by means of an equipment list (document) differs from a specification that it shall be described by means of an equipment list data set. Because the two are defined differently and there may be different requirements for their content and format. An equipment list is defined as a document for which an obligatory lay-out may be specified, for example in a company standard (such as in a Design and Engineering Practice standard), which may be delivered as an electronic file in a particular, such as in a DOC or PDF format or in an Excel format (XLS). An equipment list data set is an electronic file with a specified data structure, for example a Gellish Expression Table in the form of an MS-Access database or in the form of an Excel spreadsheet.

The types of documents, as well as the types of data sets shall be defined in a domain dictionary.

Minimum content per document type

The formal dictionary shall contain a definition of all used document types. A definition of a document type usually specifies what kind of information should be contained in a document in order to be classified by that type. However, this is only in very general terms and in many cases there are expectations of minimum requirements for such content, especially from a professional, good quality engineering perspective. However, it is valuable to make such requirements explicit, in order to have an explicit reference for document quality verification.

the fact that Formal English data sets are human readable as a structured subset of natural English.

12.9.12 Step 8: Requirements for kinds of occurrences

Occurrences include processes, activities (functions) and events.

Requirements models of physical objects may also include a specification of processes, activities and events in which the object shall be involved, for example as a performer or by being subject in an activity. For example, the fact that a vessel shall be subject in a pressure test can be expressed as follows:

vessel		shall be subject in a		pressure test

Concepts such as pressure test shall be defined in the formal dictionary. Specific subtypes such kinds of activities may for example include a definition of a particular test method.

12.10 Verification of product models against requirements

Product and process models either resulting from design or models that describe delivered and built products and processes may be verified using software that not only verifies whether the models consist of correct formal language expressions but also verify whether the expressions comply with modeled knowledge and requirements of things of the modeled kinds.

Requirements in the form of a requirements models are meant for usage for computer supported verification of designs of individual things or for verification of delivered and inspected things. Such verifications are possible when product models of individual things are expressed as a collection of formal expressions, while knowledge and requirements are expressed by using the same formal language. Then each expression in a requirements model for a kind of thing should have its counterparts in the product models about things of that kind. This uses the above-mentioned relation between a conceptual relation and a relation between individual things in the inverse direction. For example, if a product model of the individual object P-101 that is classified as a centrifugal pump is specified to have no impellers, then computer software should be able to verify whether that is possible according to the knowledge model and whether it is allowed according to a requirements model about a centrifugal pump.

Statements only comply with specified knowledge about possibilities when the related individual things are classified by the concepts that are mentioned in the knowledge models or by their subtypes.

Figure 116 illustrates an example of the verification of a model about P-101.

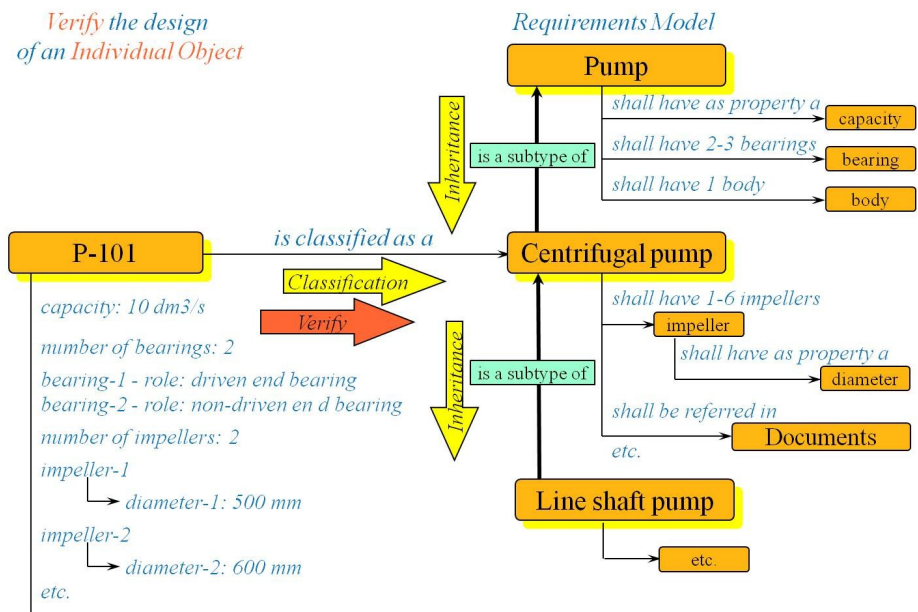


Figure 116, Example of verification of facts against requirements

The left hand side of Figure 116 illustrates statements about P-101, including also the statement that P-101 is classified as a centrifugal pump. The right hand side of the figure illustrates requirements that are expressed about a centrifugal pump. Those requirements include requirements that are inherited from its supertype ‘pump’. The verification includes a check whether each requirement is reflected in a statement about the individual P-101 and whether the constraints are satisfied.

The verification process uses the same relations between relation types that express requirements about kinds of things and relation types that express facts about individual things as are used for the creation of proposals and that are documented in the extended upper ontology section of the formal dictionary (see the previous chapter).

The application software should generate a report about inconsistencies and omissions.

12.11 Summary of relation types for modeling of requirements

The following relation types are used to create a core of requirements models as described in this document. Additional relation types will be needed to add other kinds of requirements in the form of data or when the content of documents is modeled.

Note that each relation type has an inverse expression. When the inverse expression is used the left hand object and right hand object shall be exchanged.

Relation type UID	Relation type name	Example	Description
1.191	shall have as part a	road <shall have as part a> lane	is a relation that specifies that a physical object of a specified kind shall have as part a physical object of another kind, within cardinality constraints. The requirement is only valid in a specified validity context. Note that the cardinality constraint may indicate that such a part is still optional.
5.282	shall have as aspect a	motorway <shall have as aspect a> width	is a relation that relates a kind of physical object or a kind of occurrence to a kind of aspect, that specifies that a physical object or an aspect of the kind shall possess an aspect of which the qualification shall be specified. Typically by a quantification on a scale.
5.282	shall have as possessed aspect a	motorway <shall have as possessed aspect a> road width	is a relation that relates a kind of physical object or a kind of occurrence to a kind of possessed aspect, that specifies that a physical object or an occurrence of the kind shall possess such an aspect of which the qualification shall be specified. Typically by a quantification on a scale.
5.238	shall have a part with as possessed aspect a	car <shall have a part with as possessed aspect a> wheel diameter	is a relation that specifies that each physical object of the specified kind shall have a part that shall have an aspect that shall have a role of the specified kind.
1.146	is a specialization of	motorway <is a specialization of> road	is a relation between two concepts that specifies that the first concept is a subtype of the related second concept (the supertype).
1.726	is a qualification of	1: red <is a qualification of> color 2: requirement-1 <is a qualification of> information	is a relation between a qualitative aspect and a conceptual aspect that specifies that the qualitative aspect has the conceptual aspect as its nature. Note that a quantitative aspect is a subtype of qualitative aspect.
4.714	can have a role as a	room <can have a role as a> meeting room	is a relation that specifies that an individual thing of the specified kind can play a role of the specified kind. Irrespective of whether it actually plays such a role.

5.398	shall be compliant with	boiler <shall be compliant with> specification-1	is a relation that specifies that any object of the specified kind shall be compliant with the specified qualitative requirement. Typically an item, story or document of a kind that shall be conform to specific information that is the contained in a document.
4.846	shall be one of the	car color <shall be one of the> list of allowed car colors	is a relation between two concepts that specifies that a member of the first concept has by definition one or more parts that are member of the second concept.
4.730	is an element in collection of classes	red <is an element in collection of classes> list of allowed car colors	is a collection relation that specifies that the collected class is collected in the collector collection of classes.
5.737	is by definition quantified on scale as equal to	6 mm <is by definition quantified on scale as equal to> 6 (on scale mm)	is a relation that specifies that a qualitative property has by definition a magnitude that is equal to a value on a specified scale.
5.051	shall be quantified on scale	width <shall be quantified on scale> mm	is a relation that specifies that properties of a specified kind shall be quantified by quantitative values (typically numbers) on a specified qualitative scale.
5.658	are requirements for a	requirements for bus stop type-1 <are requirements for a> bus stop type-1	is a relation that indicates that individual things of the specified kind shall be compliant to (all) the requirements in the collection. Typically a kind of item or document which members shall be compliant to the collection of requirements that is a collection of required facts, typically expressed as a collection of shall be... or shall have... relations.
5.398	is a requirement for a	req-1 <is a requirement for a> fire resisting door	is a relation that specifies that a qualitative requirement (information) is applicable for things of a specified kind. The requirement has a name (e.g. R-21) and can be made textually explicit by a description. A requirement can also be modeled, for example by an if-then-else relation.
5.227	is modeled as	req-1 <is modeled as> req-1 if-then-else model	is a relation that indicates that a modeled object is modeled by a relation.
4.996	is presented on	req-1 <is presented on> req-1.pdf	is a relation that indicates that a piece of qualitative information is expressed on an individual physical object. It is expressed as an aspect of the physical object. For example as a spatial (shape and pattern) aspect of ink on paper.
5.630	is information that is included in	req-1 <is information that is included in> document A	is a relation that indicates that qualitative information is included in other qualitative information. Typically a piece of text that is included in a chapter or in a document.
5.778	has by definition as objective a	req-1 <has by definition as objective> simplification of maintenance	is a relation that indicates that individual objects of the specified kind have by definition a purpose of existence of the specified kind.
5.782	has by definition as objective to prevent a	req-1 <has by definition as objective to prevent> corrosion	is a relation that indicates that individual objects of the kind are by definition preventing to achieve a state of the specified kind as their purpose of existence. Typically the purpose or objective is a kind of occurrence.

4.972	has by definition as condition a	if-then-else model-1 <has by definition as condition a> conceptual composition relation (can have as part a)	is a relation that indicates that a possible fact of a specified kind is by definition involved as condition in a conditional consequence relation. This is the 'if' condition in a qualitative if-then-else clause. Several of this kind of conditions for the same relation imply that the consequence is only applicable when all those conditions are satisfied. Thus when condition-1 and condition-2 etc. is satisfied.
5.783	has by definition as alternative condition a	if-then-else model-1 <has by definition as alternative condition a> conceptual composition relation (can have as part a)	is a relation that indicates that a possible fact of a specified kind is by definition involved as alternative condition when another condition might not be satisfied in a conditional consequence relation. This is the 'or if' condition in a qualitative if-then-else clause. Several of this kind of conditions for the same relation imply that the consequence is applicable when any of those conditions is satisfied. Thus when condition-1 or condition-2 or condition-3 etc. is satisfied.
4.976	shall have as consequence a	if-then-else model-1 <shall have as consequence a> conceptual composition relation (can have as part a)	is a relation that indicates that a state of a kind is achieved or shall be achieved if or when a condition of a kind is fulfilled, otherwise an alternative state of a kind may be achieved. This is a consequence when the condition or sufficient conditions are satisfied (then) in a qualitative is-then-else relation type. For example the consequence that an occurrence of a kind takes place or shall take place when a property of a kind becomes equal or greater than a specified value.
4.979	shall have as alternative consequence a	if-then-else model-1 <shall have as alternative consequence a> conceptual composition relation (can have as part a)	is a relation that indicates that a state of a kind is achieved or shall be achieved if or when a condition of a kind is not fulfilled. This is an alternative consequence when the condition or sufficient conditions are not satisfied (else) in a qualitative is-then-else relation type. For example the consequence that an occurrence of a kind takes place or shall take place when a property of a kind is not equal or greater than a specified value.
4.994	shall be classified by a subtype of	pump <shall be classified by a subtype of> pump	is a relation that indicates that individual things of a specified kind shall be classified by subtypes of the conceptual classifier kind. The requirement is only valid in a specified validity context. For example, it might be specified in a particular validity context that every pump shall be classified by a subtype of pump.
5.103	shall be described by means of a	process unit <shall be described in a> specification	is a relation that indicates that things of the specified kind shall be described in a piece of information (such as a document) of a specified kind. This implies also a required kind of physical document or physical file about a kind of thing.
5.165	can be fulfilled by a	shall have as part a <can be fulfilled by a> has as part	is a relation that indicates that a requirement that is expressed by a relation of a specified kind can be fulfilled by a relation of a kind that is a fulfiller.

Relati on type id	Relation type name	Required first role	Allowed first role player	Required second role	Allowed second role player
4956	shall have as aspect a	possessor	physical object	possessed	aspect
5282	shall have as possessed aspect a	possessor	physical object	possessed aspect	role
5238	shall have a part with as possessed aspect a	conceptual possessor via part	physical object	conceptually possessed role	possessed aspect
1146	is a specialization of	subtype	concept	supertype	concept
1726	is a qualification of	qualifier	qualitative aspect	nature	concept
4714	can have a role as a	conceptual player of a role	physical object	conceptually played role	role
4751	shall be compliant with	conceptually specified	physical object	qualitative compliance criterion	information
4846	shall be one of the	common options possessor	aspect	common options	collection of classes
4730	is an element in a collection of classes	collected class	concept	collecting plural class	collection of classes
5279	can be quantified on scale as	commonly quantified characteristic	characteristic	common quantifier of a characteristic	mathematical space

13 Modeling of product types

13.1 Product libraries and electronic product catalogues

This section contains guidelines for the modeling of product types as are for example available in suppliers product catalogues and in standards. Electronic catalogues are kinds of ‘object libraries’ or ‘product libraries’ that are suitable for application in design systems or procurement systems in E-commerce, possibly searchable via the Internet.

This section is especially aimed at product catalogue developers and engineers who are involved in creating electronic catalogues and at software developers who develop software tools or extensions of various kinds of applications that import and/or export catalogue item data. For example, for developing procurement databases that enable selection of products and retrieval of their data in E-commerce or enable loading of product catalogues in design systems.

Supplier catalogues written a formal language enable simplifying the selection and comparison of products. When buyers express requirements in the same formal language, then computers can support selection and comparison of products.

A *product type* is a type of thing or a manufacturer’s model that has a more detailed specification than a more general concept of which it is a subtype. Product types are defined by qualitative aspects (aspect values or constraints) and by relations with other concepts. Often many product types are defined as subtypes of a common supertype concept, each with different values for particular aspects. For example, product types of the same shape, but with varying sizes or colors. To facilitate the specification of such product types, templates may be defined for supertype concepts that facilitate filling in aspect values and relations that need to be specified for various subtype product types of the same supertype. Sometimes the definition of a product type also includes relations with individual things, for example, a relation with the individual company that manufactures products of such a type.

The qualitative aspects (values) for the aspect may still allow for various options within a range or list of allowed values. For example, a product type can still leave the option open whether the delivered items should be red or blue.

Specifications of product types can be made for different purposes. Each purpose may require particular guidelines. Therefore the following scenario's are distinguished:

- Product types that exclude the specification of packaging.
- Product types that include the specification of delivery of collections in various packages.
- Buyer specifications. A buyer's specification has a role as requirements in an inquiry or in an order.
- Supplier specifications. A supplier's specification has a role as offering, commitment or guarantee in an offer. It has also a role as a requirement towards this internal production process within the supplier's organization.

The specification of product types in the form of an Expression Table implies that the product type specifications can be read by or imported in application systems. Such application systems may use the product type specifications for comparison or for selection, but may also use the product types as typical design solutions and as knowledge base for the design of individual objects. This document does not describe how application systems may use the product type models.

13.2 Product type information models

The specification of a product type is modeled in a formal language as a product type information model. Such a model consists of a collection of expressions of ideas with values or ranges for a number of aspects for the type of product, typically on a scale (with a unit of measure).

Product types should be distinguished from individual products. Individual products are for example those that are actually delivered and can be measured and may or may not conform to the specification for the types. But product types are subtypes of more general supertype kinds of products and they inherit the aspects that are defined for those supertypes. For example, the buyers catalog 'Materials and Equipment Standard Catalog' (MESC) item 76.65.79.766.1 and MESC item 76.65.79.766.2 are both product types that are a subtype of MESC item *supertype* 76.65.79.766 for which a product family requirements specification is defined in the MESC system (as is further discussed below).

The collection of expressions of facts that form the specification of a product type is thus called a product type information model.

The information about a product type generally consists of two categories of product type information models:

1. *Product family requirements models.*

A product family requirements model is a specification that is intended to have a role as template for the creation of specific subtypes of a kind of product. Each time such a template is filled-in a new product type is defined and its product type definition model is created. A kind of product for which a template is defined is a supertype of the product types that are created by filling-in the template through the specification of explicit values for the aspects that are included in the template.

2. *Product type definition models.* A specific product type that has by definition particular aspect values (qualitative or quantitative aspects) or components is defined as a subtype of a supertype kind of product, whereas the supertype does not by definition have those aspects values of components. If a template product family model is available for such a supertype kind of product, then usually the specific product type definition models are created through filling-in values in the template.

Note, that a product family requirements model expresses requirements and may express constraints for the qualification of aspects of the product types that are defined by filling-in the template.

13.3 Categories of product types

We distinguish three main categories of kinds of products:

- *Generic kinds of products*, also called kinds of physical objects. They may be single component products or assembled products that are composed of other products.
For example: nut, bolt, pipe flange, valve, pipe, pump, car, airplane, etc.
- *Product types (types of products)*, which are subtypes of the above kinds of which a number of aspects are defined to have particular values or that are defined to have components of particular kinds. Such types are typically defined by some organization as standardized subtypes of kinds of products.
For example: ISO standard bolts, Bridge type A, etc.

- *Manufacturer's models*, which are (further) subtypes that are defined by manufacturers as their proprietary models of various sizes, according to their internal specifications and tolerances of aspects and components.

For example: Manufacturer X flange, Volvo S40

Note: the term '*Manufacturer's model*' denotes a kind of product and not a kind of information model.

For each of kind of product in each of these categories there will be a definition model that defines the kind of product. It may be that even more than one information model is created for a particular kind. For example, there may be one *definition model* for a kind of product and for that same kind of product there may be various *requirements models* defined by different parties (there can also be several *knowledge models* for that kind of product, but knowledge models are not relevant in the context of this document).

Product type *definition models* are information models that specify (define) the aspect values and components that a products type or a manufacturer's model by definition has. A definition model has the form of a collection of Formal English expressions that mainly use relation types of the form <has by definition...>.

13.3.1 Product families

A 'product family' (a better name would be a 'product type family') is a collection of product types that have specific choices of values for characteristics in common while excluding other possible choices for those values. Standard product types and catalogues of manufacturer's models often contain a number of similar components that only differ in the values of a collection of aspects, such as sizes or materials of construction. Such a collection of similar product types is called a '*product family*'. Instead of specifying each family member independent of the other family members it is a good convention to specify first the common aspects of all family members as a requirements model or template for the creation of the definition models of the various members.

A *requirements model* for a family of subtypes of a kind of product specifies the compliancy requirements for the subtypes. Figure 117 illustrates that a requirements model <is a requirement for (the subtypes of) a> kind of product, e.g. for subtypes of 'welding neck pipe flange'. Thus it specifies the requirements for the definition of each defined product type that is a subtype of the kind of product (welding neck pipe flange) and thus is a member of the

family. Such a requirements model should specify the kinds of aspects and components for which values or composition relations are to be specified for each defined subtype product type or manufacturer’s model.

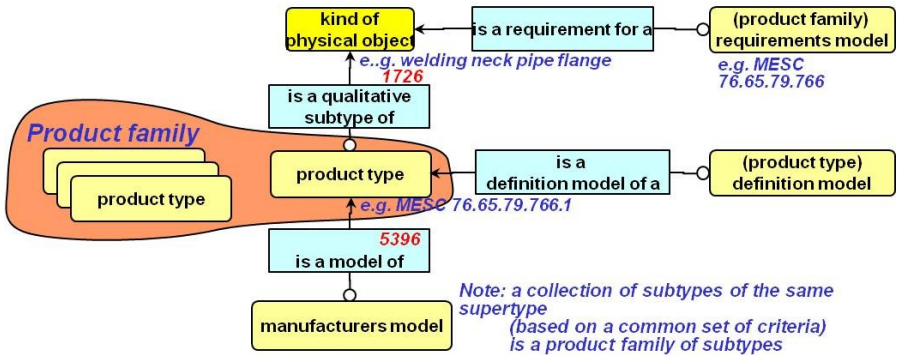


Figure 117, Specialization hierarchy, requirements models and definition models

Note that a manufacturer’s model may or may not be a subtype of a particular product type, although it still can be a subtype of a particular kind of physical object, i.e. a welding neck pipe flange. For example, a manufacturer may specify not only that his manufacturer’s model

model X is a model of welding neck pipe flange

but also that his ‘model X’ satisfies the definition of a product type MESC 76.65.79.766.1 in the requirements specification of his customer, by specifying that

model X is a model of MESC 76.65.79.766.1

With this specification he declares that the specifications of manufacturer’s model X are within the constraints of (or ‘is a subtype of’) that product type’s specification.

Often large organizations define templates of requirements and fill-in or ask manufacturers to fill-in those templates in order to define various subtypes in a family of products. Those templates are not product types, because values for various aspect are missing. Therefore we distinguish between:

- *Product family requirements models*, which can be used as templates for the creation of product type definition models. These requirements models fix some aspect values and possibly kinds of component for a kind of product, but leave many degrees of freedom

for the values of other aspects and components of products of such a kind.

- *Product type definition models*, which are specifications of the definition of particular (standardized) product types or manufacturer's models. Models of these kind define subtypes of kinds of products by fixing aspect values and component types, although there may still be degrees of freedom for a number of other aspect values and components.

13.4 Procedure for creating object libraries & product catalogues

An electronic object library or product catalogue is a collection of definition models of product types, also called product type models. The process to create such object libraries consists of the following steps:

- Step 1: Streamline the object library / product catalogue per product family for a kind of product
- Step 2: Determine characterizing aspects and component
- Step 3: Extend the taxonomic dictionary with components, intrinsic aspects and allowed values
- Step 4: Create a product family requirements model
- Step 5: Define a standard product type as a subtype of the kind of product
- Step 6: Specify values for the definition of the standard product type
- Step 7: Create a definition model for the standard product type, using the family requirements model as a template

13.4.1 Step 1: Streamlining object libraries & product catalogues

An object library or product catalogue usually consists of a collection of product type models that are grouped into product families. A product family is a collection of subtypes of a rather detailed kind of product. To some extent the choice of the kind of product for which a family is defined is a bit arbitrary, but is usually based on historic business practices determined by search experiences in design and procurement.

It should be ensured that the kind of product for which the subtypes are to be defined is itself defined and included in the formal language taxonomic dictionary as a consequence of the following guideline:

Guideline 1: *A product type shall be defined as an explicit subtype of a kind of physical object that is selected from the formal language taxonomic dictionary or from a properly defined subtype of such a concept.*

This guideline holds for the creation of any kind of physical object, including also any product type, catalogue item or manufacturer’s model. In this case for kinds of products for which subtypes will be created by using a product family requirements model as a template.

For example, assume that we want to create a product catalogue that should contain catalogue items that are subtypes of ‘welding neck pipe flange’. Figure 118 illustrates how a typical welding neck pipe flange looks like.

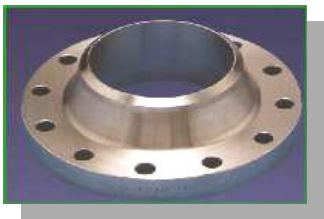


Figure 118, A typical welding neck pipe flange

The various catalogue items will differ in diameter, thickness, material of construction, etc. The collection of subtypes of ‘welding neck pipe flange’ in the product catalogue forms a product family.

Assume further that the concept welding neck pipe flange does not exist yet in the taxonomic dictionary of the formal language, whereas pipe flange does appear therein. Finally, assume that we define that welding neck pipe flange is a subtype of pipe flange. Then Guideline 1 is expressed by the following part of a line in an Gellish Expression format that provides the definition of a welding neck pipe flange:

UID -1	Left hand object name	Idea UID	Relation type UID	Relation type name	UID -2	Right hand object name	Description
1	welding neck pipe flange	100	1146	is a kind of	2	pipe flange	with a neck that is suitable for being welded.

Before we create the various standard product types as subtypes of a kind of product it is best practice to develop a product family requirements model. For example when we want to define product types that are subtypes of ‘welding neck pipe flange’, then a product family requirements model for a ‘welding neck pipe flange’ kind of product should be developed. Such a requirements model is intended to be used as a template for the creation of the definition models of the various catalogue items that are subtypes of the kind of product.

13.4.2 Step 2: Determine characterizing aspects and components

Typically standard product types are described in various ways. Especially in the specification of aspects there is usually no standard convention used and in many cases no definitions of the kinds of aspects are available. Thus agreement should be developed on the lists of aspects by which each product family is unambiguously defined. This shall include also the allowed values for each of the aspects.

13.4.3 Step 3: Define kinds of components, aspects and allowed values

If a concept does not exist yet in the taxonomic dictionary, then it shall be extended (as a private or as a formal extension). This holds especially for components, intrinsic aspects and qualitative aspects that are allowed as values for the aspects.

In the above example it might be that the concept ‘nominal diameter of a valve’ is not included in the taxonomic dictionary yet. The definition of that concept can be added as follows:

UID of left hand object	Name of left hand object	UID of idea	UID of kind of relation	Name of relation type	UID of right hand object	Name of right hand object
1,745	nominal diameter of a valve	103	1146	is a kind of	4,289	intrinsic aspect
1,745	nominal diameter of a valve	104	4714	can be a role of a	551563	nominal diameter
1,745	nominal diameter of a valve	105	5317	is by definition possessed by a	820153	valve

Note 1: Idea 104 is applicable for nominal diameters of all subtypes of valve. For example, object ...16428 inherits that idea because of idea 102.

13.4.4 Step 4: Create product family requirements models

A *product family requirements model* is a requirements model that is intended to have a role as a *template* for the specification of subtypes of a kind of product (the family master product type). Such a requirements model is a collection of expressions that expresses requirements for collections of facts that shall be created, whereas each collection defines a subtype of the family master product type. Such requirements are specific for a particular organization who expresses the requirements. Thus the requirements have a particular ‘validity context’, which means that they are not required when that context is not applicable.

Each created collection of filled-in requirements defines a subtype of the family master product type. A collection of subtypes may form a collection of catalogue items, object library or product library.

For example, we may define a *product family requirements model* for a ‘welding neck pipe flange’. That requirements model specifies for which kinds of aspects it is required to specify values in order to define a particular subtype of welding neck pipe flange. This illustrates that the requirements model is used for the definition of various subtypes of welding neck pipe flange. In other words it is a template for the creation of definition models.

Such requirements models are usually applicable only within a particular context. For example, the requirements for welding neck pipe flanges in the Shell’s MESC catalogue are MESC specific requirements and are only valid where MESC is applicable.

Similarly, a manufacturer may specify requirements for the aspects that shall be qualified to define product types (manufacturer's models) in the catalogue of product types that he produces and sells.

In general, a product definition model is an expression of aspect values and components that form requirements or constraints for the individual products that shall or will be delivered. In other words, they form the collections of requirements for the individual things that shall or can be delivered conform the definition of the product type.

A product family requirements model expresses which values for particular kinds of aspects and relations of particular kinds shall be provided in order to specify a subtype of the product type for which the requirements are applicable.

In addition to the specification of the aspects for which values shall be provided, a product family requirements model may also contain constraints for the values that are allowed for its subtypes.

A product family requirements model can be presented in a 'user view' template form with empty fields. This enable a user to enter values for the various aspects, within possibly defined constraints. Application software may use the entered data to generate different UIDs for the facts, objects and values and create a collection of expressions of facts that makes the definition model of the applicable subtype product type.

For example, assume that in a catalogue with a product family of 'welding neck pipe flanges' the requirements model specifies that for each subtype it is required to define:

- a DN nominal diameter,
- a material of construction,
- an ASME pressure class,
- a schedule of hub,
- etc.

An example of a user view (template) on such a product family requirements model for a welding neck pipe flange is given in Figure 119.

WELDING NECK PIPE FLANGE		
ATTRIBUTES	VALUES	Allowed VALUES
design specification dimensional specification of pipe material of construction material specification flange facing flange facing finish pressure class mandatory additional requirements inspection certificate schedule number of hub nominal diameter		ASME B16.5 specification ASME B36.19 specification UNS 32100, UNS 34700 ASTM A182 GRADE F 321, ASTM A182 GRADE F 347 raised face, flat face, ... smooth, ... pressure class 300 psig, ... MESC SPE 76/001 1999 MESC SPE 76/002 2003 ISO 10474-3.1B 10S, ... 25 mm, 30 mm, ...

Figure 119, User view (template) on a product family requirements model (including allowed values)

A product family requirements model defines *requirements for the definition models of product types that are subtypes* of a kind of product and thus it defines the constraints for any of those subtypes. For example, the above requirements model defines the constraints for any ‘MESC subtype’ of a welding neck pipe flange according to MESC item product family requirements specification 76.65.79.766. In other words MESC item 76.65.79.766 is the MESC code that is a synonym of the general concept ‘welding neck pipe flange’ (in the context of MESC), whereas the requirements model contains MESC specific constraints for the definition of subtypes of ‘welding neck pipe flange’.

When a product family requirements model is created, then each concept in the requirements specification shall be selected from the formal language taxonomic dictionary or shall be added as subtype of the concepts that already exist in the taxonomic dictionary. For example, in Figure 119 all concepts, such as the kind of product (‘welding neck pipe flange’), the aspects (under the heading ‘attributes’) and the qualitative aspects (under the heading ‘allowed values’) shall be selected from the taxonomic dictionary or its extension.

13.4.5 Step 5: Define product types

Once a product family requirements model with or without allowed values is created, the model can be used to create specific *definition models for each product type* that is a subtypes of the kind of product for which the requirements model was created. The creation of a standard product type

includes its definition as a subtype of the kind of product for which the family model is developed.

For example: in the MESC catalogue there is a MESC item 76.65.79.766.1 defined as a product type that is a subtype of welding neck pipe flange. This fact means that Guideline 1 requires that the MESC item is related to its direct supertype as follows:

UID of LH object	Name of left hand object	UID of idea	UID of relation type	Name of relation type	UID of RH object	Name of right hand object	Full definition
103	MESC item 76.65.79.766.1	101	1146	is a kind of	102	welding neck pipe flange	According to product definition model-1

Note 1: The definition of a subtype, such as ‘welding neck pipe flange’ implies that all the aspects of the supertype (such as ‘pipe flange’) are inherited by the subtype. Thus the above Mesc item inherists characteristics of a welding neck pipe flange.

Note 2: The UIDs of the (new) left hand object and the UID of the idea shall not within the range that is reserved for the formal dictionary (0 – 10.000.000). Non-overlapping UIDs can be created by using a proprietary prefix, other than Gel:.

Note 3: Relation type ID, relation type name, UID-2 and right hand object name shall be selected from the formal dictionary.

Note 4: If the left hand object will be proposed for inclusion in the formal dictionary, then the left hand object name shall be compliant with the naming conventions for objects in the formal language.

Note 5: The textual description of a subtype is actually superfluous if that expression is modeled by defining the value of the distinguishing aspect of the subtype. For example, by defining that a pipe flange has as aspect a connector type and that a welding neck pipe flange has a connector type that is a welding neck. For that reason the textual description of the MESC item is left blank.

13.4.6 Step 6: Specify values for the definition of product types

The ‘user view’ of product type definition model consists of a list of values for the degrees of freedom that are specified in a product family requirements model.

An example of such a subtype (the MESC catalogue item 76.65.79.766.1) is given in Figure 120.

MESC item 76.65.79.766.1	
ATTRIBUTES	VALUES
design specification	ASME B16.5 specification
dimensional specification of pipe	ASME B36.19 specification
material of construction	UNS 32100, UNS 34700
material specification	ASTM A182 GRADE F 321
flange facing	raised face
flange facing finish	smooth
pressure class	pressure class 300 psig
mandatory additional requirements	MESC SPE 76/001 1999 MESC SPE 76/002 2003
inspection certificate	ISO 10474-3.1B
schedule number of hub	schedule 10S
nominal diameter	25 mm

Figure 120, User view on a specific product type definition model

Note that the values that define a specific product type can still have some degrees of freedom by allowing for a range or for more than one option. This is illustrated in Figure 120 for the material of construction, where there are two allowed values for one aspect. In that case individual products or subtypes comply with the definition when the qualitative aspect is one of the values that is defined in the collection of values. With the mandatory additional requirements there are two requirements for the same aspect.

13.4.7 Step 7: Create definition models for product types

Guideline 2: **The information about a product type (often referred to as ‘attributes’ shall be distinguished in:**

- aspects that are possessed by such a product .
- aspects that are possessed by a part of such a product.
- information to which such a product shall comply.
- other things that shall be related to such a product by a particular kind of relation.

The general structure of a model of an aspect that can be, shall be or is by definition possessed by a physical object, the possessor role of the physical object and the ‘possessed aspect’ role of the aspect is presented in Figure 121.

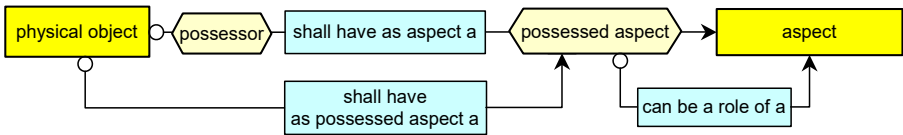


Figure 121, General structure of a model of an aspect that is possessed by a physical object.

This structure is used to derive the product family requirements models (that is used as a template for the creation of product types or catalogue items). The generic template for such a requirements model is illustrated in Figure 122.

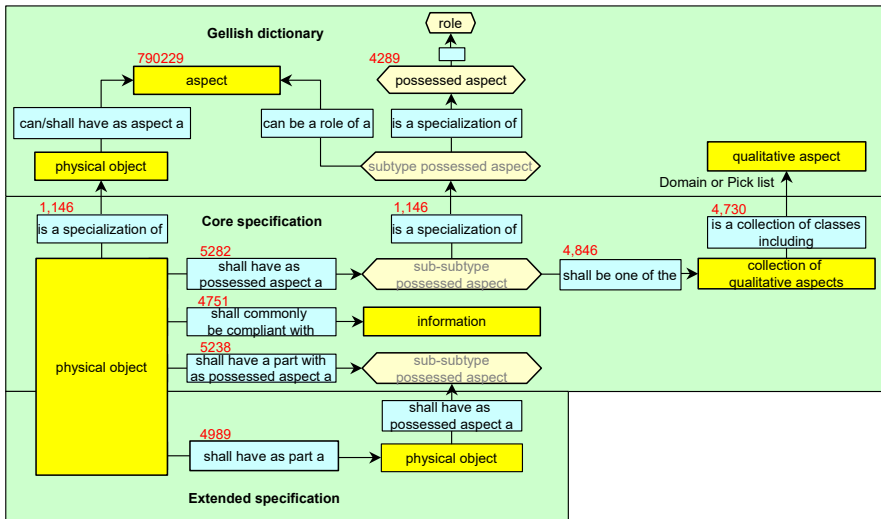


Figure 122, Product family requirements model (generic template)

The generic requirements model consists of a core specification, in which new concepts occur that shall be defined as subtypes of concepts that are already available in the formal language taxonomic dictionary . This is applicable for all concepts, but is illustrated only for two concepts!

For aspects that are actually aspects of parts or physical features of the physical object it is recommended to model not only the indirect relation of the aspect with the product type (the physical object), but also to model that a physical object is a part or feature of the product type and to define that the aspect shall be an aspect of the physical object that acts as part or feature.

This enables to map systems that do not support a composition structure to systems that support such a composition structure.

Note 1: The above generic model defines not only requirements such as:

product type shall have as aspect a aspect

However, this enables also to specify constraints for those aspects.

Those constraints are however not generally applicable for the kind of aspect as such, but only for the aspect as possessed by this product type. In other words: for the aspect in this role only. Therefore, the template relates a product type to an 'intrinsic aspect' (= an aspect in a particular kind of role) as follows:

product type shall have intrinsic aspect a intrinsic aspect

Note 2: A supertype has a distinguishing aspect of which each subtype by definition has a qualitative aspect (value). This can be expressed in text as a textual definition or it can be modeled by defining qualitative aspects of the subtype.

The above *generic* requirements model can be used as a *generic template* that is applied for kinds of products and several kinds of aspects to create product family requirements models. An example of such a product family requirements model (a filled-in generic template) for a ‘lined flanged ball valve’ is given in Figure 123.

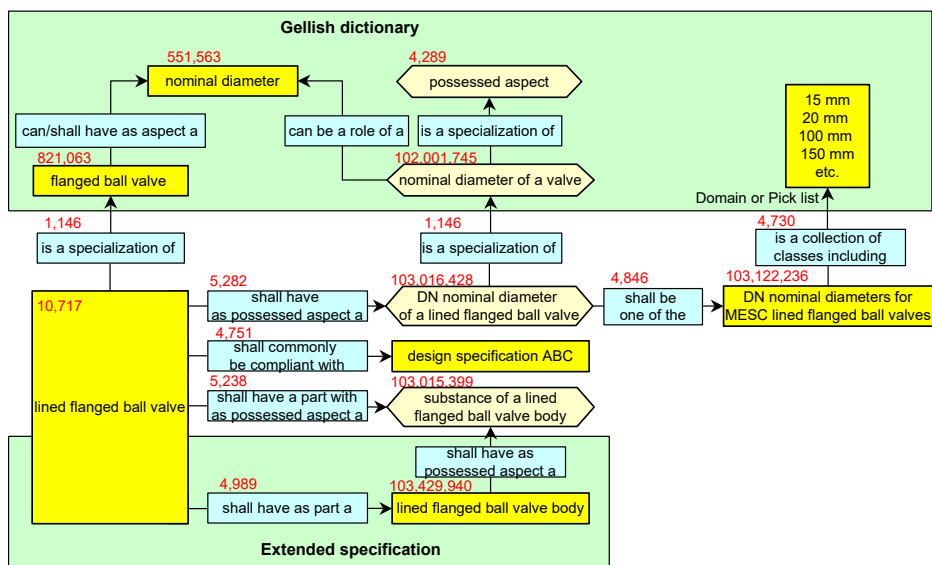


Figure 123, Example of a specification of product family requirements model

The example specifies among others, that it is a requirement (in the applicable validity context) that a lined flanged ball valve shall have a DN nominal diameter. This implies that a subtype of intrinsic aspect shall be created, called ‘nominal diameter of lined flanged ball valve’ and the lined flanged ball valve shall be related to that subtype.

This shall be specified as follows:

UID-1	Left hand object name	Fact UID	Relation type ID	Relation type name	UID-2	Right hand object name
10717	lined flanged ball valve	101	5282	shall have as possessed aspect a	16,428	DN nominal diameter of a lined flanged ball valve
16,428	DN nominal diameter of a lined flanged ball valve	102	1146	is a specialization of	1,745	nominal diameter of a valve

13.4.8 Step 8: Specify scales and units of measure

The example of Figure 123 illustrates that a template specification can define a list of allowed values, which is a list of qualitative (or quantitative) aspects. For example, for colors the allowed values may be red, green and blue, for nominal diameters the allowed values might be constrained to 15 mm, 20 mm, etc.

The latter qualitative aspects are qualitative classes that can be quantified on a scale, in this example a length scale, such as mm. Such qualitative aspects shall also be standardized in a formal language. This means that a number of concepts such as 15 mm, 20 mm, etc. are predefined concepts in the formal dictionary.

For example, the quantitative aspect ‘15 mm’ is defined in the formal dictionary as follows:

UID-1	Left hand object name	Fact UID	Relation type ID	Relation type name	UID-2	Right hand object name	UID of UoM	UoM
581029	15 mm	426	1726	is a qualification of	550018	distance		
581029	15 mm	832	5279	can be quantified on scale as	920268	15	570423	mm

Because they are defined already, including their mapping on scale to numeric values, the qualitative values can be used directly in the template specification, whereas the mapping to numbers on a scale can be retrieved from the formal dictionary as and when required.

In case such qualitative aspects are not yet present in the formal dictionary they can be added in the user environment in the same way as the above example.

After creation of a template all product types, aspects, qualitative aspects and relations types that are used shall be present in the formal dictionary or shall be expressed as subtypes of concepts in the formal dictionary.

13.5 Usage of product types for classifying individual items

The following expressions illustrates how product types can be used in a design.

Assume that product type XYZ is created according the above guidelines. And assume that the individual valve V-1 is created in a project, whereas it is specified that V-1 is a part of piping system-1 and assume that V-1 is classified as a product type XYZ.

These facts can be expressed in a product model of the valve as follows:

UID-1	Name of left hand object	UID of fact	UID of relation type	Name of relation type	UID-2	Name of right hand object	UID of UoM	UoM
1001	V-1	106	1190	is a part of	1002	system-1		
1001	V-1	107	1225	is classified as a	1003	XYZ		

Individual things may be classified multiple times. Thus the above classification might be in addition to an earlier classification that states:

V-1 is classified as a valve

13.6 Summary of relation types

This section provides guidelines in which the following relation types are used:

Relation type id	Relation type name	Required first role	Allowed role player	Required second role	Allowed role player
5282	shall have as intrinsic aspect a	conceptual possessor of an aspect	physical object	conceptually possessed aspect	aspect

5238	shall have a part with as possessed aspect a	conceptual possessor via part	physical object	conceptual intrinsic role of part	possessed aspect
1146	is a specialization of	subtype	concept	supertype	concept
1726	is a qualification of	qualitative subtype	concept (qualitative kind)	nature	concept
4714	can be a role of a	conceptual player of a role	physical object	conceptually played role	role
4751	shall be compliant with a	conceptually required compliancy criterion	information	conceptually specified	physical object
4846	shall be one of the	qualitative options possessor	aspect	qualitative options	collection of classes
4730	is an element in collection of kinds	collected kind	concept	collecting plural kind	collection of kinds
5279	can be quantified on scale as	conceptually quantified characteristic	characteristic	conceptual quantifier of a characteristic	mathematical space

Note 1: Sometimes an inverse phrase is used to refer to a relation type. For example, Figure 122 and Figure 123 use the inverse phrase for the name of relation type 4730 <is a collection of classes including> instead of the phrase mentioned in the above table.

Note 2: This document also refers to relation types that are used in the formal language taxonomic dictionary , but are not directly used for the specification of product catalogues.

14 Modeling principles and guidelines

14.1 Classification and specialization guidelines

This guideline is about how it is defined what individual things are and how new concepts (kinds of things) are defined. It also addresses how an entity type or an attribute type in an existing data model is represented in a formal language and how an instance is represented.

Guideline: classification

Every individual thing shall be classified by at least one explicit classification relation with a concept that is either selected from a formal dictionary or that is a user defined concept that is explicitly defined by a user as a proper subtype of a concept that is selected from the dictionary. Multiple classifications of the same individual thing is allowed.

The above also holds for the classification of individual relations. Thus each relation shall be classified by a kind of relation (relation type) that is selected from the formal dictionary and shall be applied in a semantically correct way. This means that it shall satisfy the constraints that are expressed for the kind of relation in the formal dictionary-ontology. For example, the constraint that the related things are of the correct type as defined by the types of things that can play the roles that are required by the relations.

Guideline: specialization or subtyping

Every (new) concept (or kind) shall be defined by a specialization relation with a concept that is either selected from the dictionary or that is a user defined concept that is defined as a specialization of a concept that is selected from the dictionary. A concept may be a subtype of more than one supertype and may be a supertype any number of subtypes.

Guideline: qualitative subtyping

Any qualitative or quantitative concept shall be defined by a qualitative subtype relation with a (conceptual) concept. Note that a qualitative subtype relation is a kind of a specialization relation.

Explanation

An entity type or an attribute type in a conventional data model is equivalent to a concept in a formal language. An instance that is an individual thing, such as the Eiffel tower, shall be classified by a concept or by one of its subtypes (conform guideline 1). For example:

the Eiffel tower is classified as a tower

Such a classification relation is equivalent to an instantiation relation between an instance and an attribute type.

An instance that is a qualitative or quantitative concept is represented in a formal language as a qualitative or quantitative concept.

The (implicit) instantiation relation in conventional models is represented explicitly in a formal language by either a classification relation or by a specialization or a qualification relation, depending whether the instance is an individual thing, a (conceptual) concept or a qualitative or quantitative concept.

The consistent application of this rule has as a consequence that the formal language concepts are all arranged in one consistent specialization hierarchy (= subtype-supertype hierarchy).

The application of this rule has also as consequence that aspects of concepts are inherited by a subtype concept from (all) its supertype concepts. This includes the inheritance of kinds of roles that it can play in relations of a kind. This means that it can be verified whether any relation is of a type that is semantically correct as being allowed according to the definition of that relation type and consistent with other relation types.

For example, if P-5601 is classified as a pump, then it follows from the classification relation that P-6501 is an individual pump, and from the specialization hierarchy above pump it follows that it is an individual physical object and thus only relation types are allowed that are defined for an individual physical object (or for one of its supertypes).

Examples

Name of left hand object	Name of relation type	Name of right hand object	Partial description (of left hand object)
P-6501	is classified as a	pump	
capacity of P-6501	is classified as a	capacity (mass flow rate)	
P-6502	is classified as a	bicycle pump	
bicycle pump	is a kind of	pump	that is intended to

			inflate tires of bicycles.
583.4 dm ³ /s	is a qualitative subtype of	volume flow rate	
RGB 60, 40,10	is a qualitative subtype of	color	

14.2 Assembly, compositions and collections

Single things and pluralities need to be distinguished. This guideline is about the criterion to determine whether something is a single thing or a plural thing. It is also about how can it be avoided that things are counted more than once in a bill of materials when some things are a part of or an element of more than one bigger whole.

Guideline: collections

A collection is a group of items that have no fixed position relative to each other and that have no specific role towards the whole.

The relation type used to describe the relation between the element and the collection is: <is an element of>. The inverse relation type is called: <is a collection of individuals for>.

Guideline: assemblies

An assembly is group of items that have connection relations among them and have a fixed position relative to each other and that form integral parts of the whole.

The relation type used to describe the relation between the part and the assembly is: <is a part of>. The inverse relation type is called: <is an assembly of>.

An item may be element of more than one collection and/or the same item may be part of more than one assembly. This means that relations are added, but the item is not duplicated. This avoids duplications in bills of material.

Guideline: composites

A composite is composed of components that may be assembled or arranged in a particular pattern without being assembled. Thus assembly and arrangement are subtypes of composite.

Guideline: unions

Multiple collections can be united into a collection that is a 'union' of the united collections. A union does not contain duplicates that may appear in the united collections. It uses the relation type 'is united in'.

Note: A collection typically can have as aspects: number of items, (total) mass, (total) price, but not a length of the collection.

Examples

Example of a collection of individual things:

Left hand object name	Relation type id	Relation type name	Right hand object name
collection of bolts B	1225	is classified as a	collection of items
collection of bolts B	4843	each of which is classified as a	bolt
bolt-1	1227	is an element of	collection of bolts B
bolt-2	1227	is an element of	collection of bolts B
collection of bolts B	2853	is a union of the elements of	collection of nuts and bolts C
collection of nuts N	2853	is a union of the elements of	collection of nuts and bolts C

A collection of classes (kinds of things):

collection of RGB base colors	1224	is a particular	collection of classes
red	4730	is an element in collection of classes	collection of RGB colors
green	4730	is an element in collection of classes	collection of RGB colors
blue	4730	is an element in collection of classes	collection of RGB colors

An assembly of individual things:

wheel-1	1190	is a part of	my car
motor-1	1190	is a part of	my car

A conceptual assembly (possible relation between members of classes):

wheel	1191	can be a part of a	car
motor	1191	can be a part of a	car

14.3 Aspects of parts with product decomposition

How should aspects of parts of products be specified when the decomposition of the product is also specified? For example a ‘wheel diameter’ of a car.

Guideline: aspects of parts

Specify an individual thing as being a part of the whole product and allocate the aspect to the individual thing that has the role as part of the whole.

Examples

wheel-1	1190	is a part of	my car
wheel-1	1727	has aspect	diameter-1
diameter-1		is classified as a	diameter

14.4 Aspects of parts without product decomposition

How should aspects of parts of products be specified without decomposing the product?

Guideline: aspects of parts without assemblies

Define an intrinsic aspect (a role of an aspect) and allocate the intrinsic aspect to the whole product.

Explanation

The preferred method is: Define a part and allocate the aspect to the part of the product (see the previous guideline).

In conventional systems, especially for procurement, products are often not decomposed and as a consequence aspects of parts are then recorded as aspects of the whole product. This requires that an aspect of a part is distinguished from others by specifying that the aspect is possessed by the part. This is usually done by adding the name of the part to the name of the aspect. Such a ‘intrinsic aspect’ is a role of an aspect that specifies it as by definition being an aspect of a part.

Examples

1. *Specify that a bearing of a pump can have a diameter.*

A bearing can possess a ‘diameter’, but in conventional systems the ‘bearing diameter’ is often recorded as being an intrinsic aspect of a pump.

The concept ‘bearing diameter’, or its synonym ‘diameter of a bearing’ need to be introduced in the formal language. The definition of kinds of intrinsic aspects requires three expressions that express what is by definition the case.

After that the relation between the intrinsic aspect of a part and the whole pump can be expressed. The expressions for such a specification are:

diameter of a bearing	5738	is by definition an intrinsic aspect of a	bearing
diameter of a bearing	5343	is by definition a role of a	diameter
diameter of a bearing	1146	is a kind of	intrinsic aspect
diameter of a bearing	5258	can be an intrinsic aspect of a part of a	pump

It is recommended to ensure that the associated knowledge is also defined. For example, the knowledge that a bearing can have a diameter. This should be done as follows:

bearing	2069	can have as aspect a	diameter

In the Upper Ontology of the formal language it should be defined what an intrinsic aspect is. This includes at least expressions such as:

intrinsic aspect	1146	is a kind of	role
intrinsic aspect	5343	is by definition a role of an	aspect

2. Specify the diameter of a driven end bearing of a pump

Driven end bearing is by definition a role of a bearing. So in this case the intrinsic aspect ‘diameter of a driven end bearing’ is not specified to be an aspect of a bearing, but of a role of a bearing.

Therefore, the formal language expressions for such a specification are:

diameter of a driven end bearing	is by definition a role of a	diameter
diameter of a driven end bearing	is a kind of	intrinsic aspect
diameter of a driven end bearing	is by definition a role of an aspect of a	driven end bearing
pump	can have a role of a part with as intrinsic aspect a	diameter of a driven end bearing

3. Specify the type of a part

Often applications allow or require that for one or more of the parts of an assembly it is specified of what kind such a part is. For example it may be required to specify a kind of bearing that is applied in a particular pump, without explicit modeling of the bearing as a separate component. Such a specification is a short-cut relation between the pump and the kind (type) of one of its parts. In other words it is a classification relation between an individual assembly and a kind that does not classify the assembly, but it classifies a part of the assembly. This requires a special short-cut relation which can be called <has a part that is classified as a>. For example:

P-101	has a part that is classified as a	roller bearing

To specify on a class level that for a pump it is required that the ‘type of bearing’ is specified, the concept ‘type of bearing’ need to be defined in the dictionary. Concepts such ‘type of bearing’ classify the role of subtypes of bearing is subtype-supertype relations. Therefore they are subtypes of the concept ‘subtype’. Thus the concept ‘type of bearing’ is defined as follows:

Name of left hand object	Name of relation type	Name of right hand object	Partial description
type of bearing	is a kind of	subtype	that is played by a kind of bearing

Validation method

The relation type definitions specify the kind of role that is played and the kind of (the nature of) the role players that are or may be related by such a relation. This shall be consistent with the specialization hierarchy of the related objects.

For example, the relation type ‘can have a part with as intrinsic aspect a’ requires that the first role player is a physical object and that the second role player is a role. Therefore, it should be verified whether ‘pump’ is a subtype of ‘physical object’ and whether ‘diameter of a bearing’ is a subtype of ‘role’.

14.5 Subtypes of aspects and roles of aspects

What is the difference between a subtype of an aspect and an intrinsic aspect (a role of an aspect)? For example: is an internal diameter, a boiling point, etc. a subtype of property or a role of a property?

Guideline

When a possessor kind of physical object as well as an aspect is included in the definition of a concept (a kind of extrinsic aspect or intrinsic aspect), then it is a kind of aspect that is intrinsic to the kind of object. This means that the concept is a kind of role, else it is a kind of (extrinsic) aspect. In other words, if something is by definition possessed by a particular kind of physical object, then it is an intrinsic aspect.

Explanation

Adjectives in names of kinds of aspects result in subtypes of aspect. Physical objects in names of aspects result in subtypes of intrinsic aspect. The used kinds of aspects and kinds of physical objects must be defined in the dictionary.

Example

Name of left hand object	Name of relation type	Name of left hand object	Description
internal diameter	is a kind of	diameter	(being an aspect)
pipe diameter	is a kind of	intrinsic aspect	of a diameter when being possessed by a pipe
pipe diameter	is by definition an intrinsic aspect of a	pipe	
pipe diameter	is by definition an intrinsic	diameter	
pipe internal diameter	is by definition an intrinsic	internal diameter	

Note: the kind of aspect ‘diameter’ and the kind of physical object ‘pipe’ are already defined in the formal dictionary.

14.6 Properties of properties (not allowed)

There are correlations between properties, but does that mean that properties can have (or be possessed by) other properties?

Properties are subtypes of aspects. Thus this question can be generalized as: can aspects have aspects?

Guideline

In general: Aspects cannot have aspects, except that they can have roles.

Two aspects for which one is thought to be an aspect of the other should both be modeled as an aspect of a possessing totality. It may be specified however that the aspects are *correlated* (this correlation causes that people might think of properties of properties), for example, when one property constrains the other or one is dependent on the other. A *correlation* between properties is specified through a correlation relation between the aspects. When more than two aspects are correlated the correlation is modeled as a higher order relation.

Examples

For example, a temperature may be a limit that is the end of a temperature range. Both have to be specified to be possessed by the same possessor. Then the limit can be defined to be correlated with the range by a relation that indicates for example that the limit is a lower limit of the range.

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
1	thermometer	can have as property a	temperature range
2	thermometer	can have as property a	upper temperature limit
3	upper temperature limit	is an upper boundary of a	temperature range

14.7 Collections of aspects

Aspects, especially qualitative aspects are often specified to be elements of collections of aspects. Typically such collections are intended to be used as pick-lists or lists of allowed values.

Special collections are: ordered lists (pairs, triples, etc.) which can be homogeneous or inhomogeneous.

Guideline

Allowed nominal diameters for a welding neck pipe flanges according to DIN or according to ASME.

14.8 Correlations between aspects

Examples

1. Colors defined in the RGB system

Red, green and blue are qualitative aspects of color.

Subtypes of red are for example: Magenta-1 = (R=76, G=12, B=14), Magenta-2 = (R=86, G=22, B=24).

Magenta-1	1726	is a qualitative subtype of	color	RGB(76,12,14)
Magenta-2	1726	is a qualitative subtype of	color	
Magenta-1-RGB	1146	is a kind of	3D coordinates point	
Magenta-1	4786	is a result of	Magenta-1-RGB	
76	5672	is a first coordinate of	Magenta-1-RGB	R=76 (76,0,0)
12	5673	is a second coordinate of	Magenta-1-RGB	G=12 (0,12,0)
14	5674	is a third coordinate of	Magenta-1-RGB	B=14 (0,0,14)

This is a particular case of the general equation:

$y = f(a, b, c)$, where $f(a, b, c)$ is a function or correlation in which a , b , and c are the parameters. Thus the general correlation is expressed in the formal language as follows:

y	4786	is a result of	Magenta-1-RGB
R=76 (76,0,0)	4962	is a parameter in	Magenta-1-RGB
G=12 (0,12,0)	4962	is a parameter in	Magenta-1-RGB
B=14 (0,0,14)	4962	is a parameter in	Magenta-1-RGB

2. Total width of an assembly is the sum of the widths of some parts.

14.9 Dictionaries and taxonomies versus knowledge models

How does a manufacturer's model with its detailed specification (constraints) relate to a concept in a taxonomic definition model?

Which concepts (including relation types) are used for such descriptions (in these models).

Guideline

A detailed model, such as a manufacturer's model (or model and size) is a type of thing that is a detailed subtype of a concept in the taxonomic dictionary. To distinguish a (sub)typing relation from the ordinary specialization relation there is a subtype of the specialization relation available, called 'is a type of' and a further subtype, called 'is a model of',

which means ‘is a manufacturer’s model of’. The defined type (the manufacturer’s model) will inherit aspects from the higher-level concepts, but it can have additional aspects.

Explanation

If a type or ‘manufacturer’s model’ product type is included in a taxonomy hierarchy and defined in a detailed product type model, and included in a knowledge model, then it is still just one concept. That concept may appear in a dictionary and in a knowledge library as well as in a product catalogue, but the thing has still one and only one unique identifier, being its UID.

Views on the model (the ‘cloud’ around it) can be distinguished by selecting various context(s) that are applicable and any application can specify its own criteria for the scope of a view. For example a knowledge model view can be further constrained by specifying that only relations of specific types ‘can have as aspect a’ should be made visible.

The three contexts use the following relation types:

dictionary: 4685 is defined by
 taxonomy: 1146 is a kind of (= is a subtype of) knowledge model

Examples

The concept ‘pump’ can for example appear in a dictionary, in a taxonomy in a knowledge model as well as in a product model.

Context	UID of left hand object	Name of left hand object	Name of relation type	Name of right hand object
dictionary	370,130	pump	is defined as	a rotating equipment item that is capable of increasing a pressure and a flow rate of a fluid.
taxonomy	370,130	pump	is a kind of	rotating equipment item
knowledge model	370,130	pump	can have as part a	bearing
knowledge model	370,130	pump	can have as aspect a	capacity (volume flow rate)
product model	370,130	pump	is a classifier of	P-101
taxonomy	1,000,001	pump model A	is a type of	pump
knowledge model	1,000,001	pump model A	can have as aspect a	model A capacity
knowledge model	1,000,002	model A capacity	can be qualified as	10 dm ³ /s

14.10 Functions and locations of physical objects in assemblies

How to describe that a physical object has a particular function or location as part in an assembly? This applies to kinds of things as well as individual things.

Guideline

The function or location of a thing shall be represented by a role of an occurrence (activity) or by a usage or role in a relation.

Examples

To enable the modeling of the different usages of the four wheels of a car it is required that the concepts of the four different kinds of roles for a wheel need to be present or need to be added to the dictionary. When those concepts are available it can be specified that a car shall have four wheels in those four different kinds of roles.

For example, the concepts ‘front’ wheel’ and ‘left front wheel’ are defined as follows:

front wheel	1,146	is a kind of	usage
front wheel	4,714	is by definition a usage of a	wheel
front wheel	5403	is by definition a located at a	front side
left front wheel	1,146	is a kind of	front wheel
left front wheel	5403	is by definition a located at a	left side

The requirement that a car shall have a left front wheel is specified as follows:

car	5,237	shall have as part a	left front wheel

Note: in the relation type 5237 <shall have a part with as role a> would be semantically more precise. However relation type 4989 <shall have as part a> is sufficient for a correct interpretation, because from the definition of ‘left front wheel’ it can be concluded that ‘left front wheel’ is a role of a wheel.

An individual left front wheel is specified as follows:

w-1	1,190	is a part of	c-1
w-1	1,225	is classified as a	wheel
w-1	1,588	has a role as a	left front wheel

The latest relation is a classification by role, whereas the role ‘left front wheel’ is by definition a usage (role) of a wheel. This implies that the explicit classification relation (at line 2) is superfluous.

14.11 Type of thing: Requirements to classify things

On standard forms there is often a field with the name ‘type’. How should that requirement be specified? Is this an aspect of the thing?

Guideline

Use the relation type ‘shall be classified as a subtype of’ or ‘shall have a part that is classified as a subtype of’ to specify the requirement to classify the item or a part of the item by a subtype of the specified concept.

Explanation

Fields on standard forms, such as on data sheets, not only specify an object type, but they specify primarily a requirement for a kind of fact. In other words they specify a requirement for a relation of a particular type. Furthermore, the fields do not always specify aspects of the thing about which the sheet provides information. In this case a type is not an aspect of an individual item, but is its classification. Thus the term ‘type’ specifies a requirement for a classification of something by a subtype of a particular concept.

Examples

Assume that on a ‘pump data sheet’ there is a field with the name ‘type’ and another field with the name ‘bearing type’. The first field expresses apparently a requirement to fill in the name of a subtype of the concept ‘pump’. A proper value would be ‘centrifugal pump’. The second field expresses apparently a requirement to fill in the name of a subtype of the concept ‘bearing’, whereas such a bearing is apparently a part of the pump.

An example of proper value is ‘ball bearing’.

These requirements are expressed in the formal language as follows:

pump	4,994	shall be classified by a subtype of	pump
pump	5,392	shall have a part that shall be classified by a subtype of	bearing

Note: The latter relation type is a short-cut for the full specification that a pump shall have a bearing and that a bearing shall be classified by a subtype of bearing. This short-cut relation type is available to enable the specification of such a requirement in case no decomposition is supported or wanted.

14.12 How to add relation types

Relation types can often be found by searching on one of the kinds of roles that the related things play. The name of such a kind of role is usually a term within the phrase that denoted a relation type. But, if we still cannot find a suitable relation type, how can we add a new one?

Guideline

Identify a relation type that is more generic than the required relation type and verify the suitability of its subtypes.

If not successful, then verify whether the role that the related objects play do appear in the formal dictionary. If that is the case, then determine whether there is a suitable relation type that requires that role (or its supertype or subtypes).

If still no suitable relation type is found, then add the required relation type as an extension of the formal relation types according to the applicable guidelines for the extension of the formal language.

This also includes providing feed back to the formal language manager with the proposal to include the extension in the standard formal language.

Explanation

The relation types in the formal language are arranged in a hierarchical network of subtype-supertype relations, just as any other concept in the taxonomy. This also holds for the role types that are required by the relation type and for the object types which members can play such roles. Therefore, these three elements of a relation type definition should be properly defined, so that software can verify the semantic correctness of the formal language expressions.

Examples

Assume that you want to express that object A is indirectly connected to some physical object.

First it is verified whether a suitable relation type is available by searching on the string elements 'is' and 'connect'. A good browser will find various alternatives, such as 'is connected to', 'is logically connected to', etc. (the Gellish Search Engine and the Gellish Communicator are examples of software that is suited to support such a search). Assume that these found options are not acceptable to you.

A search on 'connected' leads also to 'is connected to' and does not provide other options. Thus, you conclude that you have to add a new relation type to the formal language.

Such a new relation type, called 'is indirectly connected', can be added as a subtype of the found supertype (connection), whereas the formal language synonym and inverse synonym can be added as follows:

indirect connection	1,146	is a kind of	connection	that is ...
is indirectly connected to	1,981	is a synonym of	indirect connection	
is indirectly connected with	1,986	is an inverse of	indirect connection	

The above definition implies that an indirect connection inherits two roles, connector and connected, from its supertype, as is defined in the formal language upper ontology. That upper ontology also specifies that these roles can be played by any physical object. Now assume that it is wanted to constrain the allowed role players of the first role to only objects of the type A. Then it is necessary to define that the subtype relation type requires as first role a subtype of 'connected', possibly called 'indirectly connected', whereas such a role can only be played by an object of type A.

This is specified as follows:

indirect connection	1,146	is a kind of	connection
indirect connection	4,731	requires as first role a	indirectly connected
indirectly connected	1,146	is a kind of	connected
object type A	4,733	can have a role as a	indirectly connected

A similar subtype role and role player can be defined for the second role.

14.13 Information about a thing versus a thing itself.

How to make a distinction between a ‘cloud’ or model of information about a thing and the ‘thing itself’? In other words, how to distinguish between a network about a node and the node itself?

Guideline

A thing and the information about the thing are two different things that shall have different UIDs. They are related by the relation type <is information about> or one of its subtypes or its equivalent relation on conceptual level.

Explanation

A thing, such as a physical object is something with an identity, whereas it is represented in a formal language by a UID. However its properties, such as its diameter or mass are things in their own right represented by their own UID, which are distinguished from their possessor, although they are intrinsically related with it. Other aspects or parts may be extrinsically be related. The expressions of those facts as relations together form a collection of relations (or an information model) about the physical object.

Example

For example the physical object P-101 is a pump, whereas the specification of that pump is an information collection about the pump.

UID of fact	Name of left hand object	UID of relation type	Name of relation type	Name of right hand object
101	P-101	1,146	is classified as a	pump
102	Info about P-101	1,146	is classified as a	qualitative information
102	Info about P-101	5870	is a specification of	P-101

14.14 Roles of aspects and roles of physical objects

How to define and use roles and relations between physical objects and roles of aspects. For example: ‘input terminal voltage’ and its relation to terminal and to voltage and to the possessing physical object.

Guideline

A subtype of intrinsic aspect is defined by three relations: a specialization relation with intrinsic aspect or one of its subtypes, a ‘by definition being

possessed’ relation with the possessing kind of physical object and a ‘by definition being an intrinsic aspect’ relation with the kind of aspect.

The possessing kind of thing is typically a kind of physical object, but may be a kind of role of a physical object.

Explanation

Aspects can have roles of various kinds. One kind of role is ‘by definition being possessed’ or, in other words, being ‘intrinsic’ or clearer: being an ‘intrinsic aspect’. The name of a subtype of intrinsic aspect is typically a combination a name of a kind of physical object and a kind of aspect. For example ‘pipe diameter’.

Individual physical objects are normally related to individual aspects. Typically they are not explicitly related to roles of aspects, because there is a 1:1 correspondence between an individual aspect and an individual role. However, this differs from the conceptual level, because a kind of aspect can be related with various kinds of physical objects, whereas requirements or constraints may be specified for such an aspect in only some of its roles. In other words, constraints often hold only for some kinds of intrinsic aspects.

Example:

Context	UID of fact	Name of left hand object	UID of relation type	Name of relation type	Name of right hand object
definition model	141	pipe diameter	1,146	is a kind of	intrinsic aspect
definition model	141	pipe diameter	5,817	is by definition an intrinsic	diameter
definition model	141	pipe diameter	5,738	is by definition an intrinsic aspect of a	pipe

Note: instead of a kind of physical object, the possessor may also be a kind of role of physical object. For example, in the above example the pipe may be replaced by an ‘inlet pipe’, whereas an inlet pipe is by definition a role of a pipe.

14.15 Individual things versus kinds of things

The semantics of relations between individual things differs from corresponding relations between kinds of things. Therefore, individual things shall be distinguished from kinds of things. What is a clear criterion to distinguish individual things from kinds of thing?

Guideline

The subtle difference between an individual thing and the associated kind of thing is that a kind of thing is not so much a specific thing, such as my chair or my computer, but rather an abstract commonality among things that can be used for classifying individual things as chairs or computers. Kinds of things may include a specification from which many individual objects are made. For example a “Volvo S40”.

A kind of thing is defined as a commonality that is shared by things that belong to the kind. You can never point to a class or kick a class, whereas you can point to or kick many individual things.

Kinds of things are also called concepts, classes or categories and their subtypes ‘types of things’. Examples of individual things (instances or exemplars) are: my chair, computer #32131, pump #12 at a particular location, our planet the earth and the Eiffel tower. Other examples are: a particular activity, a project, a process, a person (such as John), a company, a measurement, a property and a fact.

In actual practice people are often not aware of this distinction between individual things and kinds of things. For instance, people may say that they talk about ‘an airplane’ when they are actually describing either the kind of thing ‘airplane’ (that which makes any airplane an airplane) or they may be describing the particular individual thing with which they travelled and that is classified as an airplane.

Examples of kinds of things, such as pump, vessel, project, task, color, red, etc. are all subtypes of ‘individual thing’, which is a subtype of concept. Those subtype concepts can be used to classify any individual thing. In fact a classification of an individual thing implicitly classifies something as an ‘individual thing’.

For example, the expression <P-6001 is a pump> implies that P-6001 is an individual physical object and thus an individual thing as well.

14.16 Imaginary objects versus installed, materialized objects

During a design process people create ‘designs’, which are descriptions or specifications of an imaginary physical object (or an imaginary process or both) that may be realized later. How do we distinguish between a model of such an imaginary thing and its aspects and a real thing and its aspects?

Imaginary physical objects (sometimes called functional objects, not to be confused with roles of physical objects), and materialized physical objects (also called real physical objects), are both individual objects. However, the first one has properties that are deemed, typically during a design process, whereas a materialized physical object has properties that can be measured, because the objects are fabricated or natural objects. As long as imaginary objects are realistic, they can be classified by the same kinds of things as the real objects.

An example of an imaginary physical object is:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
101	P-101	is classified as a	pump

An example of a materialized physical object is (as a part of a built installation):

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
102	A.123.456	is classified as a	pump

Both can also be classified explicitly as either imaginary or materialized as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
103	P-101	is classified as a	imaginary physical object
104	A.123.456	is classified as a	materialized physical object

Because in a model usually either all individual things are imaginary objects or they are all materialized objects, it is desirable to prevent such additional classification relations for all individual things. Therefore, the Gellish Expression Format has a separate table column in which it can be specified whether a left hand object is either ‘imaginary’ or ‘real’. This column represents the classification relations. Furthermore it is possible to set a default value for all objects in a whole model in a message file.

Both objects can also be classified by further subtypes of the higher level kinds of physical objects, although usually a functional physical object is not classified by a manufacturer’s model (although it is not forbidden to do so),

whereas a materialized physical object typically has a serial number and is classified by a manufacturer's model. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
105	P-101	is classified as a	horizontal triple screw pump
106	A.123.456	is classified as a	Allweiler model SNH 80

Whereas the supplier may specify that:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
107	Allweiler model SNH 80	is a kind of	horizontal triple screw pump

For both objects it can also be specified which function they have, one in the imaginary world (as designed), the other in the real world (as installed).

For example,

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
108	P-101	is a performer of	pumping fluid F1
109	A.123.456	is a performer of	pumping fluid M1

In addition to that the following can be specified:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
111	pumping fluid F1	is classified as a	pumping
112	pumping fluid M1	is classified as a	pumping
113	F1	is a subject in	pumping fluid F1
114	M1	is a subject in	pumping fluid M1
115	F1	is classified as a	stream
116	M1	is classified as a	stream

This means that 'pumping fluid F1' is an imaginary process of pumping an imaginary fluid F1 (possibly during a simulation of the process), whereas 'pumping fluid M1' is a real process of pumping a real (materialized) fluid M1.

Typically, a materialized physical object is installed at the place where the functional physical object is thought. In other words: the real object is

installed conform the requirements expressed for the imaginary object. This can be specified as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
117	A.123.456	is installed at the place of	P-101

When the real object is fabricated explicitly according to the specification of the imaginary object, this can be specified as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
118	A.123.456	is a realization of	P-101

Typically, the product model that describes a functional physical object in the imaginary world (such as a model of P-101) is used as a specification for the delivery of a materialized physical object (such as one with serial number A.123.456). This means that the materialized physical object is intended to comply with the specification and to perform the equivalent function in the real world (as realized) at least as good as the imaginary object does in the imaginary world (as designed). Whether it actually complies (to an acceptable level) is recorded separately.

14.17 Functions as roles of activities

How are functions modeled?

In natural language, the term function has various meanings. In the Gellish formal languages the following meaning of function is defined.

Function of a physical object

A function of a physical object denotes an occurrence (activity, process or event) that is intended to be performed or enabled by the physical object. For example, if P-101 is designed to be suitable to pump a liquid stream S1, then it can be stated that P-101 fulfills the function ‘pumping of S1’. This is expressed in the formal language as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
119	P-101	fulfills function	pumping of S1

Strictly speaking such a function is a role of being preformed or enabled, which is a role of an occurrence.

Thus functional modeling implies modeling of occurrences, activities and processes, while specifying their performers or enablers.

14.18 Product specification versus process specification

In Systems Engineering a distinction is made between product specification and process specification. What are the typical differences between models of those two types.

For a process specification we can distinguish between two kinds of processes:

- Processes that are or shall be performed by the (solid) physical objects and/or in the (fluid) physical objects that are modeled.
- Processes that are performed by the organizations that design or manufacture, install and operate, inspect or maintain physical objects.

The first kind of modeling results in a process model of a physical process, the second kind results in a business process model. Both kinds of processes are modeled in a similar way, although physical processes can be either continuous or discrete processes, whereas business processes are typically discrete processes.

A process can be decomposed in components, also called process steps, stages, activities or tasks, each of which is typically classified by a subtype of occurrence or activity. For example, a decomposition of a business process, such as a project (being a subtype of business process) may be modeled as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
120	P1	is classified as a	project
121	P1	has as part	Stage-1
122	Stage-1	is classified as a	project stage
123	P1	has as part	Stage-2
124	Stage-2	is classified as a	project stage

It is possible to specify the sequence between such parts of a process. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
125	Stage-1	is a predecessor in time of	Stage-2

An individual physical process can be defined in a similar way, for example as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
126	Cleaning water stream S1	is classified as a	purification
127	Cleaning water stream S1	has as part	Cleaning step-1
128	Cleaning step-1	is classified as a	filtration
129	Cleaning water stream S1	has as part	Cleaning step-2
130	Cleaning step-2	is classified as a	biological treatment

It is also possible to specify the relations between process steps or activities by specifying physical objects (including also streams and signals) and their roles as inputs, outputs (as is done for example in the IDEF0 method). If a physical object has a role as output from one activity and the same physical object has also a role as input in another activity means that the first activity precedes the latter. Such relations are specified for example as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
131	Cleaning step-1	has as input	S1
132	Cleaning step-1	has as output	S2
133	Cleaning step-1	is supported by	F-101
134	S1	is classified as	water
135	S2	is classified as	water
136	F-101	is classified as a	filter
137	Cleaning step-2	has as input	S2
138	Cleaning step-2	has as output	S3
139	S3	is classified as	potable water
140	S3	has property	pH of S3
141	pH of S3	is quantified on scale as	pH
142	pH of S3	is quantified as	7.1

The physical objects, such as S1 and F-101, are products, but may also include documents, signals, etc.. Those physical objects can be modeled themselves. In this way process and product modeling can be integrated.

Furthermore, processes and activities may be integrated with people in various roles. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
151	Activity-1	is performed by	John
152	Activity-1	is classified as a	repair
153	Activity-1	has as subject	P-101
154	John	is classified by role as	engineer
155	Activity-1	is managed by	Peter
156	Peter	is classified by role as	project manager

Note: The above classifications include classifications by substance (water) and classifications by role (engineer, project manager), whereas only in the latter two cases the subtype relation type is used.

14.19 Life cycle approach of products

A product life cycle approach is intended to record information about the product from ‘cradle to grave’. This means from the first idea about a product (or a process that shall be performed by some product) to its complete design, manufacturing, installation, operation, maintenance and demolition. How is such a life cycle modeled?

Typically the life of a product starts with the ‘function’ that shall be performed. Such a function is a process or activity that need to take place. In the process industries, this is usually a process that has one or more fluids as input and other fluid(s) as output. In other industries, the process typically involved modification or transport of solid items. In both cases the fluids as well as the solid items can be regarded as products. However, if we talk about a product life cycle we usually mean the life cycle of the facilities that are used to enable or perform such processes.

Thus the modeling of a life cycle of a product typically starts as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
161	Process-1	is classified as a	process
162	Process-1	has as input	S1
163	Process-1	has as output	S2
164	Process-1	is performed by	P-1

Note that at a very early stage it may be that no decision is made yet about the kind of physical object will be the performer of the process. We only know that there is a performer and that the performer is a physical object. So, depending on the options that are left open we may know that

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
165	P-1	is classified as a	physical object

Or that it is already classified by a more specialized kind of physical object. For example,

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
166	P-1	is classified as a	pump

During the conceptual and later detailed design additional facts about P-1 are added, and a further detailed classification may be added as well. This means that a growing product model is created, up to a full specification that is suitable as a basis for procurement of a materialized real product. Furthermore, the product model may include documents about the product and the product may become part of a bigger model, such as a Product Information Model.

A next stage in the life cycle is that a materialized physical object is created or selected to fulfill the function of the product as designed (the functional physical object). This real product is inspected, measured and may be modified and maintained, whereas the product as designed (the imaginary product) may be adapted to the reality. This process of modifying a design is usually called: the specification of the object (the imaginary object and its documentation) is made ‘as-built’.

The life of a product is terminated when all classification relations have the status ‘deleted’ or ‘history’ and that occurred at the date-time of the latest change (which is the change of status).

14.20 Modeling of information from source documents

Information as in classical documents and drawings can be presented on paper copies as well as in electronic files in various formats. For example, they can be available as scanned images, pdf-files, doc-files, dwg-files, etc. The content of such physical files can be expressed in natural language and drawings, but can also be expressed partly or completely as computer

interpretable models. How is such information modeled and related to the described thing?

If a whole document remains in natural language, then a whole document can be related to the object about which it provides information. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
170	T-123.456	contains information about	P-101

Object to text relations

When a document is partially modeled, then the content is decomposed in sections or paragraphs that represent information units, whereas those components remain in natural language. Each component may for example express a requirement. Those components, such as paragraphs, are typically not stored on separate files, but they are stored as pieces of text, whereas such ‘textual objects’ are integrated as part of the model. Each such an information component can be related to the object or kind of object about which it expresses a piece of information or a requirement. For example, a standard document ABC contains a paragraph that specifies a requirement Req-1 for centrifugal pumps. This is modeled as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
171	Req-1	is information that is included in	Standard-ABC
172	Req-1	is classified as a	requirement
173	Req-1	is described as	A centrifugal pump shall ...
174	Req-1	is a specification for a	centrifugal pump

Object to document relations

The fact that an individual object is described in a document is recorded using an <is described in> relation type, which is the inverse expression of the above mentioned relation <contains information about>. For example, the fact that P-101 and P-101 appear on the same sheet of a multi-sheet piping and instrumentation diagram (P&ID), together with the qualification of (the content of) the document can be recorded as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object

175	P-101	is described in	SH12-345-sh.1-Rev.1
176	P-102	is described in	SH12-345-sh.1-Rev.1
177	SH12-345-sh1-Rev.1	is classified as a	pipng and instrumentation diagram

The information may be part of a bigger document or aggregation of documents. For example, a drawing may be one of the sheets in a multi-sheet drawing, or a document may be included as an appendix to a bigger document. This means that the information is included in other information. This is modeled as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
178	SH12-345-sh.1-Rev.1	is a part of	SH12-345-Rev.1
179	SH12-345-sh.2-Rev.1	is a part of	SH12-345-Rev.1

As long as a document is being created its content is growing and changing, until a moment that a particular version is issued through the creation of a formalized file that is circulated for comment, for verification or for approval by one or more parties, or is issued in an approved state. From that moment there is not only a document with a name, but also one or more files with names. It is recommended that formal naming conventions are applied for both. The partial information as well as the aggregated information can be presented on multiple electronic files in various formats. Therefore, in addition to the relation between physical objects and the qualitative information (document), also the relation between the information and the files should be recorded, together with the location of the files in electronic directories. This is done as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
181	SH12-345-sh.1-Rev.1	is presented on	DR-1234.dwg
182	SH12-345-sh.1-Rev.1	is presented on	DR-1234.pdf
183	DR-1234.dwg	is classified as a	electronic data file
184	DR-1234.pdf	is classified as a	electronic data file
185	DR-1234.dwg	is an element of	http://www.part11.net/examples/
186	DR-1234.pdf	is an element of	http://www.part11.net/examples/

14.21 Roles of objects (classification by role and by nature)

Individual objects are normally classified by what they are, irrespective of how they are used. This means that they are classified on the basis of properties that can be observed about the object, even when the object is stored in a warehouse and is not used in any application. Such a classification is called a classification by nature. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
190	B1	is classified as a	roller bearing

However, a physical object can also be classified according to its usage in an actual or possible situation. This means that the object does not change when it stops being used in such a way. Therefore, this kind of classification is in principle temporal. Examples are:

An individual wheel at the left front side of a car (e.g. w-1) can be classified by nature as a wheel and by role as a 'left front wheel'. This is specified as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
191	w-1	is a part of	c-1
192	w-1	is classified as a	wheel
193	w-1	is classified by role as a	left front wheel

Note: If an individual role is made explicit as an object, then the role can be classified by nature, using a normal classification relation. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
196	W-1	has a role as	R-1
197	R-1	is classified as a	left front wheel

An individual person (John) can be classified by nature as a person and by role as a student and as a customer as well. This is specified as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object

198	John	is classified as a	person
199	John	is classified by role as a	student
200	John	is classified by role as a	customer

14.22 Inheritance of facts

The definition of a specialization relation between kinds of things (classes), also called a subtype-supertype relation, implies that the whole definition of the supertype kind of thing is by definition also applicable for the subtype of that kind of thing. In other words, all facts that are (by definition) true for a kind of thing, such as properties and composition, are also the case for all its subtypes. This is informally expressed by the statement that all subtypes inherit all the facts that are by definition true for all their supertypes.

The arrangement of kinds of things in a taxonomy (subtype-supertype hierarchy) is based on this inheritance principle.

For example, a centrifugal pump has by definition as part at least one impeller, and operates by definition according to the centrifugal principle. ISO 15926-4 states that a line shaft pump is a kind of a centrifugal pump. Thus a line shaft pump inherits from its supertype centrifugal pump that it has at least one impeller and operates according to the centrifugal principle. This means that these facts don't need to be specified again for all the subtypes.

Note, that when a kind of thing has two supertypes, then it inherits from both parents.

This inheritance mechanism is particularly powerful for detailed subtypes, such as catalogue items and manufacturer's models.

This mechanism puts constraints on the facts that are allocated to kinds of things in the taxonomy (subtype-supertype hierarchy). It means that facts should not be stated about kinds of things if those facts are not the case for all the subtypes of a kind of thing.

Note that an individual thing has a classification relation with a kind of thing and not a specialization relation. Therefore it does not inherit facts from the kind of thing. Depending on the basis for classification there are two possible situations:

An existing individual thing is described by a number of facts then those facts (possibly observations) can be used as the basis for its classification.

This means that the facts about the kind of thing are criteria for the verification whether the individual thing is indeed of that kind.

A new individual thing is deemed to be of a particular kind. For example, because a designer wants it to be as such. Then the facts about the kind of thing are intended to be used to derive facts about the individual thing.

14.23 Quantifiable versus qualifiable and literal properties

Some properties can be quantified by a number, typically on a scale. Typical examples are:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
201	L1	is quantified as	55
202	L1	is quantified on scale	mm
203	T1	has on scale a value greater than	37
204	T1	is quantified on scale	degC

Numbers of items in a collection do not need a scale. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
205	Number of bolts in stock S1	is quantified as	75

Other qualities are qualified by values that are qualitative properties. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
206	C1	is qualified as	red
207	R1	is qualified as	dangerous

Qualitative properties, such as red and dangerous, are kinds of properties that are qualifications of properties, such as color and danger (dangerousness). They should be defined as such in the dictionary in the following way:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
208	red	is a qualitative subtype of	color
209	dangerous	is a qualitative subtype of	dangerousness

Note that this is similar to the qualification of qualitative information, such as textual requirements as are described above.

Some qualities typically have only two values, or only the two extremes are used and the intermediate values are neglected. In conventional applications such qualities are often qualified by YES and NO. In such cases it may be that the qualitative value, such as ‘dangerous’, is used to ask whether something is indeed dangerous or not, whereas the user is asked to respond by either YES or NO. It is recommended in such cases to model the information in a semantically explicit way as follows: first determine the property, for example the value is about ‘dangerousness’, then use the value ‘dangerous’ or ‘safe’, whereas the question determines which one of the options is represented by YES.

Qualitative properties, sometimes called ‘literals’, are property values that may be qualified by proprietary values such as ERP key values, or private identification codes.

14.24 Scales and UoM's

Units of measures (UoMs) are qualification of scales. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
210	mm	is a qualitative subtype of	length scale
211	inch	is a qualitative subtype of	length scale

Properties can be expressed on particular scales. For example:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object
212	diameter	can be quantified on a	length scale

The above implies that any diameter can be quantified using at least either mm or inch scales. Such options may be constrained to explicit scales in a particular context as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object	Validity context
213	diameter	shall be quantified on scale	mm	My project

Individual properties can be quantified by a number on a scale by classifying the relation by a scale (unit of measure). This can be done on a single line in an Expression Table as follows:

UID of fact	Name of left hand object	Name of relation type	Name of right hand object	Name of unit of measure
214	D-1	has on scale a value equal to	30	mm

14.25 Synonyms versus preferred terms

Terms (text strings) shall uniquely denote a thing within a language and a language community. For example, within the English language and within the geographic language community (context) the term bank is used only once and then means a piece of land along a shore. To distinguish the meaning of such a term from its homonyms (the same term with another meaning) either the language or the language community (or both) shall be different.

To simplify the use of synonyms and homonyms this standard denotes a thing by a unique identifier (UID) that is unique within ISO 15926. This mechanism allows any thing to be denoted by more than one term (or even be nameless). The relation between them is recorded by a synonym relation or an equivalent relation type. The language and language community of the left hand term in a synonym relation indicates the naming context in which the term is the preferred term.

This means that a company or an application system may denote things by their own particular names or codes. To relate those terms to the terms used by others they should specify synonym relations, indicating their own company name or system as the language community of their terminology. For example, company X may use the term dwelling whenever they refer to a house and in the ERP system of company Y a centrifugal pump may be indicated by the code PUMP_C. This is recorded as follows:

UID of fact	Language community	Name of left hand object	Name of kind of relation	Name of right hand object
221	Company X	dwelling	is a synonym of	house
222	Company Y	PUMP_C	is a synonym of	centrifugal pump

By specifying the language community (Company X and/or ERP system of Y) it is possible to indicate the preferred terms to be used when displaying or reporting facts about these things.

14.26 Generic concepts, types and manufacturer's models

Usually, there is no need to make a precise distinction between concept (kind of thing), or class or type and these terms can be used as alternatives without misinterpretation. However, there are exceptions where the precise distinction is valuable.

A kind of thing is a synonym for class: both denote the same UID. In this context the preferred term is kind of thing, because the term class has an explicit meaning in the IT community, which is not what is meant in this standard. With kind of thing this standard means just the kind of thing itself and not a collection of facts about the kind of thing. In conventional IT terminology a class is something together with a number of facts about the thing.

By types this standard means subtypes of generic kinds of things (subtypes of classes). A type has as additional characteristic that it has one or more fixed qualitative properties. For example, hexagonal bolt or M6 bolt are both called types of bolt, because in one case the shape of the head and in the other case the diameter of the shaft are fixed qualitative values. If the kind of thing is very specialized, so that it is a kind of thing which specification is made specifically by a manufacturer for the kind of product that he fabricates, then the class is called a manufacturer's model. For example, a Volvo S40 is a manufacturer's model.

However, usually we mean an information model when the term model is used.

An information model is a collection of facts about a kind of thing or about an individual thing. There are various kinds of models:

- **Product models.**
A product model is a collection of facts about an individual (physical) product, its behavior and its documentation.
- **Product Information Models.**
A Product Information Model, such as a Plant Information Model or a Building Information Model (BIM), etc. is a collection of facts about a product and possibly about its operation under various conditions

(process cases), feeds and products. The scope may include its total life cycle and documentation, including requirements specification, process and mechanical design in 1D, 2D and 3D, cost estimation, procurement, fabrication, construction, commissioning, start-up, operation, control, inspection, maintenance and demolition.

- Definition models.

A definition model is a collection of facts that consists only of facts about a kind of thing, whereas the facts are by definition the case. Such definitions are typically expressed by <is by definition ...> or <has by definition ...> relations.

- Knowledge models.

A knowledge model is a collection of facts about a kind of thing that expresses generally applicable knowledge about the kind of thing. Such facts are typically expressed by <can be ...> or <can have ...> relations.

- Requirements models.

A requirements model is a collection of facts that expresses (general) requirements for such a kind of thing in the context of the party that expresses the requirements. Such requirements are typically expressed by <shall be ...> or <shall have ...> relations. For example, a government may issue a requirements model for pressure vessels to which all pressure vessels shall comply, provided they are under the jurisdiction of that government.

- Manufacturer's models.

A manufacturer's model is not a collection of facts, but it is specialized subtype of physical object that is defined by a definition model. The relation between a manufacturer's model and its supertype kind of physical object is an <is a model of> relation, which is a special kind of qualification relation. It relates two kinds of physical objects, whereby the subtype is a (manufacturer's) model of a kind of physical object.

The kind of expression <is a model of> (or reverse: <is the nature of model>) specifies that manufacturer's models of physical objects are models (qualifications) of more generic classes.

For example: a type of car, such as a 'Volvo S40' is a model of the generic class 'car'. This is expressed in part 11 as follows:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
222	Volvo S40	is a model of	car

14.27 Criterion to define a new class or specialization

The definition of any new kind of thing shall include a specialization relation with its direct super type. Furthermore, a subtype kind of thing is defined by its distinguisher(s), being the properties, composition and possibly other facts about the kind that are by definition the case for the defined kind and that distinguish the kind from its super type and from its neighboring subtypes.

In addition to the facts that are by definition the case, things of a kind may have a large number of possible (optional) facts, such as components and properties. It is only valuable to define other subtypes if there are generally recognized or standardized things of a type that have one or more qualitative properties in common.

Standardization organizations and manufacturers will define such subtypes as part of their effort to reduce the variation of product types to a suitable minimum. Nevertheless those subtypes may go to a high degree of detail. For example, details define the dimensions of H-profiles, nuts and bolts including their standardized materials of construction, etc.

14.28 Connections and ports

Connections can be specified to various levels of detail. The simplest level is expressed by a simple binary connection relation. For example a connection between a valve V-1 and a pipe P-1 is then modeled as follows:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
230	V-1	is connected to	P-1

The above fact does not state anything about the place where they are connected, nor about the way of connecting, connectors, etc. If it is required to model a connection in more detail, then it may be that the connected objects should be decomposed. For example, the above connection may be a flanged connection, so that both connected objects have a connector, being a flanged end. When we model that the connectors are connected, whereas the connectors are part of the connected objects, then it can be stated that:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
231	Upstream flanged end of V-1	is classified as a	flanged end

232	Upstream flanged end of V-1	is a part of	V-1
233	Flanged end-1 of P-1	is classified as a	flanged end
234	Flanged end-1 of P-1	is a part of	P-1
235	Upstream flanged end of V-1	is connected to	Flanged end-1 of P-1

Connectors are often modeled as separate objects that are involved in a connection. They can be used as inlet end/or as outlet. An inlet or outlet is sometimes generalized to a port or equipment port, especially as long as the kind of connector is not yet determined.

An equipment port is defined as a part of a piece of equipment at its boundary, through which material, energy, force or signals are intended to be transferred. Each equipment port may be one of the following types: an energy port, a mechanical port (an object for the transfer of force), a material port or a signal port. It can be an inlet or an outlet or both (inlet/outlet). A port of a (functional) physical object is carrier of an interaction with another port of the same kind, possessed by another (functional) physical object.

The following model illustrates the use of ports (as a role of connectors). There are two connectors defined, which are physical objects that are initially classified by their roles as energy ports, without classifying them by their physical nature. In a later stage they are classified by their physical nature as terminal box. The example also illustrates that a cable is initially classified only as a power cable, whereas it is later also classified by a particular subtype of power cable.

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
241	P-101	is classified as a	pump
242	P-101	has as part	Connector of P-101
243	Connector of P-101	is classified (by role) as a	energy port
244	Connector of P-101	is carrier of	Energy supply to P-101
245	Energy supply to P-101	is classified as a	energy interaction
246	Starter for P-101	is classified as a	switching device
247	Starter for P-101	has as part	Connector of starter P-101
248	Connector of starter P-101	is carrier of	Energy supply to P-101
249	Connector of starter P-101	is classified (by role) as	energy port

		a	
250	Energy supply to P-101	is classified as a	electric current
251	Energy supply to P-101	is carried by	Cable P-101-1
252	Cable P-101-1	is classified as a	power cable

Whereas (in a later stage) the type of cable can be reclassified (reclassified or double classified) by a subtype or manufacturer's model of cable, for example as follows:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
253	Cable P-101-1	is classified as a	YmVkas

Whereas (in a later stage) it can be specified how the port is realized as follows:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
254	Connector of P-101	is classified as a	terminal box

Note that the objects which name begins with an upper case character are individual things. Terms in lower case denote kinds of things.

14.29 Collection and assembly

Collection

A collection is a group of items that have no fixed position relative to each other and have no specific role towards the whole forms a collection.

The relation type used to describe the relation between the element and the collection is: <is an element of>. The inverse relation type is called: <is a collection of individuals for>.

Note: A collection typically can have as properties: number of items, (total) mass, (total) price, but a collection does not have an property such as a length of the collection.

Assembly

An assembly is a number of items that have a fixed position relative to each other and forms an integral part of one whole.

The relation type used to describe the relation between the part and the assembly is: <is a part of>. The inverse relation type is called: <is an assembly of> with as synonym <has as part>.

For example, a collection of individual things is defined as follows:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
261	collection of bolts B	is classified as a	collection of items
262	bolt-1	is an element of	collection of bolts B
263	bolt-2	is an element of	collection of bolts B

An assembly of individual things is defined as follows:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
1	wheel-1	is a part of	my car
2	motor-1	is a part of	my car

14.30 Parts of more than one whole

A single physical object may form a part of more than one higher assembly physical objects through multiple composition relations. In such a case composition relations are made between one (component) physical object and various larger assembly physical objects.

For example: a control valve can be part of a control loop and can also be part of a piping system. This knowledge is specified in general as follows:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
1	control valve	can be part of a	control loop
2	control valve	can be part of a	piping system

For a particular individual control valve this may be for example:

UID of fact	Name of left hand object	Name of kind of relation	Name of right hand object
1	CV-101	is a part of	Loop-1
2	CV-101	is a part of	P-1200-13-2

14.31 Inverse relations

An opinion about a fact shall be expressed by a relation that is classified by a relation type, represented by its UID of fact. A relation type shall be indicated also by a phrase that is registered as a synonym of the name of the relation type or by a phrase that is registered as an inverse phrase, dependent of which thing has a role as first role player or second role player. Only one of the two phrases is required.

The phrase that is registered as synonym in the TOPini section of the taxonomic dictionary is preferred.

14.32 Cardinality constraints

‘Shall be...’ relations shall have cardinality constraints that indicate whether the inverse relation is nevertheless optional or not.

14.33 Homonyms

A term this is a homonym, which means that the same term denoted different concepts, shall be related to the denoted concepts by naming relations that are defined to be valid in different language communities.

14.34 Kinds versus characteristics

Description of question:

Should we create a subtype kind of physical object (kind of object) when there is an alternative to define a 'kind of characteristic' for such a kind of object which can define the same information.

Solution:

When such a subtype is well known in the discipline, then you should do both. The subtype shall be defined by a specialization relation and the subtype should be defined as having by definition such a characteristic (value). The two ways of modeling are complementary and not each other's opposite, nor does one exclude the other. Subtype kinds of physical objects should only be defined when it is widely recognized that object of those kinds have common characteristics for all members of such a subtype, so that all the members of the subtype have by definition such a characteristic (value).

The fact that characteristics of objects can be used to characterize them should not be of influence to the decision whether or not to define a kind of object (class).

As a general guideline it is recommended:

- To distinguish clearly between kinds of roles (sometimes called ‘functional kinds’) and kinds of physical objects. Kinds of roles don't define common mechanical aspects of the items that play the role. For example, 'driven end bearing' is a role, because the same bearing can be used as a non-driven end bearing.
- To define only those kinds of roles (functional kinds) that are already commonly used in the engineering practice.
For example a 'driven end bearing' is not different from a 'back end bearing'. Nevertheless it may be useful to classify bearings according to their location, e.g. in order to capture differences in maintenance requirements. So the concept ‘driven end bearing’ is a proper kind of role, but not a proper kind of physical object.
- To define all kinds of characteristics for kinds of objects that are required to characterize an objects in that class. It is accepted that this may lead to duplicate information, which shall be kept consistent (which can be automated).
For example, the concept 'single stage pump' is a valid class. It implies that the concept has a characteristic called 'number of stages' and that for the concept ‘single stage pump’ the value of 'number of stages' = 1.

14.35 Characteristics required for activities

Description:

Should we record characteristics of objects that are required for activities.

Solution:

Yes, all kinds of characteristics that are recognized as relevant for kinds of objects are valuable to record. Within ‘views’ it is possible to identify the characteristics that are relevant within the context of such a view. The question which kinds of characteristics are relevant in which context of activity is complicated. In many cases the characteristic appears to be possessed by an object that is involved in the activity and not by the activity itself. A recommended strategy is to define characteristics of activities only

for explicitly defined qualified activities, such as "exchange scenario's". An example of such an exchange scenario is the activity: *'order an item'*.

14.36 Requirements for detailed kinds of Systems

Description:

For the modeling of facilities, a lot of 'kinds of systems' can be identified. Is there a need to identify and define all kinds of systems for inclusion in the dictionary?

Solution:

The strategy is to define only kinds of systems when they appear to be used and valuable in practice, especially when knowledge or requirements are available that are specific for such a kind of system. The amount of facts (knowledge) about kinds of systems will grow especially their composition and intended role.

Note that such knowledge and requirements is typically not included in the public domain formal dictionary, but may be included in proprietary systems and standards.

Definition of subtypes of systems become more valuable when those subtypes have defined standard configurations, such as the kind of system "API 614 compliant lubrication oil system". Manufacturers may define further detailed standard product types by defining subsystems that are other assemblies of the same kinds of components with the same set of kinds of characteristics as the higher level ones, but different values.

For example, a 'methanol injection system' is a specialization of 'chemical injection system'.

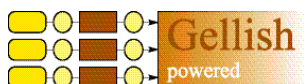
The concept 'chemical injection system' is a suitable concept for a wide variety of chemical injection systems, irrespective of the substance that is injected (methanol or other). Often it is sufficient to express the fact that a specific system, say S-1 'is classified as a' chemical injection system (with the description:) that is intended to inject methanol (or: that is a methanol injection system).

The 'facilities and systems' peer group will decide which kinds of systems will be standardized.

15 Certification

Software that is able to import and/or export and operate on data sets with information models that are expressed in a formal language of the Gellish family of formal languages can become certified by an authorized certifier. It is then allowed to carry the “Gellish Powered” certificate. The “Gellish Powered” certificate can also be obtained for information models themselves. The certificate is protected by the Gellish Registered Trademark.

Organizations may be accredited by Gellish.net to certify software as being able to import and/or export data files compliant with the Gellish.net defined formal languages and their taxonomic dictionaries and to certify formal language domain taxonomic dictionaries. Compliant software or data sets can be recognized by the logo ‘Gellish powered’ in the following layout:



16 Managing the formal language definition

16.1 Rules for acceptance of proposals for extensions

Gellish formal languages are regularly improved and extended. Any person or organization can contribute by issuing proposals according to the procedure below.

Because the definition of the formal languages is maintained in electronic form it is recommended apply the latest electronic version as published on the Gellish.net website. Furthermore it is recommended to minimize private local modifications by proposing enhancements for the centrally maintained language definition and its dictionary. This will allow compatibility with new issued versions of the taxonomic dictionary .

The following rules for management of the definition of a formal language apply:

1. Use of Gellish in commercial systems is at everybody's own *responsibility and risk*. It should be noticed that the dictionary is subject to regular updates.
2. *Proposals* for additions, deletions or other enhancements shall be provided as follows:
 - Use a Gellish Expression Table.
 - Proposals can be issued via a focal point or directly to the language manager. A proposal shall satisfy the quality criteria as described in this guide before being considered by the peer groups.
 - All changes shall be introduced via a new row in the table, either by insertion or by copying an existing row.
 - Fill in the fields in the new row as it is or should be in your opinion or preferably as the common opinion of peer group members.

Filling in the following fields is obligatory:

- Fact id, Unique id = *blank* for new line or *duplicate of id* of the line that is proposed to be modified.
 - Status = 'proposed', or 'propdel', which means: 'proposed to be deleted'.
 - Originator of proposal = *your name*.
 - Date of start of life = *current date* for a new line.
 - Date of latest change = *current date* for new or for modification of existing lines.
 - Source / reference = *source of proposal* (*in case of an authoritative reference*).
 - Remarks = option to argue why the proposal is issued.
 - Validity context (optional) = The name of a context in which the fact is applicable, 'required' or 'relevant'.
- *Select only your proposals* from the table, via sorting on your name and date of latest change; copy your proposals into a separate table and return that table (as a CSV or XLS file) to the language manager.
4. The *language manager* may consult subject area experts, formal standards and dictionaries to *review* the proposals and provide technical advice on their acceptance or rejection.
 5. The *language manager* has the authority to accept proposals or to reject proposals for inclusion in the taxonomic dictionary.
 6. The *language manager* or his delegates are the only authorized persons to change the status of the proposals into *accepted, replaced, deleted, inherited, ignore or issue*.
The quality and completeness of the taxonomic dictionary is the result of volunteered co-operation of various parties. However no one can be formally held accountable for the quality of the taxonomic dictionary content.
No compliance to the taxonomic dictionary can be claimed, until its formal publication as a standard and until official tests are performed as part of the test procedures.
 7. The *language manager* will issue regular *updates* of the definition of the formal languages.

16.2 Return proposals to the language manager

With reference to the rules of taxonomic dictionary management as described in Section 7, a spreadsheet which contains the proposals in the prescribed format should be returned to the language manager. This can be arranged by sending a file via email.

The e-mail address for sending attached files to the language manager is

info@gellish.net

16.3 Version management of Gellish languages

The language manager will ensure that regularly back-up copies are made and that consistent releases of the formal language variants will be published regularly on the Internet.

A new version will consist of the following content:

One or more files, each with a particular release date.

Each file will consist of a first sheet with an Expression Table that has in its first line a date of the latest change within the file.

Each line in an Expression Table has a date of latest change. From that date it can be retrieved which lines were changed since a previous version.

There may be a sheet called 'Deleted' with an Expression Table that contains lines that are removed from the first Expression Table. Those lines may have the status 'replaced', together with a UID of a successor of fact. This indicates that the replaced line is succeeded by the line indicated by the UID of the successor.

17 References

1. Formalized natural languages - Definition and application of universal information modeling languages, Andries van Renssen, Gellish.net (2014); 2nd Edition of Gellish, A Generic Extensible Ontological Language (PhD), Delft University Press, 2005.
<http://www.lulu.com/shop/dr-ir-andries-van-renssen/formalized-natural-languages/paperback/product-21904436.html>.
2. Semantic information modeling in formalized languages, Andries van Renssen, Gellish.net (2014). <http://www.lulu.com/shop/dr-ir-andries-van-renssen/semantic-modeling-in-formal-english/paperback/product-21538016.html>.
3. Formal English Taxonomic Dictionary and Upper Ontology,
<http://www.gellish.net/index.php/shop/category/17-formal-english-dictionaries-taxonomies.html>.
4. Gellish syntax and contextual facts - Definition of universal semantic databases and data exchange messages (also included in Ref. 2),
<http://www.gellish.net/index.php/downloads/category/1-free-introduction.html>
5. Example of a road network,
<http://www.gellish.net/index.php/downloads/file/3-gellish-english-example-road-v4-doc.html>.
6. ISO 1087-1:2000, Terminology work - Vocabulary - part 1: Theory and application.
7. IDEF0: <http://en.wikipedia.org/wiki/IDEF0> and www.idef.com.
8. ISO 10303-42, Product representation and exchange – Integrated generic resource: Geometric and topological representation
9. ISO 10303-46, Product representation and exchange – Integrated generic resource: Visual presentation
10. ISO 10303-221, Application protocol: Functional data and their schematic representation for process plants
11. ISO 15926-2, 3, 4, 11: Life cycle data for products and processes.
12. ISO 16354, Guidelines for knowledge libraries and object libraries.
13. ISO 16739 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries

14. ISO 12006-3 (IFD) Organization of information about construction works -- Part 3: Framework for object-oriented information.
15. https://repository.jisc.ac.uk/6438/2/Den_Haag_Persistent_Object_Identifier_%E2%80%93_Linked_Open_Data.pdf
16. Digital Object Identifier (DOI).
https://en.wikipedia.org/wiki/Digital_object_identifier#cite_note-iso-1

18 Glossary

AIM

Asset Information Management, being a synonym of Technical Information Management. This includes the management of technical data and documents about a facility and its operation during its complete lifecycle.

Data set

A data set is a collection of data values (character strings or numbers) that represent *facts* (expressions of “what is the case”) and that is arranged in a formal structure. Typically arranged in a tabular structure. Ideally expressed in a *formal language*. For example arranged in one or more database or Excel spreadsheet tables structure.

Document

A document is information in a textual and/or graphical (or even audible) form that contains representations of *facts*. Usually, but not necessarily, in natural language form.

Electronic smart dictionary

An electronic smart dictionary is a dictionary that is computer interpretable (expressed in a formal language) and includes relations between the defined concepts, especially subtype-supertype relations. This enables to link additional knowledge to the concepts in the dictionary.

For example, a smart electronic dictionary enables a computer to ‘know’ that a line shaft pump is a subtype of a centrifugal pump and thus also shall have an impeller as one of its parts.

Fact

A fact is something that is the case. A fact is typically expressed as a relation between two objects. Typically facts are expressed in data exchange files or database tables.

For example: K-1201 <is a part of> U-1200.

Formal language

A formal language is a method to express *facts* in an unambiguous formally defined way, using a standardized electronic ‘smart’ dictionary and standardized relation types

(or standard ‘data model’) to create expressions. Expressions in a formal language are computer interpretable.

Information

Information includes *data sets* and *documents* that convey meaning about a subject. The same information may be expressed in various ways on different *information carriers*.

Information carrier

An information carrier is a physical object that carries information. For example, several electronic files in various formats and multiple paper copies are information carriers that may carry the same information.

