TECHNICAL SPECIFICATION

ISO/TS 15926-11

Second edition
2023-04

Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities —

Part 11: Simplified industrial usage of reference data based on RDFS methodology

Systèmes d'automatisation industrielle et intégration — Intégration de données de cycle de vie pour les industries de "process", y compris les usines de production de pétrole et de gaz —

Partie 11: RDFS méthodologie pour un usage industriel simplifié des données de référence

Reference number
ISO/TS 15926-11:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 4, *Industrial data.*

This second edition cancels and replaces the first edition (ISO/TS 15926-11:2015), which has been technically revised.

The main changes are as follows:

— as a basis for the initial set of relationships a set of use cases in the context of systems engineering, ISO/IEC/IEEE 15288 is used rather than a set of formal information models derived from systems engineering;

— the document has been aligned with ISO 15926-2;

— a resource description framework (RDF) statement has been added as reification method additional to the RDF named graph;

— a method has been added for applying configuration management using this document;

— a method has been added to create a data exchange file between involved parties in a project.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The ISO 15926 series describes the representation of process industries facility life-cycle information. This representation is specified by a generic, conceptual data model that is suitable as the basis for implementation in a shared database or data warehouse. Another application is to create handover files containing explicit, unambiguous life-cycle data which complies with a commonly shared data model and reference data library (RDL).

The data model of the ISO 15926 series is designed to be used in conjunction with reference data, i.e. standard instances that represent information common to a number of users, production facilities, or both. The support for a specific life-cycle activity depends on the use of appropriate reference data in conjunction with the data model.

This document focuses on a simplified implementation of the afore mentioned data model in the context of systems engineering data in the area of the process industry, including the oil, gas, power (fossil, nuclear and renewable energy), but can also be used in the area of manufacturing and aerospace industries. It is intended for developers of configuration management processes and systems in general. This document offers a dual use methodology. Alternatives include a Common Data Environment (CDE) or data handover environment using design tools that create project and systems engineering data.

Systems engineering deals with the development of requirements, their allocation to the items that are being designed and developed when these items are considered as part of a system. This document concentrates on the system as a whole, as distinct from the parts considered individually. It requires verification that the design is properly built and integrated and how well the system meets its initial by stakeholders stated goals.

This document provides the capability to express a product model and or systems engineering data with RDF triples which can be reified by means of an RDF statement or RDF named graphs and a standardized set of natural language relationships. The results can be used for an exchange or handover file that can be shared and relatively easily understood in industry.

There is an industry need for this document:

— The triple relationships are easy to understand by an engineer so that an engineer can understand the product model intuitively. This has been proven by the Program Integral Collaboration for the Maritime Industry which developed an RDF based implementation for standardized exchange of product data. This project was completed in 2013 by a group of Dutch shipbuilding companies, its contractors and its suppliers.

— The standard data sheets from, e.g. the American Petroleum Institute (API), NORSOK, used in industry for pumps, compressors, instruments, etc. can be supported by an RDF product model enabling industry to continue to work with their specific data sheets and yet exchange the data in a standardized way according to this document.

— It is used in some projects, e.g. the Pallas nuclear facility project in the Netherlands, in which based on the ISO 19650 series, a CDE is built upon this document, including facility data handover to the client.

— This document can be used as a front-end engineering layer for the template methodology used by ISO/TS 15926-7 and ISO/TS 15926-8. This makes the content of those projects easier to access by engineers.

— This document can be used in combination with reference data libraries from various sources. In process industries ISO/TS 15926-4 would typically be used as RDL to which missing reference data would be added.

— An engineering, procurement and construction (EPC) contractor has used this document in various tunnel projects for information modelling in systems engineering which was required by the Dutch authority regulations. With this document enriched by the knowledge from ISO/IEC/IEEE 15288, this became possible. They also built a performance measuring system for operational data in

tunnel installations where the methodology of this document is used to justify the performance to the ministry of transportation in the Netherlands.

The ISO 15926 series is organized as a series of parts, each published separately. The structure of the ISO 15926 series is described in ISO 15926-1.

NOTE 1 For examples and representing the ontology of this document, TriG is used as serialisation method in this document.

NOTE 2 RDFS doesn't include reasoning based on OWL and or SHACL. If one wishes this kind of functionality, one can make use of SPARQL, which is used in this document for validation purposes. It is broadly implementable and relatively simple. That is why references in this document only make use of RDFS.

Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities —

Part 11: Simplified industrial usage of reference data based on RDFS methodology

1 Scope

This document enables a flexible creation of product knowledge models and data that supports systems engineering processes. The payload or design data can be exchanged across organizations or with the supply chain by combining resource description framework (RDF) triples, reference data dictionaries and a standardized set of relationships.

This document is appropriate for use with the ISO 15926-series based reference data libraries, and it is applicable to the process industry, including oil, gas and power. However, manufacturing and aerospace industries can also benefit from this document.

The following are within the scope of this document:

— process plants in accordance with ISO 15926-1;

— a methodology with low threshold for using reference data in combination with RDF triples for representing statements as defined in the ISO 15926 series;

— an initial set of relationships required for process plant life-cycle representation;

— a method to implement configuration management to trace back additions, changes and deletions in product and project data and enabling baselining;

— data sharing, integration, exchange, and hand-over between computer systems.

The following are outside the scope of this document:

— serialisation methods;

— definition of reference data libraries;

— the syntax and format of implementations of either product data models or instance data using this document, or both;

— any specific methods and guidelines other than RDF(S) for implementing ISO 15926-2.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 15926-2, *Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 2: Data model*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1.1
asset
any item, thing or entity that has potential or actual value to an organization

[SOURCE: ISO 55000:2014, 3.2.1, modified — Notes 1 to 3 to entry were deleted.]

3.1.2
attribute
quality or feature of something

3.1.3
blank node
BN
node in a resource description framework (RDF) graph representing a resource for which a literal is not given

Note 1 to entry: The resource represented by a blank node is also called an anonymous resource. According to the RDF standard W3C RDF-1.1,[16] a blank node can only be used as subject or object of an RDF triple.

3.1.4
constraint
thing that limits something, or limits one's freedom to do something

EXAMPLE A design constraint, legal constraint or implementation constraint.

3.1.5
data
representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers

[SOURCE: ISO 10303-1:2021, 3.1.29]

3.1.6
designing
activity of developing by creating, planning, calculation, or laying out for a predetermined purpose

3.1.7
domain
set of all possible independent values the relationship can take

Note 1 to entry: It is the collection of all possible inputs (the "left-hand" of a relationship).

3.1.8
engineering
activity of designing or producing by methods of technical sciences

Note 1 to entry: During the activity, the properties of matter and the sources of energy in nature are made useful for human beings in structures, machines, and products.

3.1.9
engineered item
type of equipment created by specific and unique engineering and specifications, developed with a supplier or manufacturer

Note 1 to entry: An engineered item has a unique identifier (UID).

3.1.10
engineering data
data that represents the design and or engineering of a system or a system element

Note 1 to entry: The scope can be limited to a specific discipline (electrical, mechanical, civil), however after integrating all engineering data obtained from engineering tools, the result should represent the integrated design in a consistent way, which implies appropriate quality and harmonization of the data, obtained from the various tools.

3.1.11
enterprise
any private or public business or company

3.1.12
entity
something that exists separately from other things and has its own identity

3.1.13
enumeration
complete, ordered listing of all the items in a collection

Note 1 to entry: The term is commonly used in mathematics and computer science to refer to a listing of all of the elements of a set.

3.1.14
facility
permanent, semi-permanent, or temporary commercial or industrial property built, established, or installed for the performance of one or more specific activities or functions

EXAMPLE A building, plant, or structure.

3.1.15
functional requirement
requirement defining either the functional capabilities or behaviour, or both, that a product shall have

3.1.16
interoperability
ability of effective interaction between systems based on the exchange of information

Note 1 to entry: Systems can be computerized systems or enterprises.

3.1.17
information
facts, concepts, or instructions

[SOURCE: ISO 10303-1:2021, 3.1.41]

3.1.18
formal syntax
specification of the valid sentences of a formal language using a formal grammar

EXAMPLE An extensible markup language (XML) document type definition (DTD) is a formal syntax.

Note 1 to entry: A formal language is computer-interpretable.

[SOURCE: ISO 8000-2:2022, 3.9.1, modified — Notes 2 and 3 to entry were deleted. EXAMPLES 2 and 3 were deleted.]

3.1.19
model
simplified description, especially a mathematical one, of a system or process, to assist calculations and predictions

3.1.20
model based systems engineering
MBSE
formalized application of modelling systems engineering information

Note 1 to entry: The application of modelling supports system requirements, design, analysis, verification and validation (V&V) activities with their mutual relationships, beginning in the conceptual design phase and continuing throughout development and later life-cycle phases.

3.1.21
resource description framework graph
RDF graph
graph structure formed by a set of RDF triples

[SOURCE: W3C Recommendation 2014][16]

3.1.22
resource description framework triple
RDF triple
atomic data entity in the resource description framework (RDF) data model

Note 1 to entry: An RDF-triple represents a relationship between the objects or data that it links.

Note 2 to entry: A triple comprises at least:

— an object called “subject”;

— a predicate (also called property) that denotes a relationship between a subject and an object;

— an object or data called “object”.

[SOURCE: W3C Recommendation 2014][16]

3.1.23
resource description framework statement
RDF statement
statement stating facts, relationships and data by linking resources of different kinds

3.1.24
reference data
facility life-cycle data that represent information about classes or individual things which are common to many facilities or of interest to many users

[SOURCE: ISO 15926-1:2004, 3.1.18, modified — "process plants" replaced by "facilities".]

3.1.25
range of relationship
set of all possible dependent values the relationship can produce from the domain values

Note 1 to entry: The domain values are in the “right-hand” of a relationship.

3.1.26
semantics
study of meaning, concerned with the relationship between signifiers that people use when interacting with the world, and the things in that world that these signifiers denote

EXAMPLE 1 The signifiers can be words, phrases, signs, and symbols.

EXAMPLE 2 Things denoted by signifiers can be entities, concepts, ideas.

Note 1 to entry: The goal of semantics is the creation of a common understanding of the meaning of things, helping people understand each other despite different experiences or points of view.

3.1.27
semantic encoding
concept encoding
technique of replacing natural language terms in a message with identifiers that reference data dictionary entries

EXAMPLE ISO 8000-110 specifies how semantic encoding supports the exchange of master data that is characteristic data.

Note 1 to entry: By applying semantic encoding to data, an organization creates a basis for portable data by ensuring the semantics of the data are explicit.

Note 2 to entry: Semantic encoding is necessary to create characteristic data, where the replaced natural language terms are properties (for each of which the data set includes a corresponding value).

[SOURCE: ISO 8000-2:2022, 3.9.2]

3.1.28
semantic data modelling
development of descriptions and representations of data in such a way that the latter's meaning is explicit, accurate, and commonly understood by both humans and computer systems

Note 1 to entry: In semantic data modelling, all concepts used to model a system are explicitly defined by ontologies, capturing the "meaning" of data with all its inherent relationships in a single graph.

3.1.29
statement
information that is regarded as indivisible and which is the case, independent of natural language

Note 1 to entry: Adapted from ISO/TS 15926-6:2013.

Note 2 to entry: A statement can be recorded as an instance of the entity relationship in ISO 15926-2. A set of one or more statements can be recorded in shorthand form as a single item as an instance of a template, as defined in ISO/TS 15926-7.

3.1.30
signature statement
statement that states that the performer of an activity applied including the sign-off date on an individual named graph

3.1.31
stakeholder
person or company that is involved in a particular organization, project or system

EXAMPLE Involvement of a person or company can concern safety, environment or other.

Note 1 to entry: The stakeholder's involvement is especially because they have invested money in it or have a functional responsibility.

3.1.32
stakeholder requirement
requirement defining how a stakeholder wants to interact with an intended solution for a requirement

3.1.33
system element
part of a system which can be inanimate physical objects (not alive) and animate physical objects (alive)

Note 1 to entry: A system often is called a "set of elements".

3.1.34
systems engineering
interdisciplinary approach governing the total technical and managerial effort required to transform a set of stakeholder needs, expectations, and constraints into a solution

EXAMPLE A solution can be a system.

Note 1 to entry: The approach supports a solution throughout its life.

3.1.35
system requirement
result of the transformation of the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user

3.1.36
tagged item
equipment and major electrical and instrumentation item that has a specific tag number and which is treated individually for tracking and tracing purposes

Note 1 to entry: Bulk materials are excluded from this definition which normally are identified batches.

3.1.37
technical solution
solution to a problem that is dealt with so that the difficulty is removed by applying an appropriate technology or design principle

3.1.38
payload data
actual data in a data packet or data container minus all headers attached for transport and minus all descriptive meta-data

Note 1 to entry: In a network packet, headers are appended to the payload for transport and then discarded at their destination.

3.1.39
V-model
graphical representation of a system's development life cycle

Note 1 to entry: It is used to produce rigorous development life cycle models and project management models.

3.1.40
validation
proof that the system accomplishes or, toned down, can accomplish its purpose

Note 1 to entry: It is usually much more difficult and much more important to validate a system than to verify it, and give an answer to the question: 'Have we made the correct product?'

3.1.41
verification
proof of compliance with the specification

Note 1 to entry: Compliance may be determined by an objective test, analysis, demonstration, inspection, etc. for each requirement or set of requirements. It answers the question: 'Have we made the product correctly?'

Note 2 to entry: In general, verification is seen as the process of checking the compliance with a requirement.

3.1.42
3D model
representation of a physical body using a collection of interconnected points in a three-dimensional space

Note 1 to entry: Interconnected points can form triangles, lines, curved surfaces.

3.2 Abbreviated terms

API	American Petroleum Institute
BN	Blank Node
GUID	Global Unique Identifier
GBS	Geographical Breakdown Structure
FMEA	Failure Mode and Effect Analysis
FMECA	Failure Mode, Effect and Criticality Analysis
IDM	Information Delivery Manual
ITB	Information to Bid
ITT	Information to Tender
MBSE	Model Based Systems Engineering
OWL	Web Ontology Language
P&ID	Piping and Instrumentation Diagram
RDL	Reference Data Library
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SHACL	Shapes Constraint Language
SPARQL	Protocol and RDF Query Language
SBS	System Breakdown Structure
SE	Systems Engineering
SKOS	Simple Knowledge Organization System
URI	Uniform Resource Identifier
V&V	Verification and Validation
WBS	Work Breakdown Structure
W3C	World Wide Web Consortium

4 Purpose, objectives and principles

4.1 General

Systems engineering is concerned with identifying and developing requirements of a system of interest and their assignment to the items designed and built as part of the system of interest. The emphasis of systems engineering is, on the system as a whole, as distinct from the parts considered individually. It requires verification that the design is properly built and integrated and how well the system meets its intended goals (by validation). Model Based Systems Engineering (MBSE) is a methodology of systems engineering that focuses on creating and exploiting domain models as the primary means of managing

conformance to requirements. Digitization of systems engineering processes is a precondition for MBSE and the latter a foundation for the creation of digital twins.

This document provides a semantic data modelling methodology of engineering data that is created and or used in systems engineering processes and that can be relatively "easily" understood by engineers and that is flexible in terms of tailoring the methodology for a specific domain or project. The provided modelling methodology is based on the existing parts ISO 15926-2 and an RDL such as ISO/TS 15926-4. To achieve this, the triple concept of the W3C RDF standard is adopted and augmented with a set of relationships further called the "initial set of relationships" (Annex A) which easily can be expanded with relationship relevant in the context of a specific project. With respect to the presented semantic modelling methodology, this document makes use of EN 17632.

A main reason for developing this methodology can be found in the fact that product specialists and systems engineers, especially within small and medium enterprises (SMEs), who in general have limited skills in the area of information modelling and related techniques, should be supported in their product and or systems engineering knowledge modelling activities by a simple to use methodology, close to natural languages. Using this methodology will lead to models which describe specific systems engineering information by semantic encoding of this information and that in potential are upgradable to fully ISO 15926-2 compliant models, i.e. this methodology provides a bridge to the much more complex ISO 15926-2 world and provides a low entry threshold to the ISO 15926 series. Also, in design development work, humans like to work with simple table-based structures rather than relatively complicated schemes, and this should be respected as far as feasible.

Engineering data in the context of this document cover not only the output and or input of engineering processes, but also the engineering processes themselves. In this way, this document enables the integration and exchange of product data together with data, relevant in the context of systems engineering processes. Other parts of the ISO 15926 series do not sufficiently cover the features of systems engineering when exchanging information over the life cycle of a plant or when designing new products. In line with this, this document also offers a way to industry partners to set up and exchange their product model using a low-level modelling methodology based on statements which can be represented and exchanged in a table manner. For that purpose, this document provides a normative set of rules that allows engineers to build product and plant life cycle models using statements based on a normative set of relationships and normative plant reference data.

A statement can be used to classify things as "being the case", also called a "fact". Statements can be expressed in languages as relationships between two roles of things (respectively "thing playing role 1" and "thing playing role 2"). This process can be seen as semantic encoding using a formal syntax.

From the point of view of data handover, the design, engineering and construction of a process plant or facility in general is fragmented and based on tools from different vendors and different versions of these tools, even by discipline. Over the life-cycle of a facility, in general, multiple information systems and databases from different vendors are used for different purposes. Most of these systems are not integrated with one another and cannot easily share plant data during different phases of the plant life cycle, such as design, operation, and decommissioning. This results in redundancies in capturing, handling, transferring, maintaining, and preserving facility configuration data. This lack of interoperability stems from the fragmented nature of the construction and building industry, paper-based document control systems, a lack of standardization and inconsistent technology adoption among stakeholders. With the help of the methodology described in this document, any kind of product or engineering information obtained from any tool can be expressed unambiguously and the information can be exchanged based on a managed set of reference data. This document provides a semantic modelling methodology for creating and exchanging engineering data, originating from systems engineering processes for example as described by natural language in the ISO/IEC/IEEE 15288.

4.2 Positioning of this document

4.2.1 Overview

The general process of data exchange in the context of the ISO 15926 series is explained from a practical point of view of conformance testing of the software implementation against the ISO 15926 series,

not only taking into account the technology available today but also the maturity of information and communications technology (ICT) skills in modern engineering environments. This clause covers data exchange in terms of physically exchanging a file between two parties (with different software systems) to exchange explicit, unambiguous asset management information. This category of interoperability approach is defined by the enterprise interoperability framework defined in ISO 11354-1 as a "unified approach".

As an example, the exchange of data during the creation of an asset, between the owner and a supplier, is used in the context of a unified interoperability approach (according to ISO 11354-1). The principle used in this example is shown in Figure 1. The handover from one company to another company is represented by a digital envelope containing the payload data as output of one or more business processes. The receiving company will be able to receive, understand and process the envelope and integrate the payload data within their own ICT environment since the data is defined unambiguously by means of the shared ontology and an RDL.

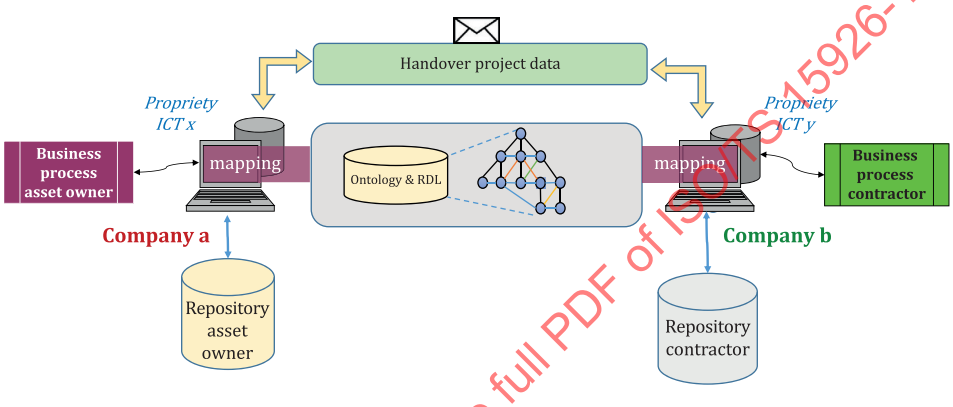


Figure 1 — Principle of data exchange on a commonly shared ontology and RDL

4.2.2 Process steps in the workflow of exchange data

With respect to data exchange activities within a project, an analysis should be carried to determine the subset of data that will be exchanged and which requires a data centric approach as delivered by this document. For that purpose, the activity model as described in ISO 15926-1 and shown in Figure 2 can be used to select one or more handover scenarios between the composing processes within a project.

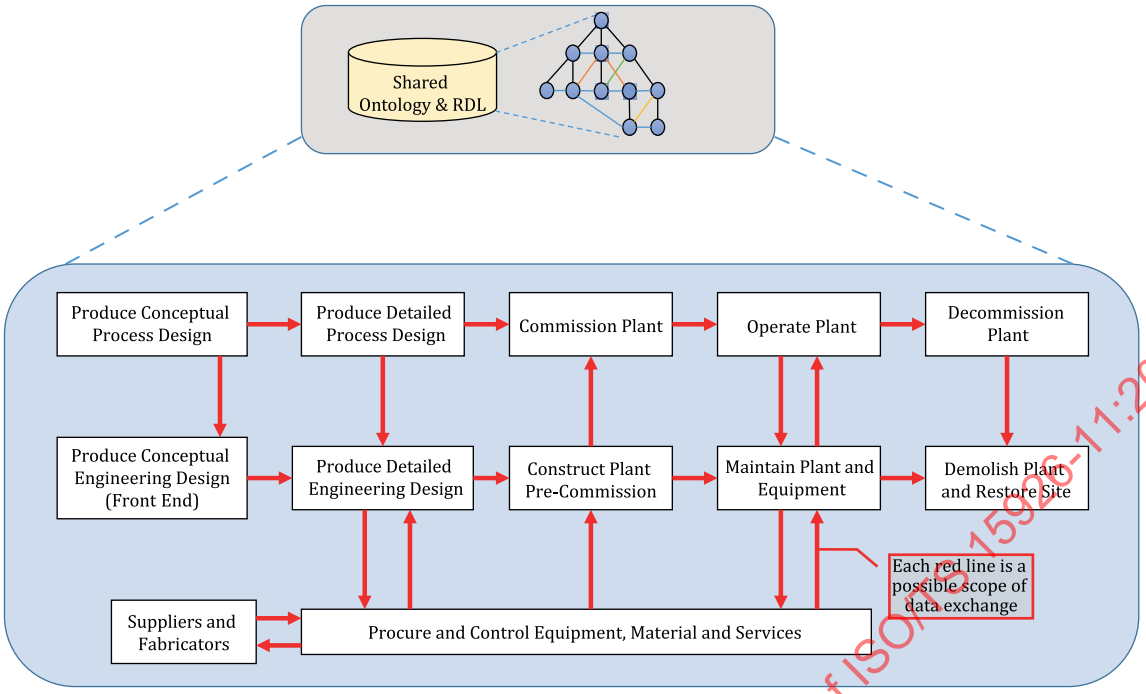


Figure 2 — Activity model as a basis for the selection of role and scope of data exchange

In this document, a systematic approach and methodology based on four software implementation layers as presented in Figure 3 is followed to organise the exchange process. Figure 3 shows the four layers that can be distinguished looking at data exchange in general. The boxes on the four layers as presented on the left side of Figure 3 are defined as normative in ISO 15926-10. The boxes on the right side of Figure 3 show how these layers are implemented in this document:

— role and scope of the data exchange (the red arrows in Figure 2);

— content: definitions of the objects and relationships that can be found in the exchange file, classified according to the shared RDL;

— semantics (meaning) of the data exchanged defined by the part or parts of the ISO 15926 series, agreed on, mainly based on W3C recommendations for RDFS;

— syntax and storage: the method of serialization and syntax used on the data level.

In the context of a specific project, each layer can be specified by means of an information delivery manual (IDM) for that project.

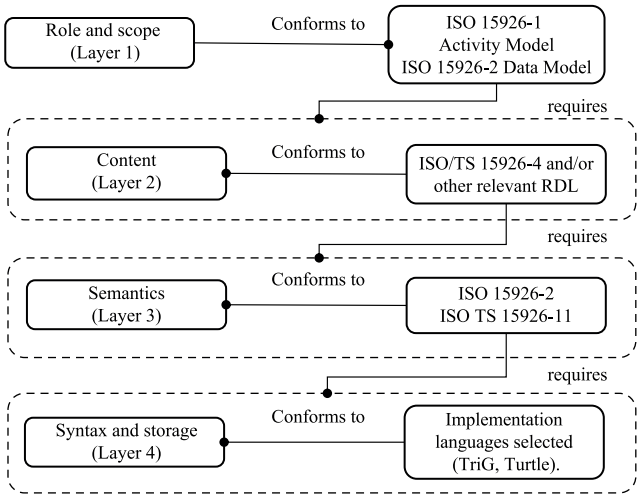


Figure 3 — Layers within the process of data exchange according to this document

4.2.3 Use cases systems engineering

ISO/IEC/IEEE 15288 defines approximately 25 processes, which have been defined to realize a system, starting from the statement of purpose (objective) of a system and a set of top level (stakeholder) requirements ending in operating and maintaining that system.

Based on the generic system life cycle process descriptions given in ISO/IEC/IEEE 15288, a set of uses cases has been derived in preparation for the initial set of relationships defined in this document. This set of relationships, including their prescribed domain and range, can be seen as a reference ontology for systems engineering.

EXAMPLE The information structure of a deliverable from a supplier/contractor can be predefined by the client and specified in an IDM. Agreed information structure can be a subset of the presented reference models in this clause (and eventually be modified).

With respect to the red lines in Figure 2, there are several areas of interest concerning the information that can be exchanged. The following eleven areas of interest, which can be the subject of information exchange, are recognized in this document:

a) Exchange of project requirements

The client has drawn up a set of project requirements for the tendering/bidding of a project. This document will be shared with tenders and suppliers. During the preparation of the technical and commercial proposal, this requirement will be, further, exchanged with candidate suppliers/ manufacturers, or vendors as part of the demand specification. The requirements are provided with additional information by means of one or more statements (e.g. of type rationale, explanation), and references to documents (e.g. standards, legal, specifications).

b) Exchange of technical requirements

The client has drawn up a set of functional, performance technical requirements and constraints for the tendering of a project. Each requirement (as singular, SMART textual specifications) will be identified by a code and assigned to a class of the shared RDL and or instance of an RDL class (e.g. an instance of a kind of system). A requirement is assigned to a specific aspect (e.g. safety, construction, availability, maintainability) and a severity (classification) if there is a non-conformance (e.g. fatal, critical, serious).

c) Exchange work package information as defined in the WBS

Work packages exchanged between the client and supplier have been predefined. Within specific work packages the supplier has defined one or more milestones and one or more activities

to be carried out. Activities have a responsible party and one or more performers (roles and or organizations). The activities are related to a specific life cycle stage of the project (e.g. the preparation or execution). Predefined examples include: Inputs, outputs, resources required to execute the activity, all directives applicable to the activities and applied or required standards.

d) Exchange plant process or logical design information

Process flow diagrams (PFDs) define the exchange from client to supplier. The diagrams depict a process decomposition and definitions of interconnecting streams where functional objects are inserted into these processes. The main characteristics and or capabilities of the processes, streams and functional objects are defined by means of properties, e.g. heat balance or mass balance.

e) Exchange physical design information as defined in the SBS

Information about the configuration of the plant (e.g. its decomposition, SBS) is exchanged between client and supplier. Use is made of an RDL that is shared by all project parties. The client prescribes the use of the RDL. Each object is classified as a thing in the RDL. The physical design is connected to the process design, especially the functional objects from the PFDs, realized by functional physical objects which can be found on the P&IDs. Drawings or diagrams such as plot plans, schematics, or architecture views accompany the SE information package. SBS details are decomposed to expose the functional breakdown structure and the geographical or environmental conditions. Objects are specified by means of properties as defined in the RDL. Each set of properties has an organizational owner. The objects in the SBS are linked with applicable requirements and statements (e.g. design decision statements, rationale, assumption and solution statements). The ITT/ITB describes roles and authoritative responsibilities and assignments of disciplines in charge. Objects in the SBS can refer to any kind of document.

f) Exchange V&V matrix

V&V activities are carried out as part of a project by a specified role and specific person or organisation. As a result of a V&V activity, one or more requirements and one or more things that must be in conformity with these requirements are combined with evidence that these requirements are met. The results are exchanged by supplier with the client to demonstrate that the design meets the set of requirements. Evidence that a requirement is fulfilled can be done by referring to a document (section) or any applicable statement. Also, non-conformities are documented in the V&V matrix and provided as digital data. 30 %, 60 %, 90 %, and the final model review are typical situations where it applies.

g) Exchange interface information

Information about external interfaces between the plant and its environment is exchanged between supplier and the client. Each interface has identification and supports one or more streams or interactions. Also, internal interfaces between systems as part of the plant are identified and managed by means of characteristics, source and target to physical objects and or role of a party (participating role and domain). Interface can be an aggregation of ports of physical objects, via these ports for example, streams and interactions are exchanged with other physical objects.

h) Exchange risk inventory list

The supplier has drawn up a risk inventory list and wants to exchange this with the client. It concerns information such as: which risk is involved, what is the cause of the risk, what is the result if the risk occurs (effect), and who is the owner of the risk and mitigation measures. Risks are characterized by the score of the probability of the risk arising and the score of the consequences of the risk in money. The risk inventory list is exchanged with the client as digital data.

i) Exchange project change management

Changes that arise during the project are registered and managed digitally. Impact of the change on the project including time, costs and quality, is qualified and or quantified. The consequence on impacted systems and related risks are analysed and managed. The change management

information is exchanged with the client as digital data accompanying correspondence with statements and relevant documents.

j) Exchange FMEA/FMECA analysis results

An FMEA activity helps to identify potential failure modes based on experience with similar products and processes or based on common physics of failure logic. Effects analysis refers to studying the consequences of those failures on different system levels of the plant. The FMECA analysis estimates the total availability of the plant. The results of the analysis will be exchanged as digital data to the client in order to support the asset maintenance strategy.

k) Exchange tag and asset register

During the course of the project, the tag and asset register will be developed, which, on a regular base (i.e. at the predefined and agreed milestone), will be exchanged with the client digitally. During detailed design, assets arise as counterpart ("is realised as", and later on time "is installed as") for the tags identified during the basic design. The location is assigned to the assets and a property set (functional and technical properties). Optionally, the function of a tag can be assigned. Each equipment or even device has a unique plate with an asset number. These are recorded in the asset register. The product equivalent is the bill of material, defining a product structure (e.g. parts, assemblies, and installations), identified by part numbers, serial numbers, or persistent IDs.

4.3 Core terms in the context of systems engineering

When analysing the use cases in 4.2.3, one can identify a collection of key terms that together form a thread through these use cases. A representative set of these key terms (not exhaustive) is represented below organized by means of core entities derived from ISO 15926-2 (the root level entities) and second level entities, introduced in Clause 5 of this document as ISO 15926-11 entities (mostly derived from either ISO 15926-2 and ISO/TS 15926-4).

Within the EN 17623 semantic modelling and linking standard the root level elements were partly derived from ISO 15926-2. In Figure 4 the core concepts of EN 17632 are brought in relation to each other by means of some fundamental relationships as defined in EN 17632. Within EN 17632 a distinction is made in objects (subdivided into physical objects and information objects) that exist and activities that happen. There are two main relationships between objects and activities: physical objects perform activities and activities are transformed by physical objects. ‘Time aspects’ must be taken care of by the actual conceptual modelling itself, here by the introduction of dynamic concepts beyond those static objects and activities being states and events. When time is not relevant in case of timeless static aspects or just one ‘snapshot’ of objects in time, these states and events can be ignored (kept implicit).

The second level of concepts will be in the context of this document addressed in later sections.

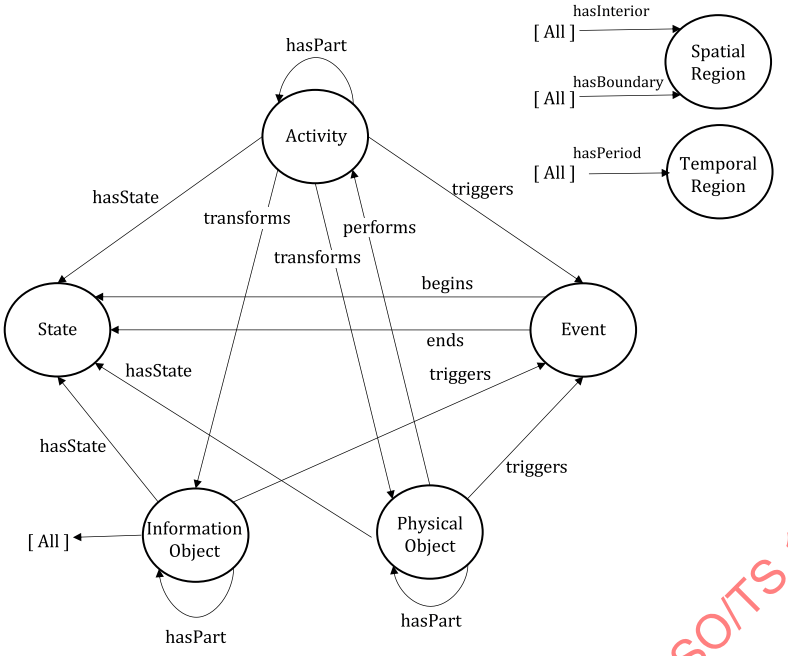


Figure 4 — Fundamental concepts and relationships which support the context and definitions of these concepts

In the context of systems engineering one can divide the life cycle of physical objects into four quadrants from the point of view of functional versus technical (logical versus physical) on the one hand and planned versus realized (imaginary versus real) on the other hand. This is introduced in EN 17632 and represented by Figure 5. Quadrant one, two and three of Figure 5 are logical steps in designing and engineering.

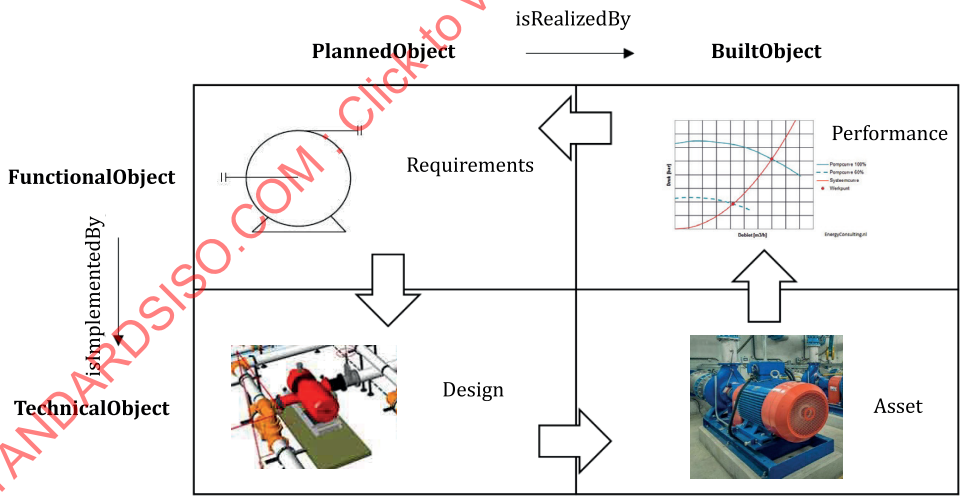


Figure 5 — Fundamental life cycle quadrants supporting systems engineering

4.4 Conformance against requirements in projects

Functional objects (imaginary or real) shall have functional requirements (unstructured/textual and/or structured). Technical objects (imaginary or real) shall have technical requirements also known as 'technical boundary conditions' (unstructured/textual and/or structured). Functional objects (imaginary or real) and technical objects (imaginary or real) can have observations. Observations for

imaginary objects deal with planned, designed, calculated, simulated aspects. Observations for real objects deal with real aspects like measurements, tests, inspections etc.

The evaluation of requirements versus observations leads to levels of conformance. The check between requirements and imaginary observations ("planned performance") is referred to as "verification". The check between (verified) imaginary observations and real observations ("real performance") is referred to as "validation". Insufficient conformance leads to a decision-making process involving variants and preferred solutions based on multi-stakeholder/multi-criteria analysis resulting in a decision about the application of measures/activities being a technical (life-cycle) activity: maintenance (kind of rebuild), renovation (involving redesign) etc. or a functional activity (like the reconsideration of the functional requirements).

A technical (life-cycle) activity leads to a change in the state of a technical object that is being (re) designed and/or (re)build. A functional activity leads to a change in the state of a functional object that is being (re)purposed with the ultimate repurposing being 'no more purpose' also known as demolishment (and potential recycling) of the object.

For the modelling of structured or unstructured requirements this document specifies the constructs as shown in Figure 6.

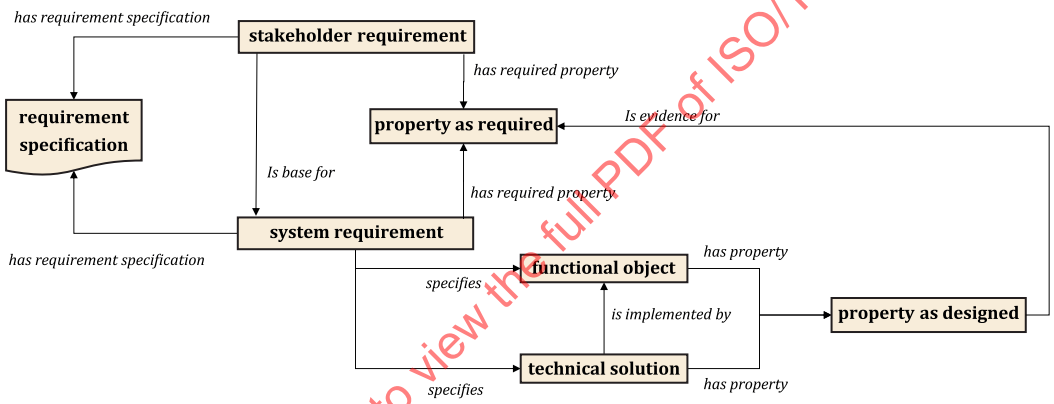


Figure 6 — Modelling requirements by means of textual requirements and or properties

According to systems engineering principles it is important to distinguish the difference between stakeholder requirements and system requirements. Both must be fully traceable to each other. Checking the fulfilment of stakeholder requirements is a validation activity ("Have we made the correct system?") and checking the system requirements is verification ("Have we made the system correctly?"). A requirement can be expressed in natural language or by means of a quantified or qualified property (property as required). Requirements can be on functional level or performance level and in general they specify functional objects. Also, the technical solution level can be specified or constrained by means of (technical) requirements. The designed functional object will possess properties (quantifiable and qualifiable) which can be used as evidence that functional requirements are correctly fulfilled (property as designed). The same is valid for technical solutions, which also possess properties which can be used as evidence that technical requirements and or constrains are fulfilled.

4.5 Breakdown structures

As defined in IEC 81346-1, function, product, and location aspects are necessary and applicable in almost every life cycle phase of an object (e.g. plant, system, equipment). They are, therefore, to be considered as the main aspects and primarily applied for structuring. However, not only the physical object is subject to structuring, also the project itself that produces the object and related business processes (including design, logistics and construction) are subject to structuring in the context of systems engineering. For this reason, this document allows as many breakdown structures as needed (fit for purpose), each related to a relevant aspect of the product and project. In this way the object and the business processes can be integrated, which allows significant optimization over the whole life cycle which is relevant for asset management. In Figure 7 various breakdown structures are shown

which are relevant when specifying, creating, and maintaining a facility or process plant (including their properties).

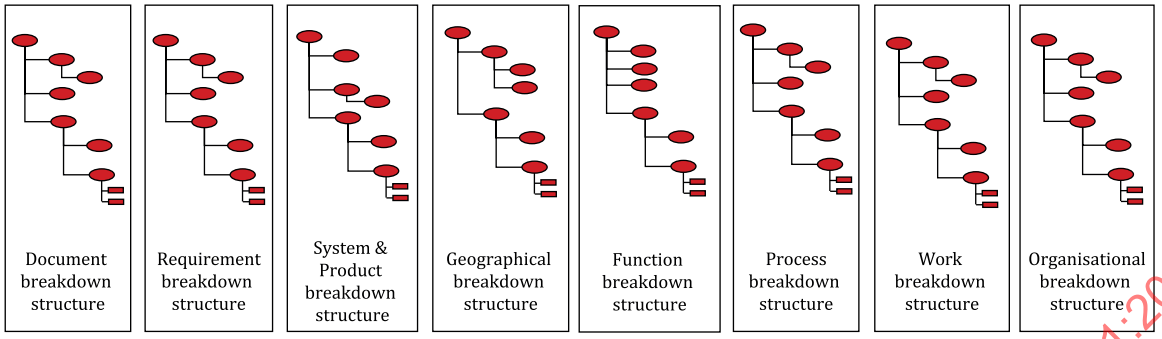


Figure 7 — Applicable breakdown structures in projects delivering a physical object

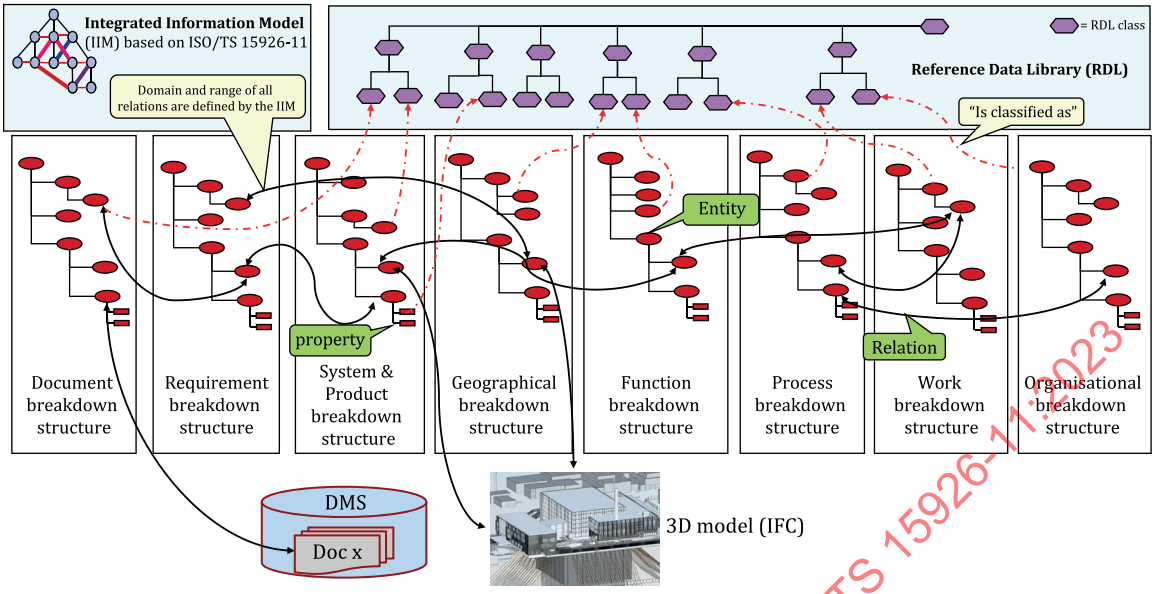
To have control over the composing elements of the breakdown structure in systems engineering a configuration management process has been defined. In this process entities such as breakdown structure elements (also called entities or configuration items) are recognized and followed in terms of, for example, status and applicable relationships with other entities and properties (sometimes also called attributes). The allowed relationships between breakdown structure elements should be defined in an integrated information model (IIM). It is called integrated since the total of breakdown structure covers all involved disciplines and covers the whole life cycle). This document offers the building blocks to build such an IIM (if required even project specific) based on entities, relationships, and attributes (properties and annotations). To be able to do this efficiently the following rules are recognised:

— Each entity and each attribute that is on a project data level should be an instance of an entity of the project specific IIM based on this document.

— Each entity and each attribute that is on a project data level should be classified according to the RDL as defined by the project.

— Each relationship between two elements is defined in an IIM represented by a set of relationships with defined domain and range.

These rules are visualised in Figure 8 where the relationships are shown of entities and attributes to document versions in a document management system to 3D objects in a 3D model of the plant or system. This allows the realization of a complete and consistent handover of data containing graphical data, non-graphical data and documents, defined as a CDE in the building information modelling (BIM) standard series ISO 19650.



Key

- Breakdown structure element (instance of an IIM entity).
- Classification of breakdown structure elements as an RDL class.
- Defined association between breakdown structure elements.

Figure 8 — Principle of connecting breakdown structure elements to an RDL and inter- connect elements via relationships

4.6 Properties

Properties are in general recognized in industry to be essential to be able to specify and characterize things and to check if these required characteristics and specifications are fulfilled in the design and during operations. This document intentionally distinguishes the difference between quantitative properties and qualitative properties. A quantitative property in general concerns a characteristic which is expressed by means of a quantity defined by a real number on a scale (mostly an explicit unit of measure). The quantity can hold metadata, e.g. where the number is obtained from, its datetime and accuracy (see Figure 9).

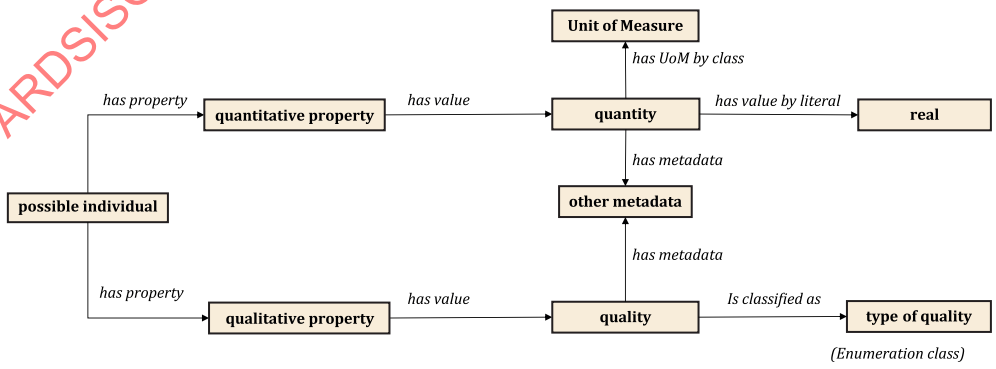


Figure 9 — Modelling of properties (expressed in a neutral semantic way)

A qualitative property concerns a characteristic or aspect which value by means of a quality only can be expressed by a string value. The value mostly is a value selected from an allowed set of values (enumeration) for that specific property. These values are collected in an enumeration or class so that

it can be validated if an assigned value is allowed or valid in case of a specific instance of a possible individual.

5 Semantic modelling methodology

5.1 General

Clause 4 presented fundamentals without any language binding. In this clause a semantic data modelling approach of these fundamentals is presented.

5.2 Substantiation of the choice for RDFS

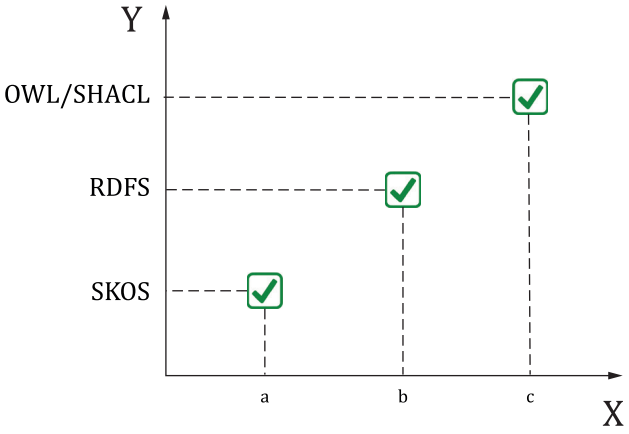
As substantiation for the choice for RDF(S) as a semantic modelling technique, in Figure 10 the three main W3C standards are classified according to their capabilities in a "capability model". The capabilities of these three main W3C standards as shown in Figure 10 can be summarized as follows:

— SKOS: offers the weakest semantic data modelling capability of the W3C standards. This level mainly targets uniformity in human understanding of terms and definitions and making sure at least that the data is machine-processable. Weak modelling can be sufficient as a first step for human interpretation. A good definition gives an end user guidance on how to classify and instantiate their data later according to these terms. Terms and definitions can be distributed as well as published on websites for others to refer to and reuse.

— RDFS: whenever uniformity is needed to exchange or share asset data between digital systems of different parties, more expressive power is needed where data is classified according to ontologies involving concepts, datatypes, attributes and relationships or restrictions. This stronger level builds upon the lowest level making the data also machine-interpretable.

— OWL: to be able to really integrate data for all kinds of decision support, even stronger semantics are needed. This stronger knowledge comes in the form of explicit constraints and rules for the data against which data can be verified respectively inferred in a closed world fashion. These constraints and rules can be definitional or operational, the latter specified by the client brief or legalisation/regulation body on top of the definitional ones.

This document aims to support the creation of project data that can be shared between parties in such a way that the structure and meaning of the exchanged data is unambiguous, despite the software that has created the data or has to interpret the data. Validation should be possible for all project data against the definitions of this document by means of the defined domain and range of the relationship. From this point of view, plain RDF(S) is sufficient to achieve this goal.



Key

X application
Y capability level
a Terms and definitions.
b Information exchange and or sharing.
c Information verification and inference.

Figure 10 — Capability model of W3C standards

5.3 The use of RDFS in this document

All things described by RDF are called resources, and are instances of the class rdfs:Resource which is the class of everything. The language defined by RDF Schema consists of a collection of RDF resources that can be used to describe other RDF resources in application-specific RDF vocabularies. This document aims to be such a vocabulary. RDF Schema is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. RDF Schema is written in RDF using the terms described in this document. These resources are used to determine characteristics of other resources, such as the domains and ranges of properties.

RDF Schema describes properties in terms of the classes of resource to which they apply. This is the role of the domain and range mechanisms described in this document. For example, the ex:author property can be defined to have a domain of ex:Document and a range of ex:Person, whereas a classical object oriented system can typically define a class ex:Book with an attribute called ex:author of type ex:Person. Using the RDF approach, it is easy for others to subsequently define additional properties with a domain of ex:Document or a range of ex:Person. This can be done without the need to re-define the original description of these classes. One benefit of the RDF property-centric approach is that it allows anyone to extend the description of existing resources, one of the architectural principles of the World Wide Web. Figure 11 shows the selected resources of RDF Schema which forms the bases of the ontology of this document. The central ones are rdfs:Resource, rdfs:Class, rdf:Property, rdfs:subClassOf, rdfs:subPropertyOf and rdf:Statement.

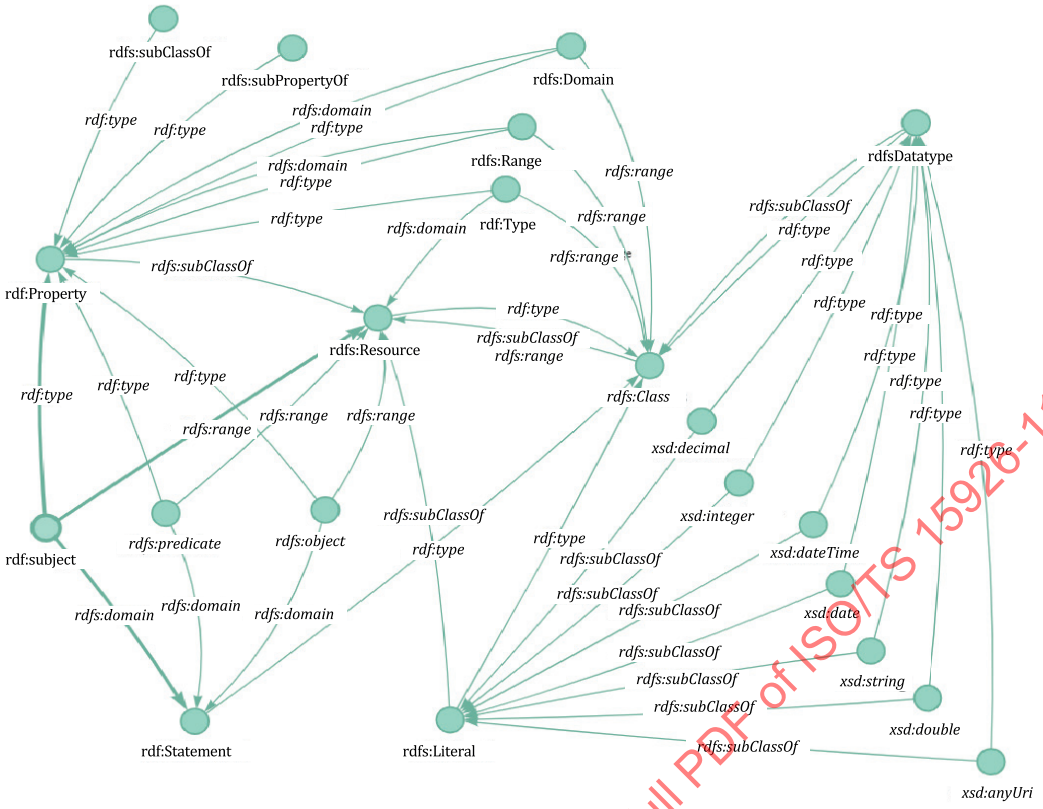
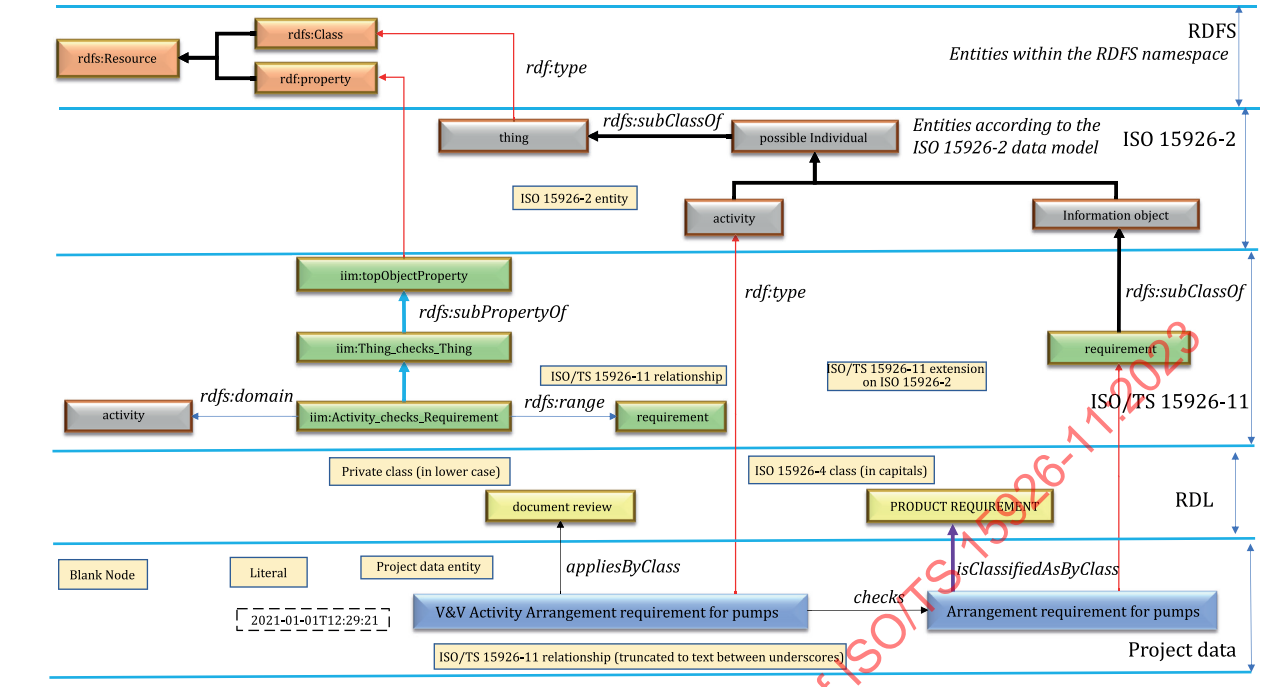


Figure 11 — Selected subset of RDF Schema as bases for this document

skos:preflabel (for assigning names) and skos:definition (for assigning definitions) are used for assigning names and definitions to resources of the ontology in this document (Annex A).

5.4 Symbols used in figures

This document contains diagrams which show instantiations of the ontology in this document. The concise notation is used in these diagrams as defined in Figure 12 (showing a few typical constructs).



Key

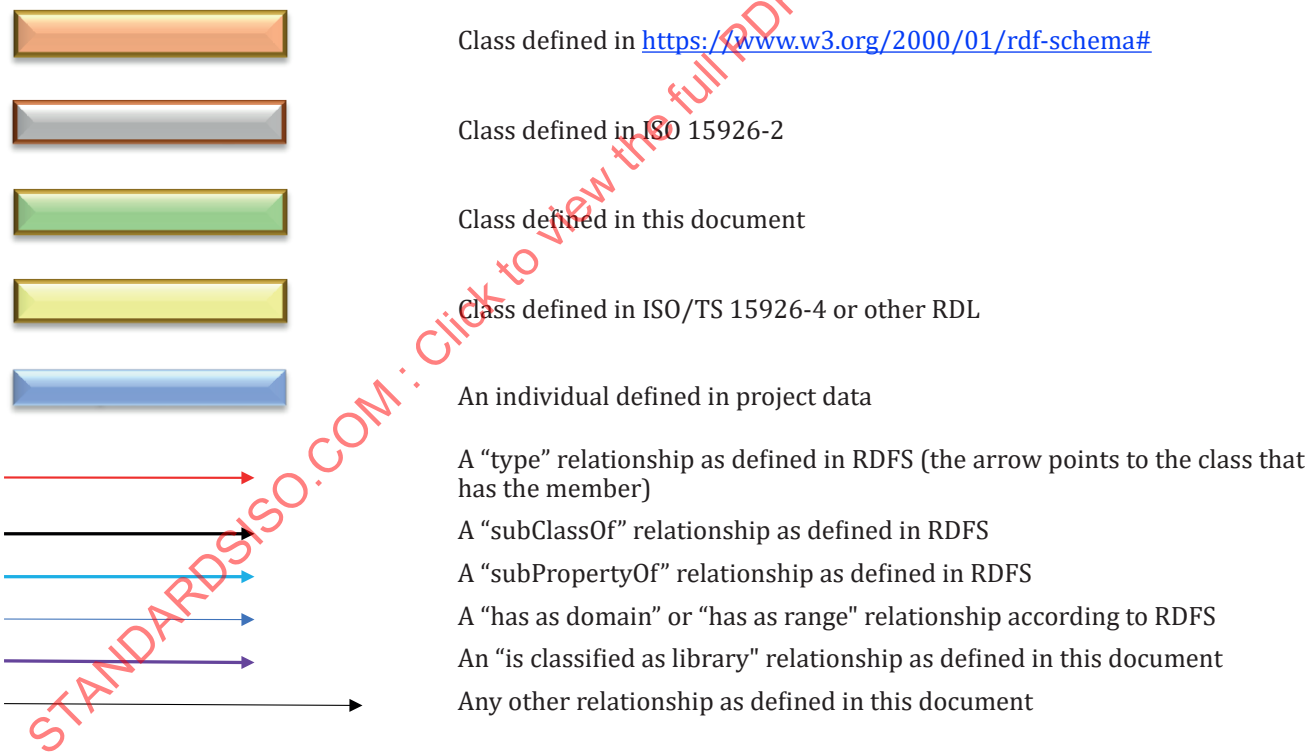


Figure 12 — Symbol and line conventions used in the figures in this document (examples)

The entity classification in ISO 15926-2 is implemented by rdf:type. The entity specialization in ISO 15926-2 is implemented by rdfs:subClassOf in case of a class hierarchy and rdfs:subPropertyOf in case of a relationship hierarchy.

5.5 Reference data

In this document, reference data can be obtained from ISO/TS 15926-4 (in examples in all capital letters, see Figure 12) or may be a “private class" (in examples entirely in lowercase only, see Figure 11). Private classes can be derived from other sources such as the ISO/IEC 81346 series, Uniclass, OmniClass and

CFIHOS. This allows project partners to bring in their own knowledge and tools as long they harmonize their libraries or families with a central project RDL. In all cases the "isClassifiedAsByLibraryClass" relationship is used to classify a project entity as an RDL class (see Figure 12). A working method hereof can be to define a specific project or company RDL in which relevant ISO/TS 15926-4 classes and the private classes (with reference to their original source including the corresponding identifier) are integrated on bases of the ISO/TS 15926-4 hierarchy. This enriches the scope and flexibility of using this document with respect to the many interrelated disciplines in a project. If a project specific RDL is chosen to be created, the ontology of this document can be used for that purpose. For the purpose of building a RDL the ontology can be extended with applicable SKOS concepts like skos:narrower and skos:broader. Further development of an RDL however is out of scope of this document.

5.6 Identification and references in the text of this document

The classes and properties which are part of the ontology defined by this document have natural language identifiers which are in lower case, in bold font, and which contain spaces where appropriate.

The classes and properties defined by this document that are presented by means of a URI with suffix derived from their natural language identifiers as follows:

— spaces are removed and encoded by camel-case;

— classes have an initial upper-case letter;

— relationships have an initial lower-case letter for the semantic relationship part (part between underscores). Both the domain and range part of the relationship starts with a capital;

— Within the text in Clauses 5 and 6, references to a class or property as part of the ontology of this document are presented using the `Courier New` font.

5.7 Incorporation of ISO 15926-2 within the ontology of this document

5.7.1 Entities

In Figure 13 both the core entities of ISO 15926-2 and EN 17632 are incorporated in the core entity hierarchy of this document. The entities are structured according to ISO 15926-2 on one hand and according to RDFS on the other hand with `rdfs:Resource` as top element. Additional `state of individual` is added as a specialisation of `part life individual` which is part of the ontology in this document. `Property` is redefined in the ontology in this document for simplicity reasons, since it originally has another place in the ISO 15926-2 ontology. In the area assigned with “ISO 15926-2” in Figure 13 the entities are according to ISO 15962-2.

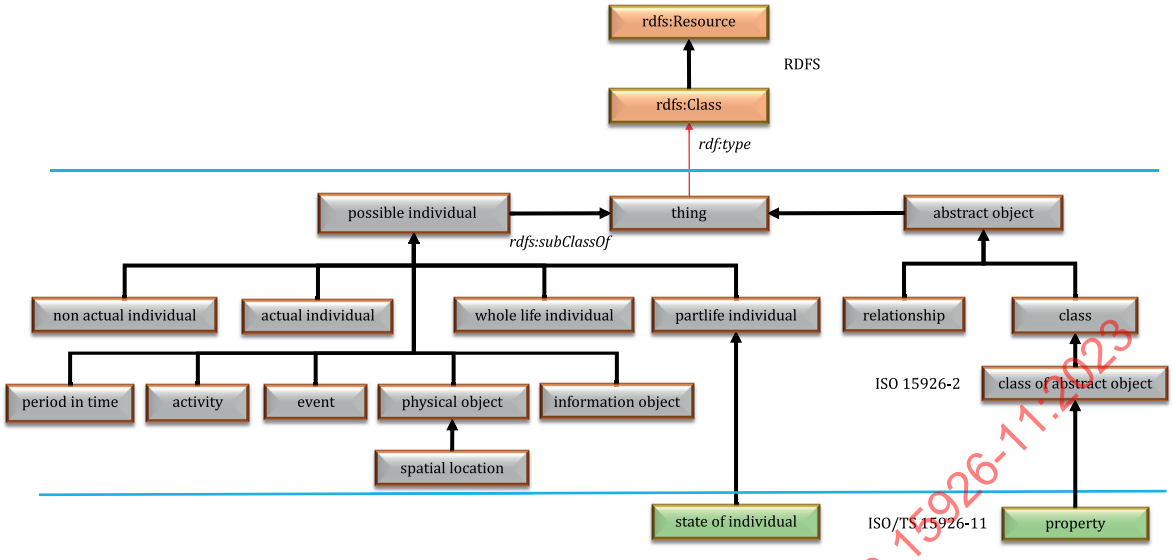


Figure 13 — Core entities of this document corresponding with the core of ISO 15926-2 and EN 17632

5.7.2 Relationships and their characteristics

Semantic relationships can be defined in RDF by creating a reference library of relationships according to the needs of each systems engineering based project.

Figure 14 is an example of how the relationships can be connected to an ISO 15926-2 compliant model. The following principles enable these relationship connections.

— The entity relationship in ISO 15926-2 is implemented by `rdf:Statement`.

— The entity class of relationship in ISO 15926-2 is implemented by `rdf:Property`.

— In data that conforms to this document, a relationship is usually represented as an RDF triple. These relationships are reified as an `rdf:Statement` to support the providence of meta data to relationships and enabling making statements about statements.

In ISO 15926-2, a `class of relationship` is bi-directional. An `rdf:Property` however has a forward direction which is also the case for all relationships defined in this document.

Classes in both ISO 15926-2 and this document can assume the role of a domain class or range for the relationships that are defined in this document.

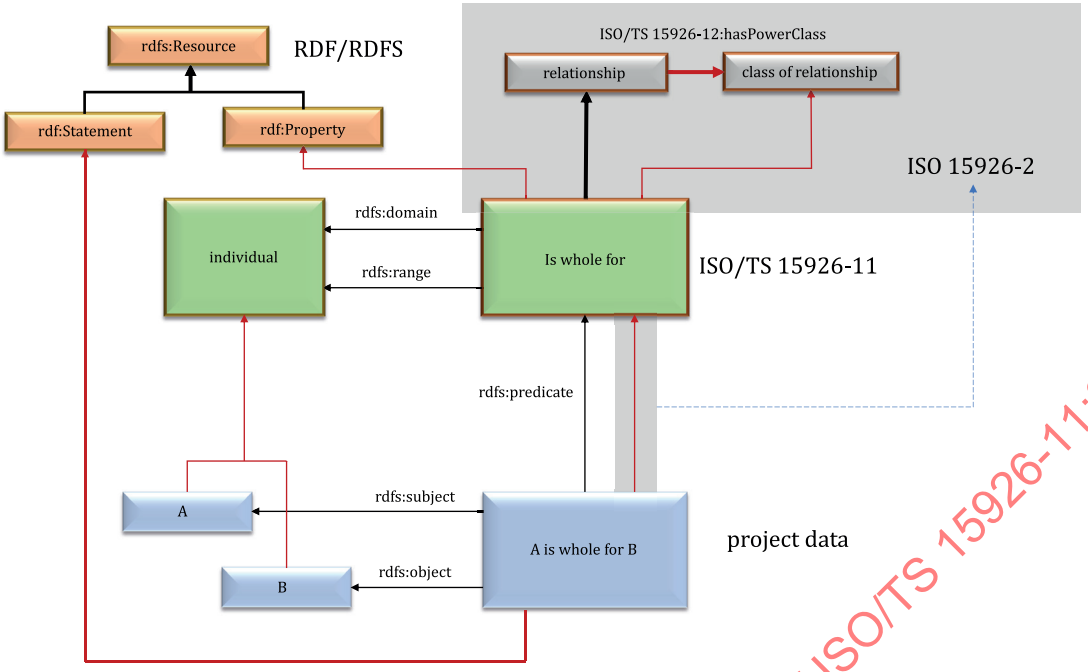


Figure 14 — How relationships of this document connect to ISO 15926-2 and RDFS

Additional to the principles shown, this document has introduced two root relationships for the RDL of relationships which both are a `rdf:type` of `rdf:property`:

— `top object relationship` between subjects and objects not being a literal.

— `top literal relationship` between subjects and literals.

These two top relationships of this document are positioned in Figure 15 with respect to ISO 15926-2 and RDFS. By means of `rdfs:subPropertyOf` specializations a hierarchy of relationships is developed in this document starting with these two top relationships. As example in Figure 15 the 'is whole for' relationship is specialized from `topObjectRelationship` (this is the inverse relationship of the "is part of" relationship in ISO 15926-2) and 'has label by literal' which is specialized from `topLiteralRelationship`.

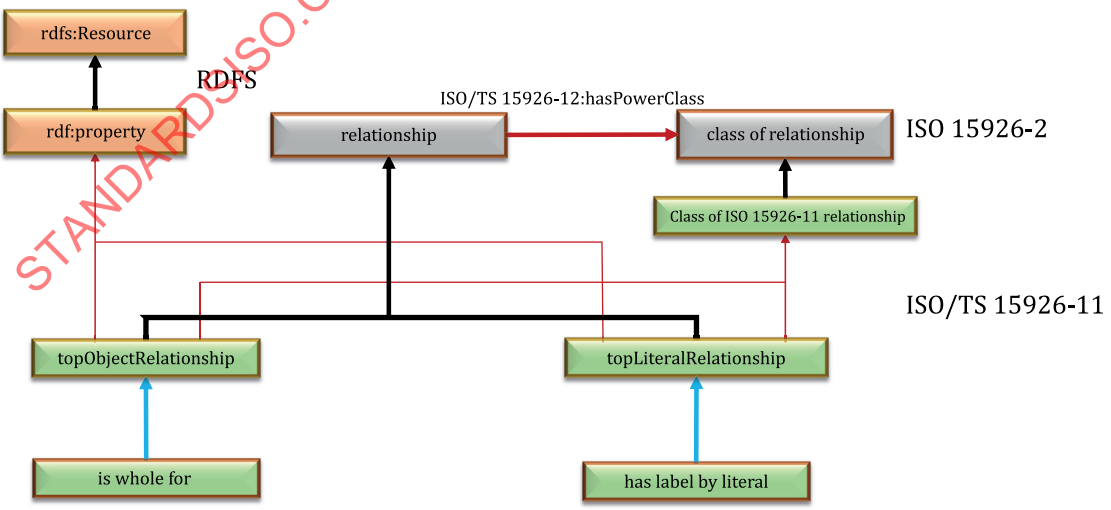


Figure 15 — Positioning of the two top relationships of this document with respect to ISO 15926-2 and RDFS

Rules for creating relationships in the context of this document:

— when statements are exchanged in the supply chain, only relationships in the prescribed direction shall be used as defined in Annex A;

— every new kind of relationship start with a root version of that new relationship where both `rdfs:domain` and as `rdfs:range` has `thing` as value;

— each relationship in this document can be specialized by means of subclasses of thing as domain and range in order to prescribe the allowed use of these specialized relationships when creating and validating project data.

By implementing these rules within a configuration management environment, the semantic quality of statements, consistency and quality of data will be improved.

NOTE 1 If applications want to use relationships in the inverse direction, it is possible to define explicitly the inverse names of the relationships, which can be presented by these applications.

NOTE 2 The relationship `topObjectRelationship` is comparable with topObjectProperty in the OWL namespace and `topLiteralRelationship` is comparable with topDatatypeProperty in the OWL namespace. In the ISO 15926 series, both relationship and property have a specific meaning and therefore the term relationship is used in the top objects of the relationship hierarchy.

5.7.3 Properties

According to ISO 15926-2 a `property is a class_of individual` whose member individuals have the same degree or magnitude of a quality or characteristic. The types of quality or characteristic are defined using `class_of_property` which can be divided between property continuums and enumerated sets of properties. In this document a `property` is interpreted by the way ISO 15926-2 defines a class of property where this document specialises a property into a `quantifiable property` and a `qualifiable property` (using an enumeration) based on what is common practice. This document allows ISO 15926-2 property and indirect property and ISO 15926-2 shape dimension to be treated as just one concept namely the ISO 15926-11 `property` (see Figure 13). Each instance of a `possible individual` in general can in this document be characterised by means of one or more properties. This can be seen as a simplification of the property concept as modelled in ISO 15926-2.

The modelling of properties in this document differs from the approach of ISO 15926-2. This is because the approach used to define properties in ISO 15926-2 does not in general match the way other RDLs are commonly represented in design software. Nor does it conform to the design practices used in larger companies. In common practice, and observable in many industry libraries, no difference is made between basic properties, indirect properties and shape dimensions. The ontology in this document defines the following specializations of `property`:

— `quantitative property`

— `qualitative property`

In order to be able to indicate date and time in the same way, the following properties have been added in this document:

— `date property`

— `duration property`

In this way, instances of these four property types can be classified as properties in any RDL. This simplifies the uses of properties, and is the recommended practice recognized in many BIM standards.

Figure 16 is an example of how to model a quantitative property as applied to an instance representing a `Spatial Location` (Building entrance) defined by an RDL class with a value and a Unit Of Measure (UoM) in reference to the RDL. A `qualitative property` is also applied to the same Building entrance using a defined value in reference to an RDL class.

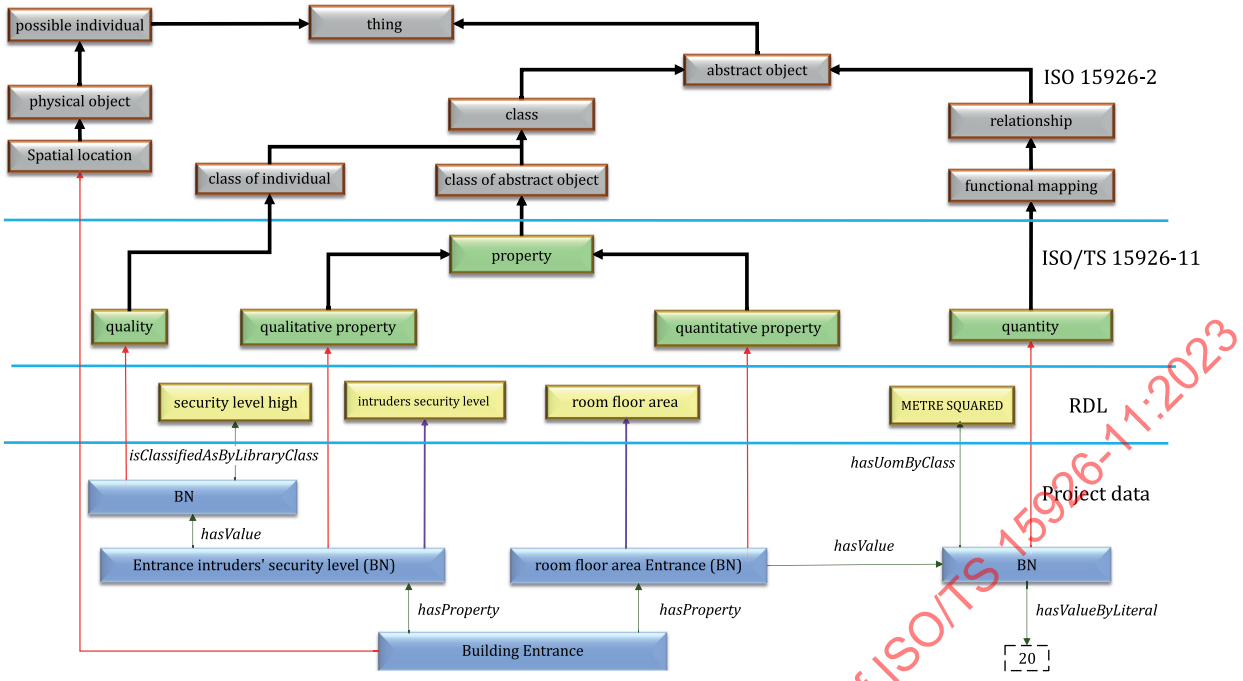


Figure 16 — An example of the modelling of a quantitative and a qualitative property.

5.8 Expanding the ISO 15926-2 ontology in this document

5.8.1 General

When analysing the use cases from 4.2 a collection of key terms can be identified that together forms a thread through these use cases. A representative set of these key terms (not exhaustive) are in this subclause placed as entities in the ontology of this document (mostly derived from ISO 15926-2 and ISO/TS 15926-4).

5.8.2 Additions to the individual hierarchies

In Figure 17 typical system engineering terms are defined as classes of this document (green boxes) as extension on ISO 15926-2 entities (grey boxes) starting with the block labelled `possible individual`. The extended classes were derived from the use cases given in 4.2.

The ontology of this document may be extended by implementers by creating private classes that are subclasses of one of the classes shown in Figure 17.

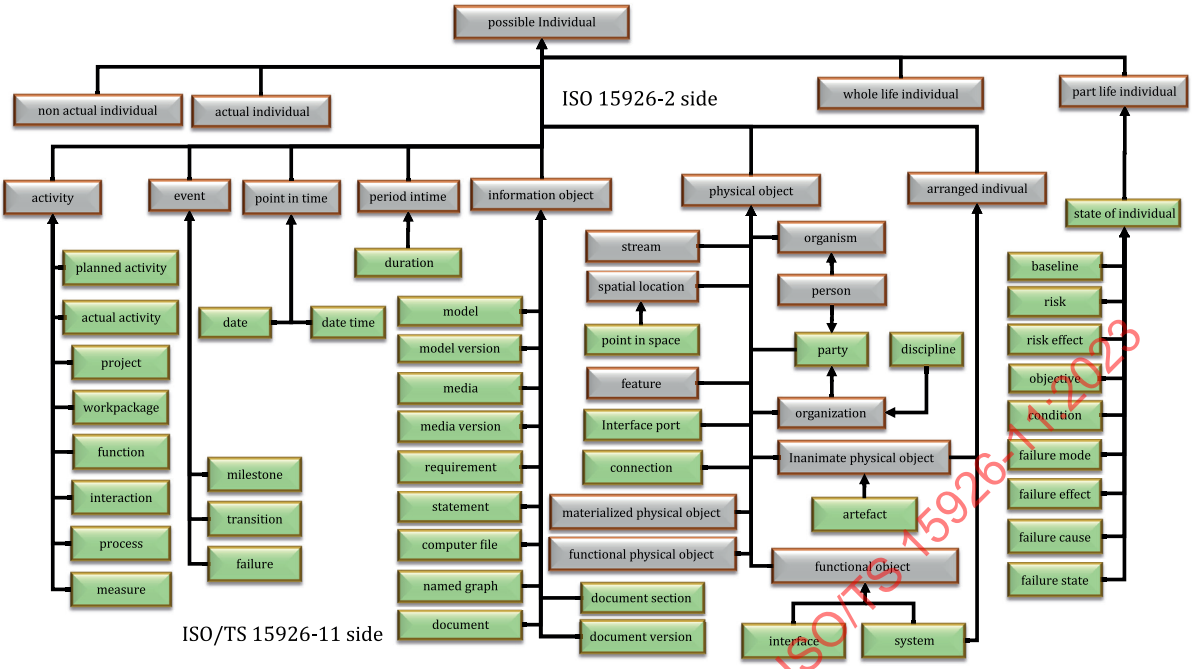


Figure 17 — Expanding ISO 15926-2 possible individual with entities in this document

5.8.3 Additions to the abstract object hierarchy

In Figure 18 additional system engineering abstract classes are defined as classes in this document (green boxes) as extension on and subclasses `abstract object`. Also, these classes were derived from the use cases given in 4.2.

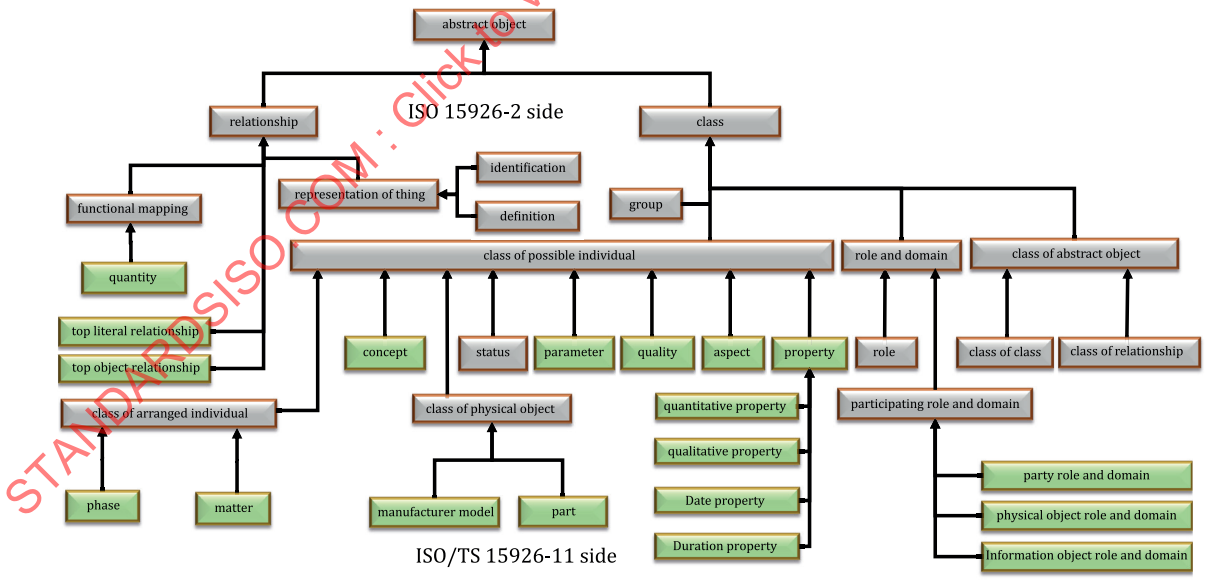


Figure 18 — Expanding ISO 15926-2 abstract object with entities in this document

The ontology in this document may be extended by creating private that are subclasses of one of the classes shown in Figure 18.

5.8.4 Creation of semantic relationships

Figure 19 gives some examples how `topObjectRelationship` can be specialized by means of `rdfs:subPropertyOf` into more specific relationships, each characterized with an applicable domain and range (which discriminates the specialized relationships from the more generic one) however the definition of the corresponding root relationship will always still be valid.

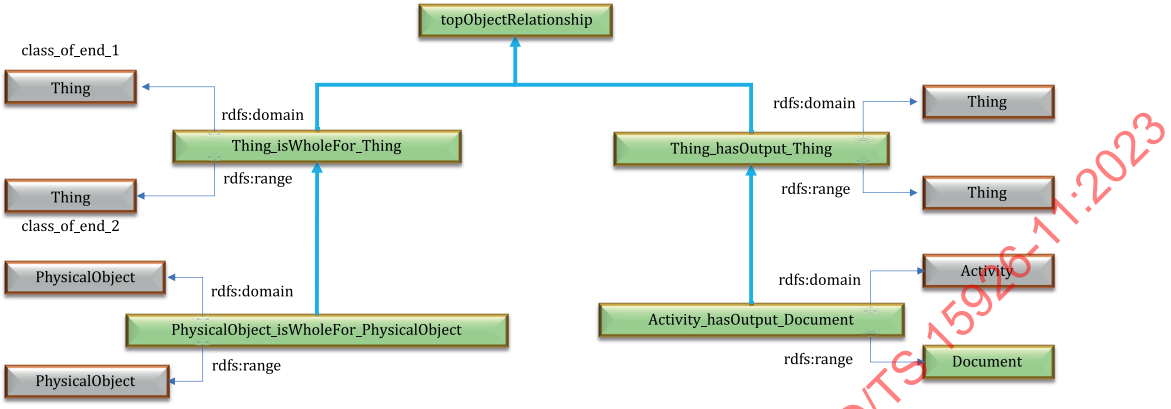


Figure 19 — Model pattern of relationships within this document

This document offers a way to define specific literal relationships to assign literals with various meanings to resources by means of several predefined 'ByLiteral' relationships rather than using the `rdfs:label` option. In this way, the role that literals play can be explicit with respect to the object. In Figure 20 some examples are shown of such 'ByLiteral' relationships which can be constrained by domain and range.

In Annex A an initial set of relationships is presented by means of a list of root relationships (with `thing` as `rdfs:domain` and `rdfs:range`) and a list of specialised relationships with specific domain and ranges.

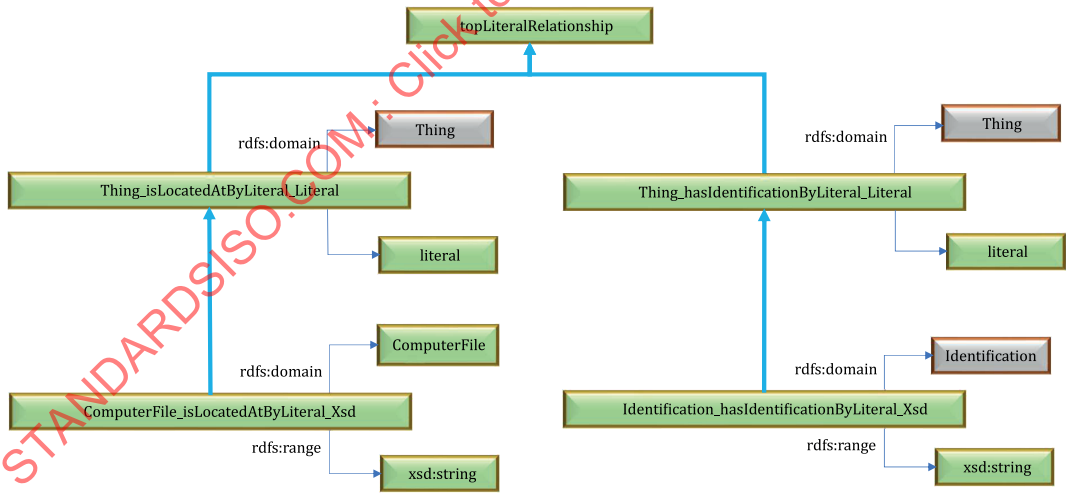


Figure 20 — Examples of specializations of topLiteralRelationship

Figure 21 is an example with instantiation of several literal relationships as part of project data.

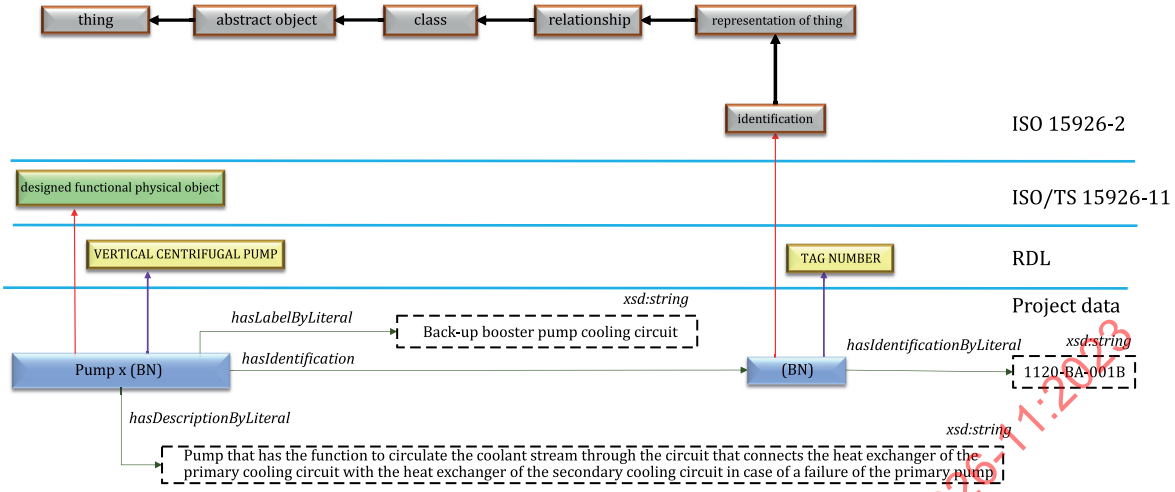


Figure 21 — Examples of relationships of type 'byLiteral' as defined in this document

5.9 Class of class mechanism

In order to be able to restrict the allowed classes or values as the range of a relationship, this document makes intensive use of a class of class mechanism. The principle of this mechanism is shown in Figure 22 by means of specific collections of statuses each fulfilling a role in a specific context. The status types are predefined in the RDL and can be a member of one or more collections (where a collection is a `class of class`). This mechanism enables an effective validation of project data.

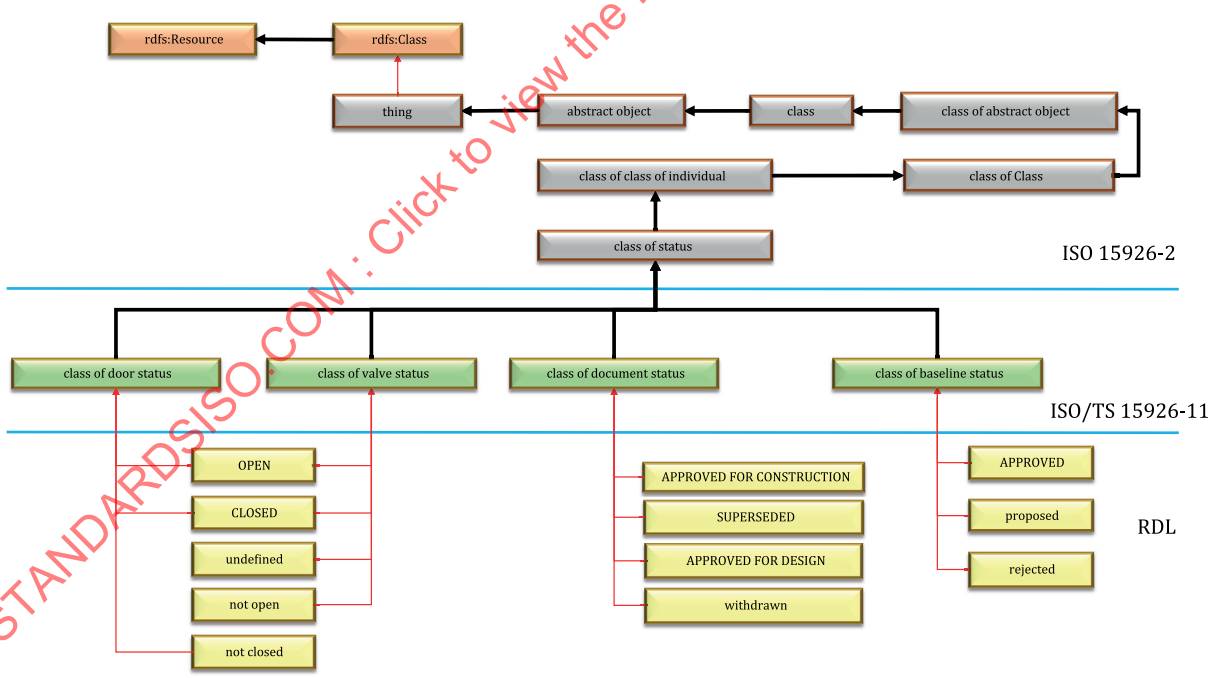


Figure 22 — Examples of the use of class of class mechanism to define controlled subsets of RDL classes

Each `class of class` in Figure 22 shown in the ISO 15926-11 area can function as the range of a `relationship` in order to constrain the usage of that relationship. Based on this, one can validate handover data by means of checking the object in a statement against the defined range of the specific relationship used in that statement (it should be an instance of a member of the class defined as range or in case of a class reference one of the members itself).

The role of the class structure is identical as represented in ISO 15926-2 however can be included in the RDL for maintenance reasons (since all of the members will be a class in the RDL). A specific kind of relationship is the classification relationship between instances of entities of this document and classes of the RDL. In that case `rdf:type` is not used. The relationship used to classify an individual project data entity (instance of an entity of this document) as a RDL class is `iim:Thing_isClassifiedAsByClass_TypeOfThing` or a sub-property of this relationship. The reason for this is that `rdf:type` is reserved for classifying a project entity as an ontology entity of this document and software tools frequently only allows one formal classification.

Figure 23 is an example for the classification of a document status according to this principle where the members of "class of document status" are defined in Figure 22.

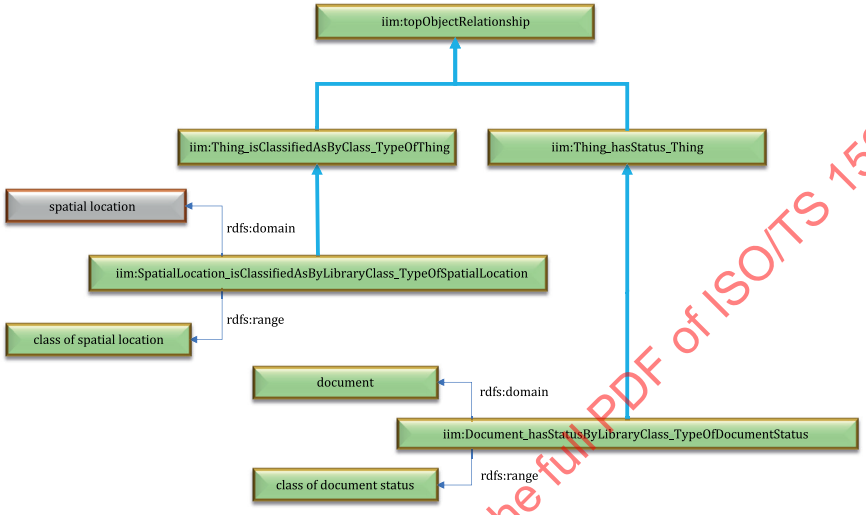


Figure 23 — Example representing the classification mechanism

Figure 24 shows an example of the relationship `iim:Document_hasStatusByLibraryClass_TypeOfDocumentStatus` which is used to assign a status to a document which must be a member of class of document status as defined in Figure 22 and Figure 23. In this way it is possible to force and or verify (by software) that the integrated data fully complies with an authorized document status.

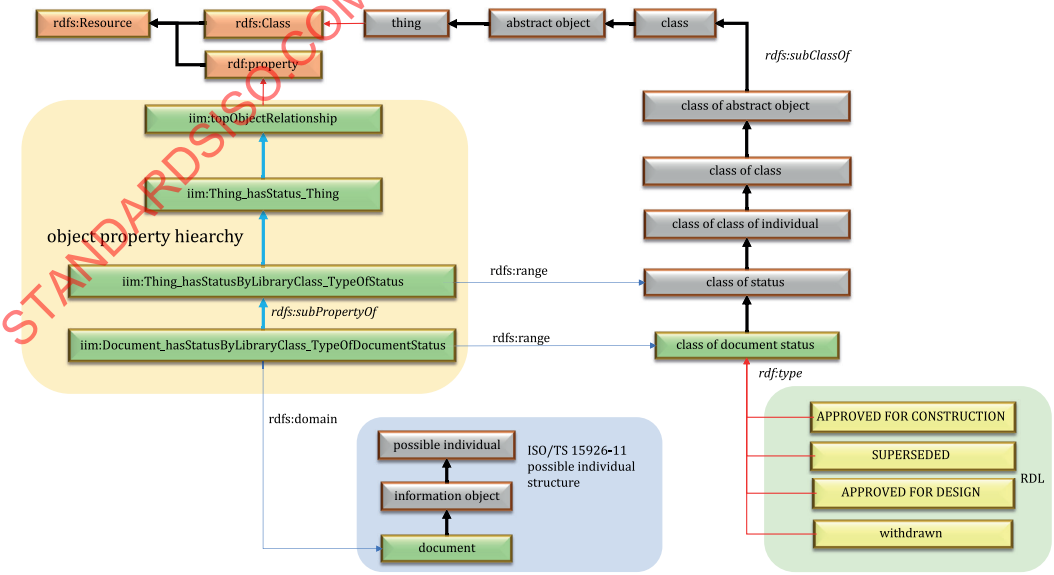


Figure 24 — Example of the use of class of class in conjunction with the corresponding relationship

One can expand the class of class hierarchy in the context of a specific project purpose. In practice this capability is needed due to the utilization of legacy project management systems which were designed for a document driven approach that relied on legacy procedures for the engineering process. Figure 25 gives an example of potential class of classes that elaborate on the class of class hierarchy utilized by ISO 15926-2.

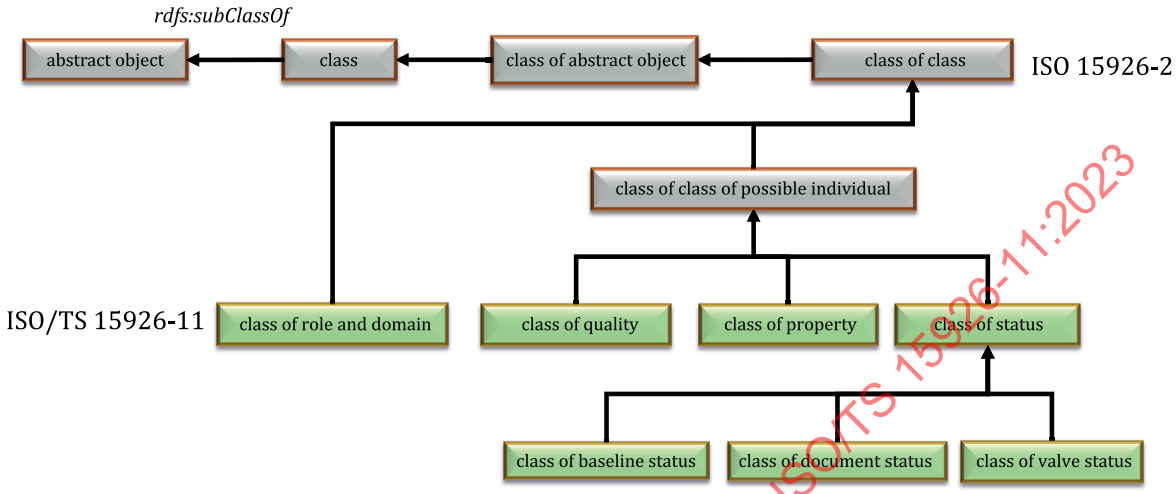


Figure 25 — Sample of the class of class hierarchy elaborated on ISO 15926-2

5.10 Life-cycle model

5.10.1 Life-cycle model as defined in this document

An essential thread within systems engineering is the recognition of life cycle stages of both a system and its composing elements. There are four main stages recognized in this document where the first two can be projected on the left side of the V-model and the latter two can projected on the right side of the V-model. The four quadrants of the life-cycle model as shown in Figure 26 (derived from Figure 5) are represented by the following entities in this document (supplemented with alternative names as used in industry):

Q1 – `designed functional physical object` (system element, tagged item, logical item)

Q2 – `designed materialized physical object` (as-designed object, technical solution)

Q3 – `actual materialized physical object` (as-built object, asset, installed equipment)

Q4 – `actual functional physical object` (actual system element, actual performance)

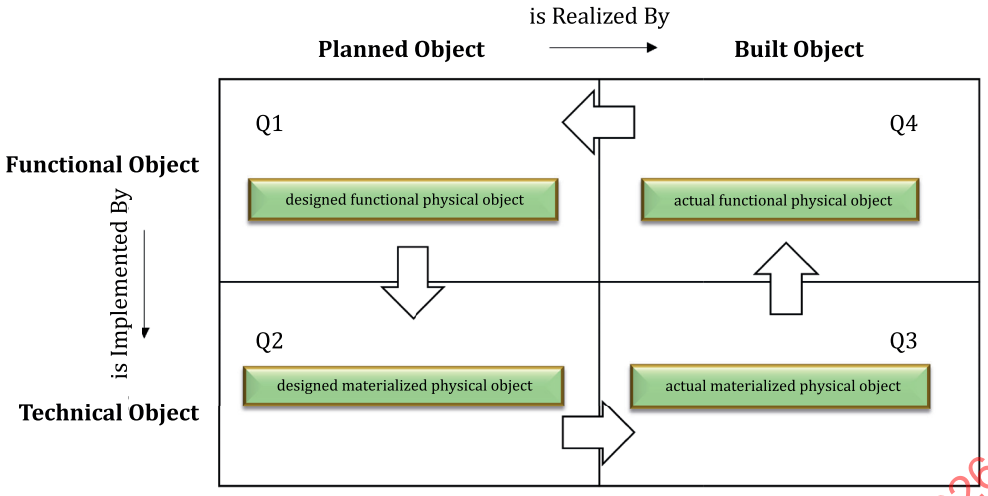


Figure 26 — Fundamental life cycle quadrants in the context of systems engineering

In Figure 27 the four life cyle entities are multi classified with respect to ISO 15926-2 entities with `functional physical object` and `materialized physical object` both as specialisations of `physical object`. This approach avoids having to repeatedly re-instance each usage to the three separate ISO 15926-2 entities.

ISO/TS 15926-11 life cycle object	Superclass 1	Superclass 2	Superclass 3
designed functional physical object	whole life individual	non actual individual	functional physical object
designed materialized physical object	whole life individual	non actual individual	materialized physical object
actual materialized physical object	whole life individual	actual individual	materialized physical object
actual functional physical object	whole life individual	actual individual	functional physical object

Figure 27 — Fundamental life cycle quadrants related to ISO 15926-2 entities

5.10.2 Usage of the life-cycle model

In Figure 28 the life-cycle model introduced in 5.10 has been expanded using semantic relationships. It applies the life cycle model to the main pump system of the intermediary cooling system of a nuclear reactor. Figure 28 covers the first three quadrants of Figure 27 (see names below the boxes in the project data area). Quadrant four is a virtual representation of the first quadrant which can be used as a placeholder for registration of operational data with the purpose of, e.g. analysis and or predictive maintenance.

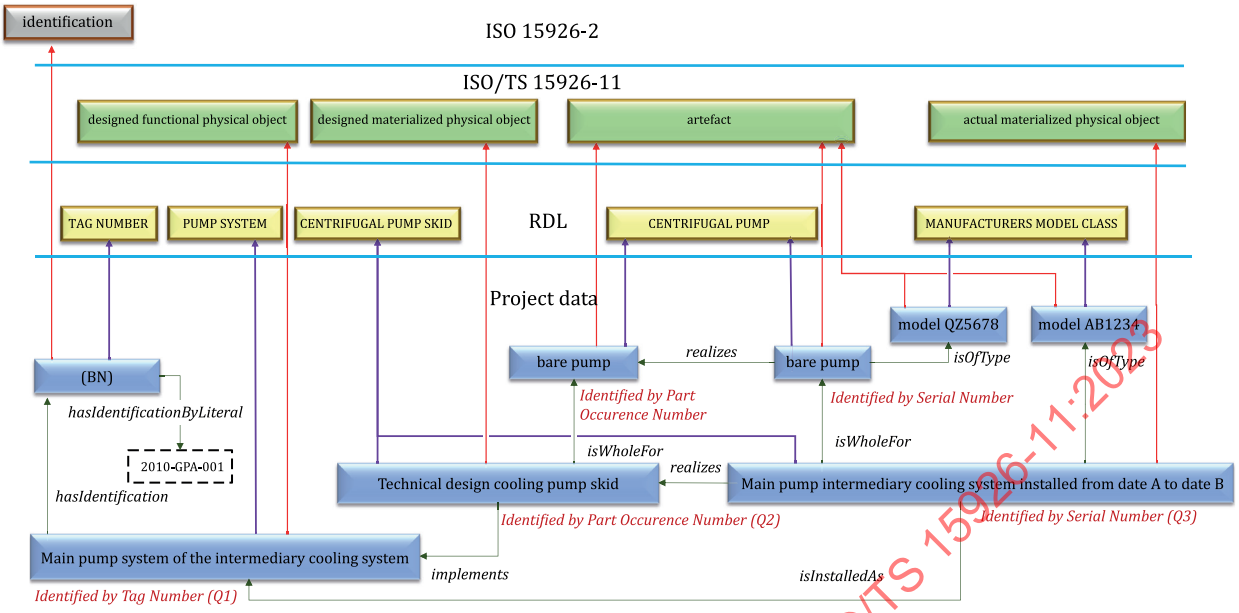


Figure 28 — Example of the life-cycle model applied on a pump

The example depicted in Figure 28 starts with identification by means of a tag number of the pump system as part of the intermediary cooling system of a nuclear reactor. In practice the pump system is called a tagged item. This `designed functional physical object` is implemented by a `designed materialized physical object` which is the representation of the technical design (which can be an occurrence of a reusable design solution defined as a `part` in the company part library). Since in fact the `designed materialized physical object` is an occurrence of that reusable design (called an engineered item, relevant for re-usages of designs and the procurement process) it is identified by a PartOccurenceNumber.

As a result of the design solution, a decomposition can be developed in which the constituent parts in Figure 28 are classified as an `artefact`. These parts can also be an occurrence of a part from the company part library. Optional the constituent parts can also modelled as a `designed materialized physical object` rather that an `artefact`.

Finally, the `designed materialized physical object` is realised by a pump skid which is installed as an `actual materialized physical object` on the place of the `designed functional physical object` (which has the tag) and "isOfType" a manufacturer or supplier model.

The `actual materialized physical object` at some point in time will be replaced by another (as result the "isInstalledAs" relationship will be replaced/updated by the new one) `actual materialized physical object`, may be based on another `manufacturer model` which on itself should fit within the instance of the reusable design solution (`part`). If not possible, e.g. because of end of life or end of service of a manufacturer model, then the `designed materialized physical object` has to be changed, eventually using another instance of the same `part` with another, similar `manufacturer model`. In case the new `manufacturer model` requires modification of the original part, a new `part` must be introduced.

5.10.3 Comparison with structuring principles defined in IEC 81346-1

Where this document offers the possibility to divide the life cycle of a system element into four separate entities, in IEC 81346-1 the entire life cycle of a system element is represented by just one entity. The different life cycle aspects are related to this entity, for example based on document types. With this, IEC 81346-1 requires a different approach to configuration management than the ISO 15926 series. The basic structuring of systems and system elements in IEC 81346-1 according to decomposition, function, location, product, and classification can be done in this document by means of relationships between the system or system element and entities that represents or covers the various aspects.

Figure 29 shows the relationship to function, location, classification, and decomposition based on relationships taken from the initial set of relationships. The information of a specific life cycle is assumed to be captured in a dedicated document such as a requirement specification as shown in Figure 29 or technical specification or manufacturer documentation.

A designation methodology is not in the scope of the ISO 15926 series and any designation methodology or procedure may be used and assigned the code by means of a literal relationship.

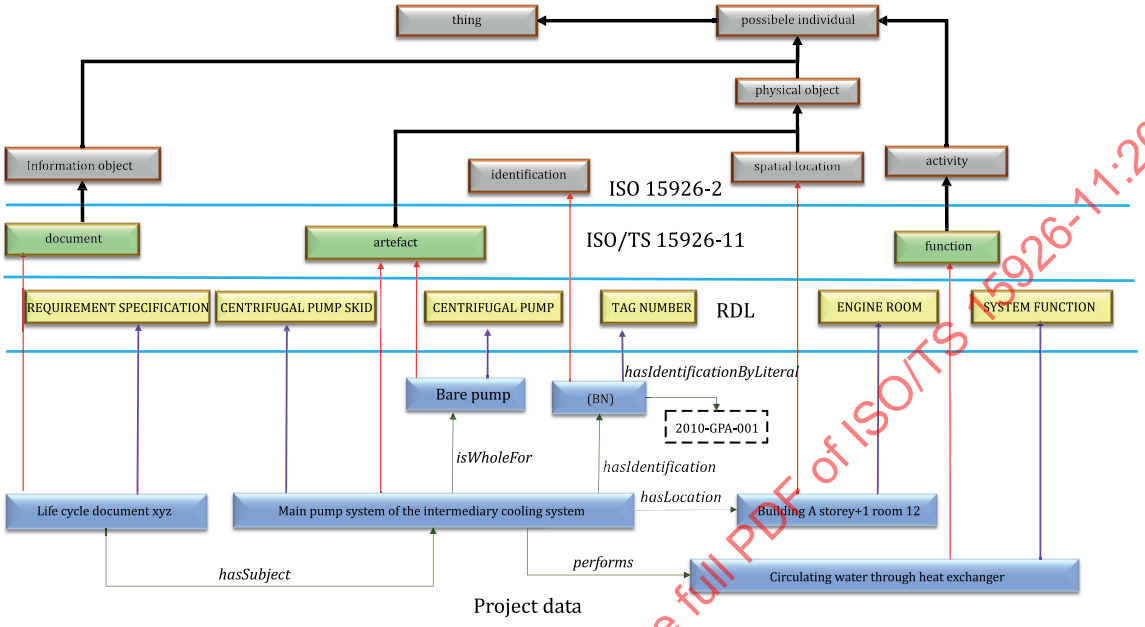


Figure 29 — Modelling structuring principles of IEC 81346-1 using this document

6 Examples of creating project data using this document

6.1 General

The aim of this document is to realize a human readable expression of engineering data that can be supported and managed by computers, where the ultimate aim of ISO 15926-2 is a full understanding and processing of engineering data by computers. The point of this document is that engineering data expressed according to this document can be (partly manually) transformed to a level that is in conformity with ISO 15926-2. For this reason, some principles of ISO 15926-2 are respected and others are simplified by means of "short cut" relationships.

The methodology of this document is built upon the principle that each project entity is at least a type of a ISO 15926-2 entity or an entity of this document by means of the `rdf:type` relationship and is classified as a RDL class by means of `iim:Thing_isClassifiedAsByLibraryClass_TypeOfThing` unless it is intentionally meant as a blank node in the data set, therefore only designated an ISO 15926-2 entity or an entity of this document. Secondly, the principle is to use extensively expressive semantic relationships which are defined by means of a domain and range. The domain is always an entity from ISO 15926-2 or this document. The range can be an entity from ISO 15926-2 or this document or an RDL class of `class of class` which has as members one or more RDL classes.

6.2 Modelling of documents: status and version

In Figure 30 a modelling example is presented of a `document` and corresponding `document version`. In this case there is an instance of a `document` which has as the subject a technical design of pump xyz and is classified as a "general arrangement drawing" according to the RDL. The document has two identifications: a document number and a discipline code. A particular version of this document exists, version A, with the status set to approved. The latter is defined by the relationship `hasStatusByClass`

pointing to the RDL class “Approved”. The `document version` is available as a PDF file which is represented by an object that is an `information object` and classified as a `computerfile` and can be found on the (cloud or network) location defined as http://www.example.org/drawings/pumpxyz-A.pdf.

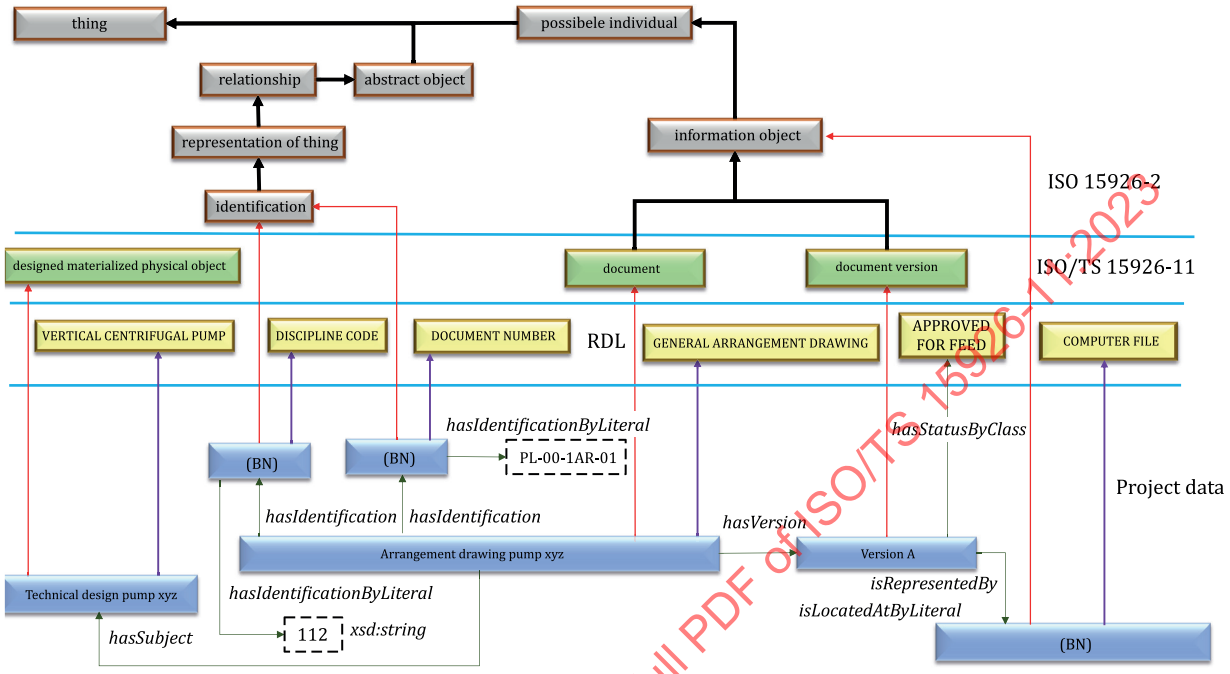


Figure 30 — Example of modelling a document and a corresponding document version

For readability reasons, in Figure 30 the relationship names are abbreviated compared to the names of the relationships as part of the initial set of relationships in Annex A.

Figure 31 is a screen-dump of an ‘off the shelf’ semantic modelling tool based on a graph database in which the model of Figure 30 is implemented based on the initial set of relationships of this document, being the ontology defined by the IIM namespace. In Figure 31 the full relationship name is shown using the IIM namespace rather than the short name in Figure 30.

The modelling tool that has been used, validated imported project data (in this case the example models) against the ontology in this document to check if the domain and range of the relationships are respected and whether the classification of project entities is valid with respect to the RDL used to import these examples.

RDL classes are defined in an RDL which was prepared for the development of this document and based on ISO/TS 15926-4.

The literal relationships (e.g. `iim:Identification_hasIdentificationByLiteral_Xsd`) are implicitly shown by the tool, the literal value is presented directly below the corresponding node (e.g. names, descriptions, and numbers).

In this subclause the examples worked out are all followed by a figure with a screen dump of the graph tool in which the example is implemented, such as in Figure 31.

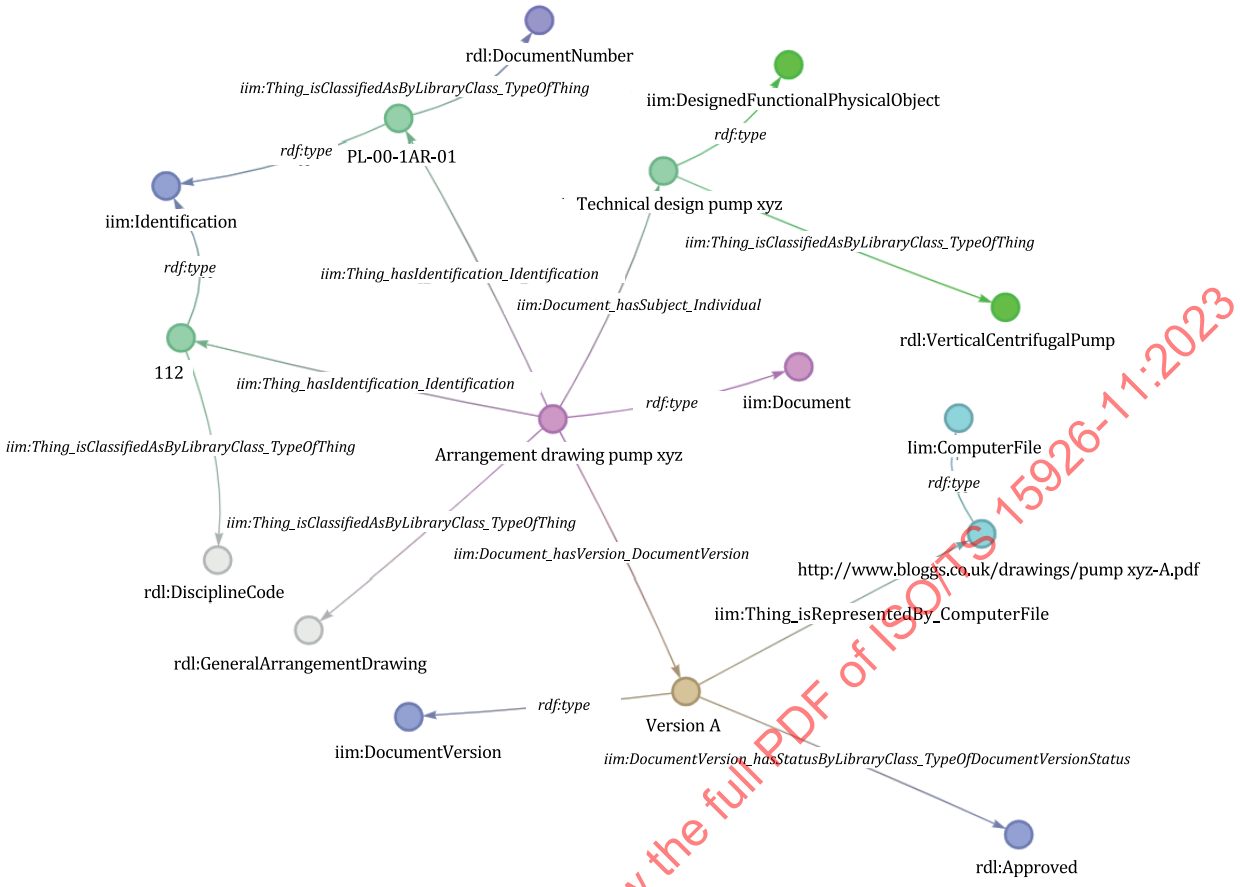


Figure 31 — Implementation of the model of Figure 30 in a graph database application

Figure 32 contains the same arrangement drawing as in Figure 30 except it is defined as an output of an activity classified as a `work package`. The work package realizes all the activities that are needed to represent the detailed design of the cooling system. The location of the designed pump xyz is defined as "isWithin" `spatial location` "room12 on the first floor in building A" and classified as RDL class "engine room".

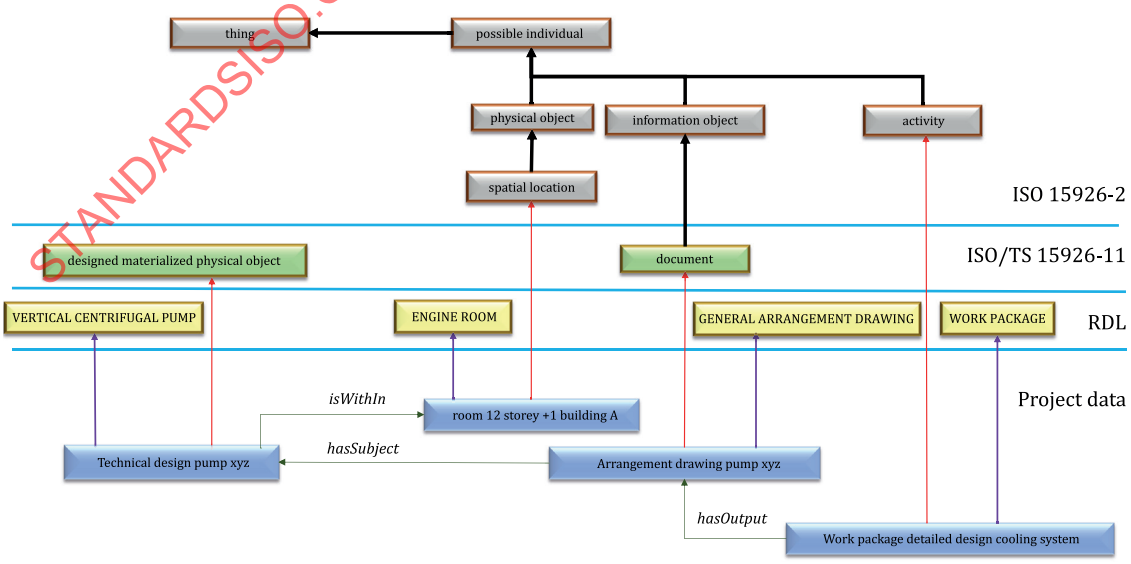


Figure 32 — Example integrating data from various information sources

From Figure 32 the domain of the relationship “isWithin” can be derived as being instances of `designed materialized physical object` and the range of this relationship are instances of `spatial location`. Software can validate if this is the case or demand that this is the case when introducing this relationship in the project data. This document intentionally uses private relationships for its RDL entry since legacy software cannot always handle more than one `rdf:type` relationship per entity. The triple <Technical design pump xyx> isWithIn <room 12 storey +1 building A> is a unique statement in the project data and can be reified by means of `rdf:statement` in order to connect context data or metadata to this unique, individual triple (see Figure 33 and Annex A.

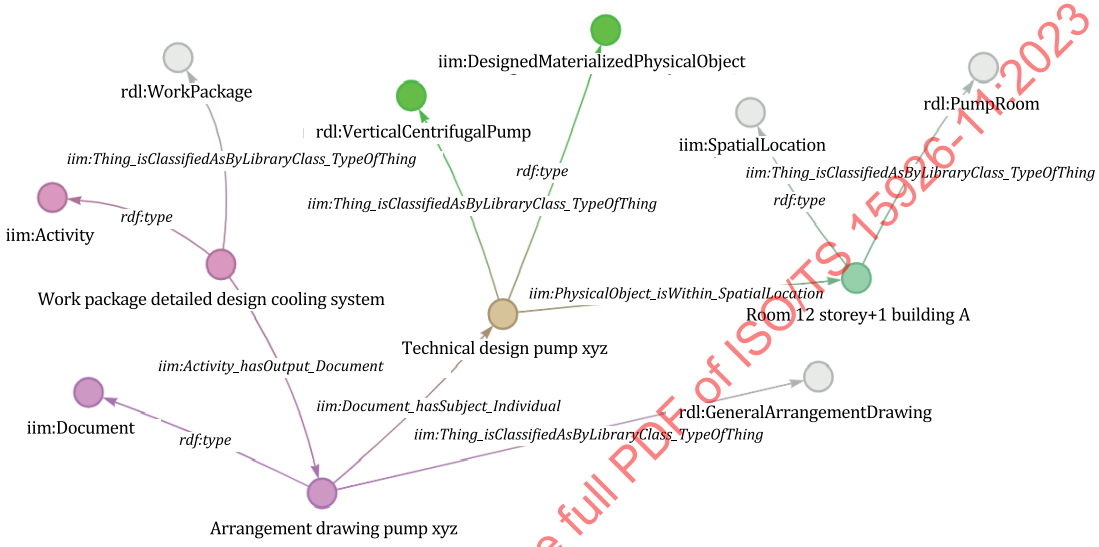


Figure 33 — Implementation of the model of Figure 32 in a graph database application

6.3 Modelling of requirements and verifications

In Figure 34 an instance of `requirement` (an individual requirement) is modelled using the approach in this document. Even though a requirement is a `class of relationship` in ISO 15926-2, in practice a requirement is seen as a specialization of an `information object`, therefore in this document it is configured as a `rdfs:subclassOf` an `information object`.

In this example the instance of requirement called “Arrangement requirement x pumps” is classified as a product requirement as defined in the RDL This requirement specifies the technical design of pump xyz. The requirement relationship ‘isSubjectIn’ joins to an instance of an `information object role and domain`, which carries the role of an evidence document within the domain of an arrangement drawing pump xyz from which one can verify that the requirement is fulfilled.

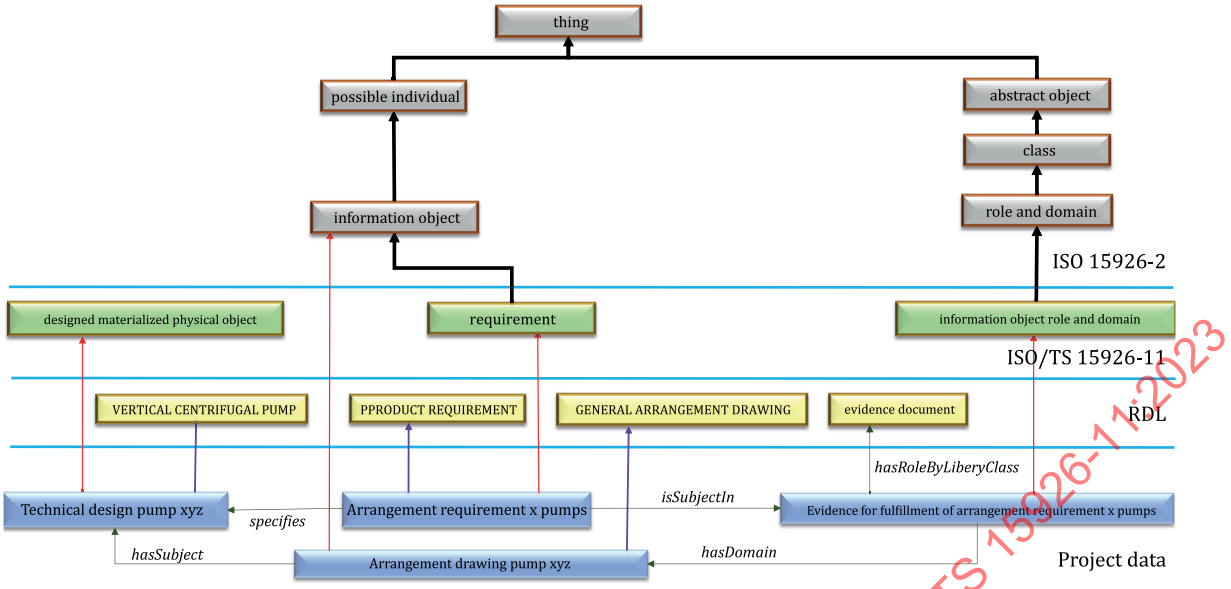


Figure 34 — Example of modelling a requirement and related evidence document

In Figure 34 the verification of a requirement is done implicitly. There is no metadata concerning the verification, which in some projects is sufficient.

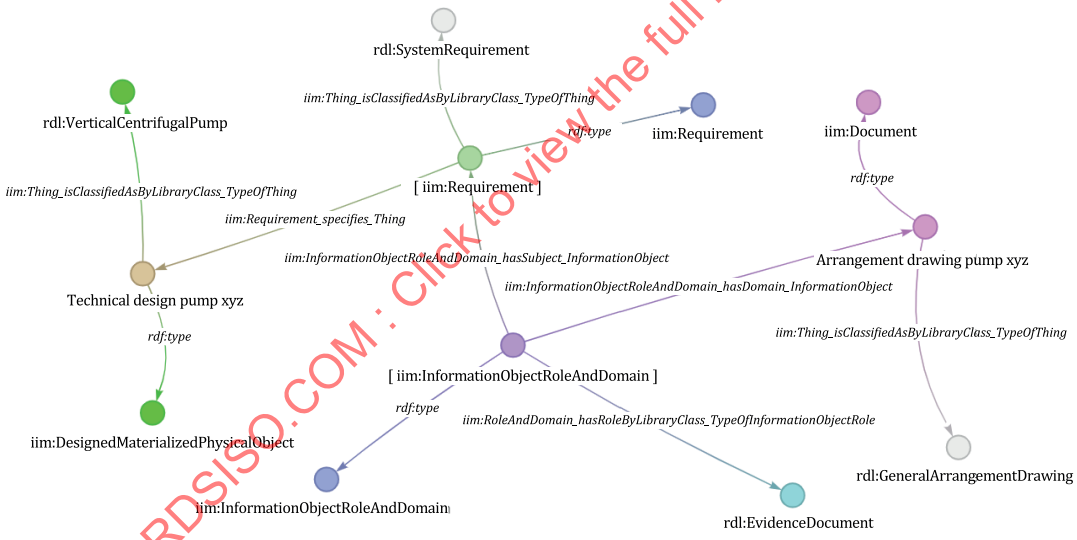


Figure 35 — Implementation of the model of Figure 34 in a graph database application

In Figure 36 the verification of the same requirement in Figure 34 is defined explicitly by introducing an instance of V&V `activity`. This activity is classified as a verification activity according to the RDL and applies a document review as the verification method.

During the verification activity, the requirement ‘arrangement requirement x pumps’ is performed with respect to the technical design of pump xyz. This verification activity is performed by a “person role and domain”, defined as an instance of `party role and domain`.

The entity “person role and domain” can be treated as a “blind node”, i.e. not having a name, only a unique identifier (UID) in the project data set.

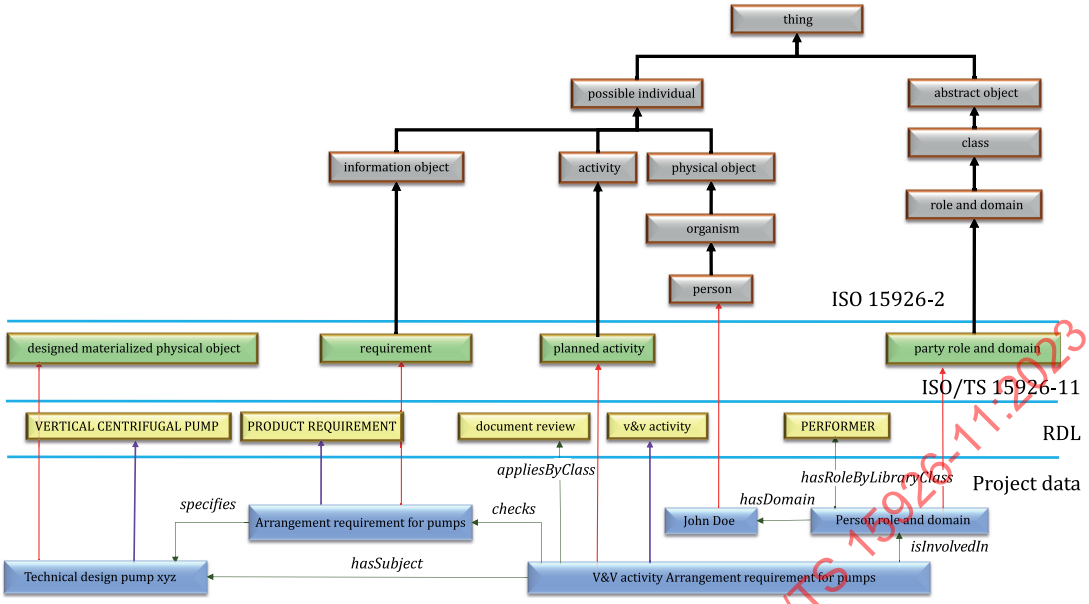


Figure 36 — Example of a planned verification of a requirement

Figure 37 — Implementation of the model of Figure 36 in a graph database application

In Figure 38 the verification activity of Figure 36 is further extended with additional relationships and metadata to capture the context of that verification activity as follows. An instance of `statement` is added that defines the scope of the verification of the specific requirement. In this case the statement declares that all individual pumps will be separately checked against the requirement rather than for an instance representing a quantity of pumps. The date the verification was completed is defined by a `date property` with as value a `point in time` (both as blank nodes). The date property is classified as a 'Date Completed' as defined in the RDL with as literal value 02-03-2018. The state of the verification activity is classified as 'compliant' (defined in the RDL). The verification activity has taken place ('performed') in the 'Basic Design Phase' as defined in the RDL. The verification activity can be followed up by a verification activity later in time, for instance with another result such as 'not compliant'.

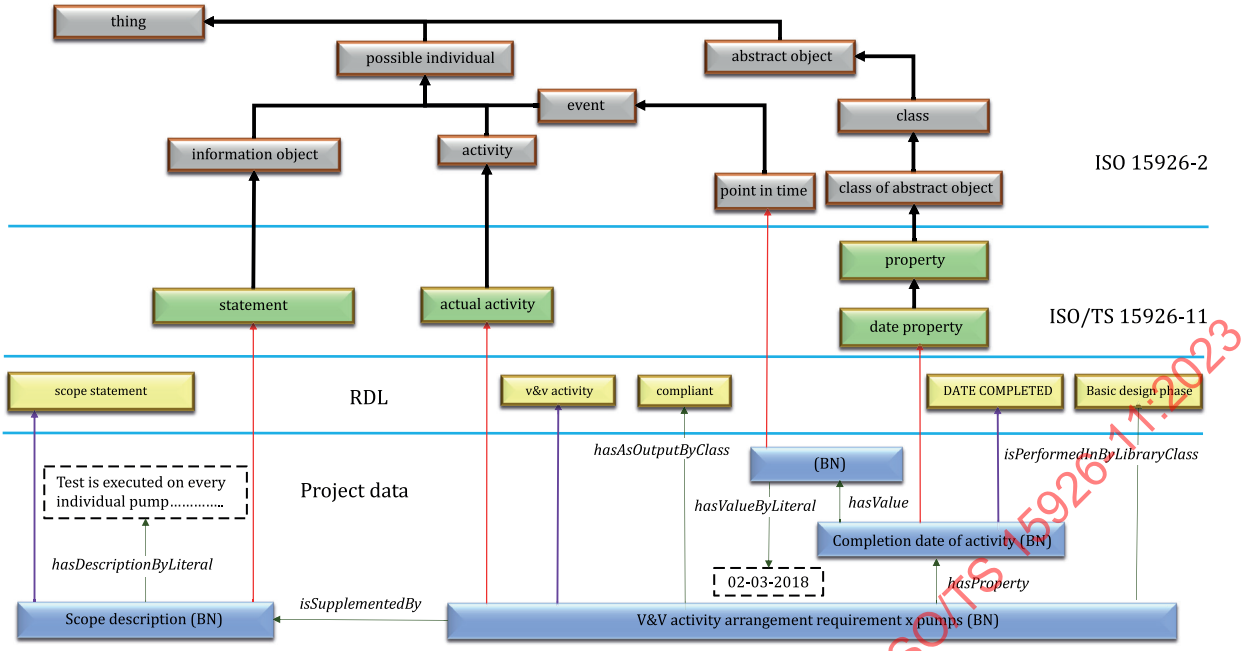


Figure 38 — Example of an actual verification of a requirement

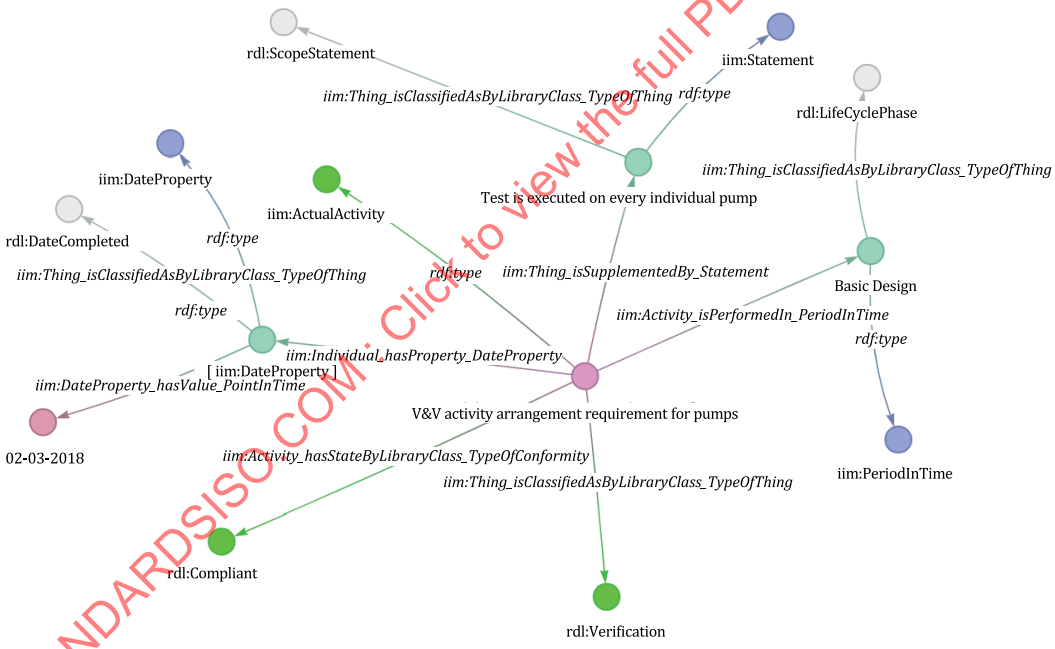


Figure 39 — Implementation of the model of Figure 38 in a graph database application

6.4 Modelling of properties

In Figure 40 an example is given of how a quantitative property is modelled as originally defined in Figures 9 and 15. It is sufficient to implement an instance of a `quantitative property` as a blank node since the context is clear: the individual object that possesses the property, and the classification of the `property` and the `quantity` is defined. The `quantitative property` is defined by a number value and a UoM. It can also be related to metadata like the origin that defines where the value was obtained from.

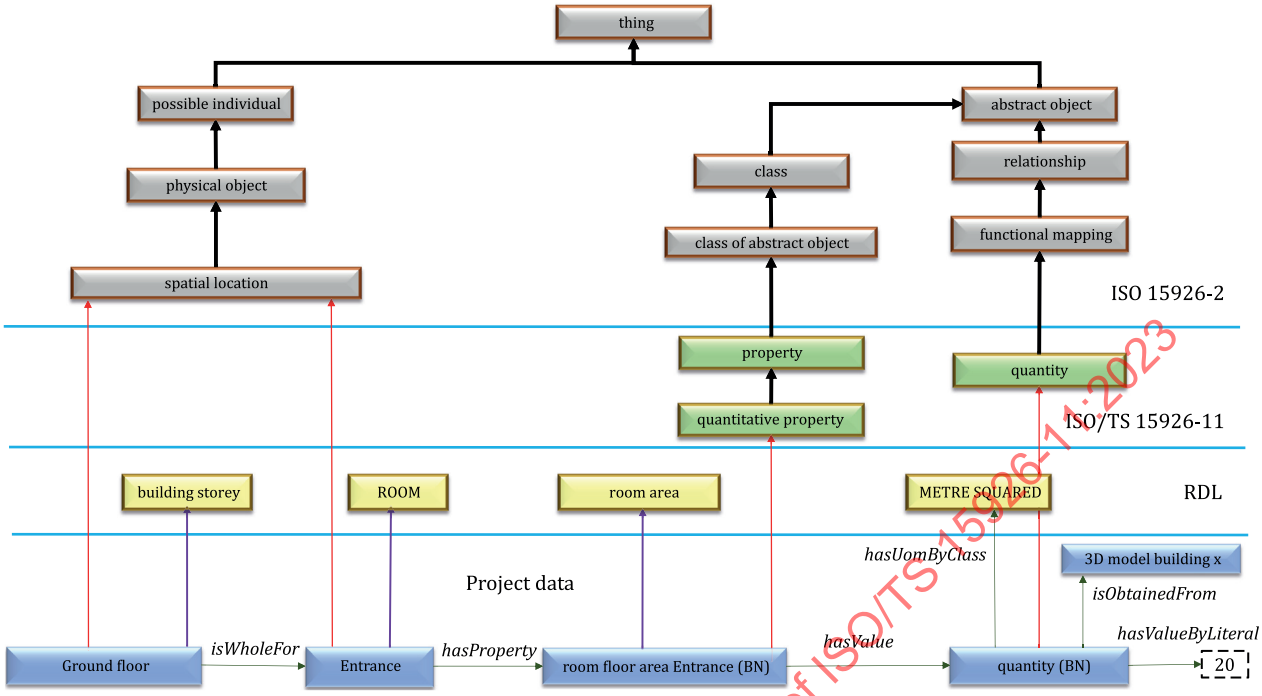


Figure 40 — Modelling pattern of a quantitative property

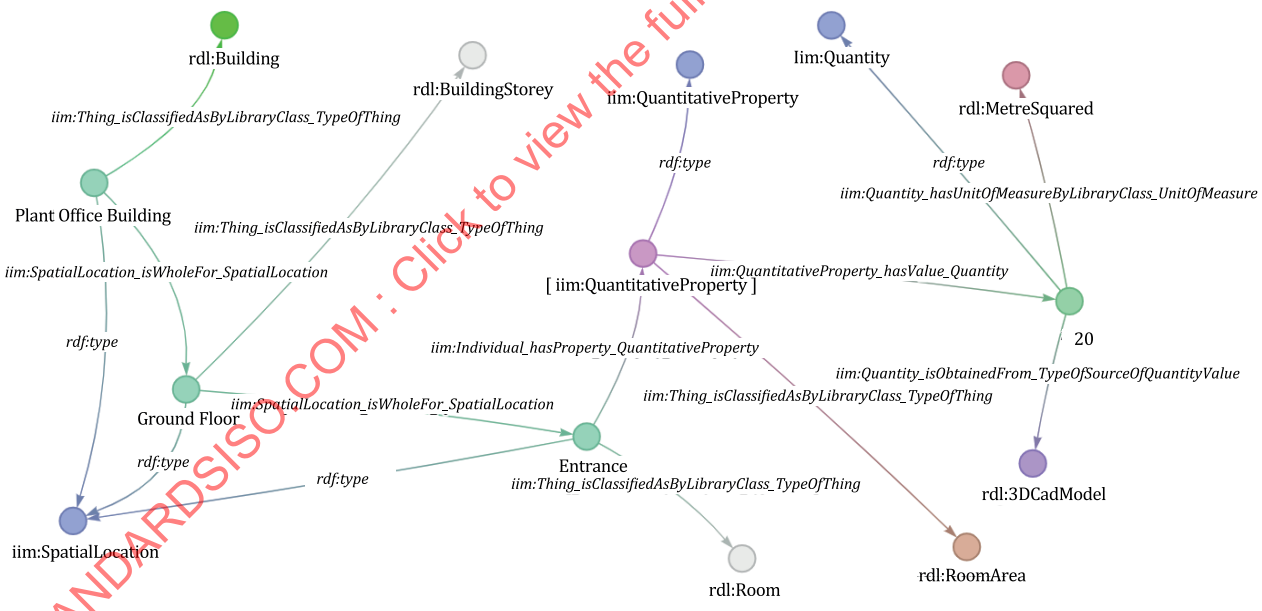


Figure 41 — Implementation of the model of Figure 40 in a graph database application

Figure 42 is an example of how a `qualitative property` is modelled as originally defined in Figures 9 and 15.

A `qualitative property` in this document can be used to assign a string value as characteristic to any instance of any `possible individual`. The string values are mostly general qualities that should be defined in the RDL. It is sufficient to implement an instance of a `qualitative property` as a blank node since the context is clear: the individual object that possesses the property, and the classification of the property and quantity are defined in the RDL. To have validation control over allowed qualities that an instance of `qualitative property` can point to, organize qualities into classes which can then be used as the range for a specific `qualitative property`.

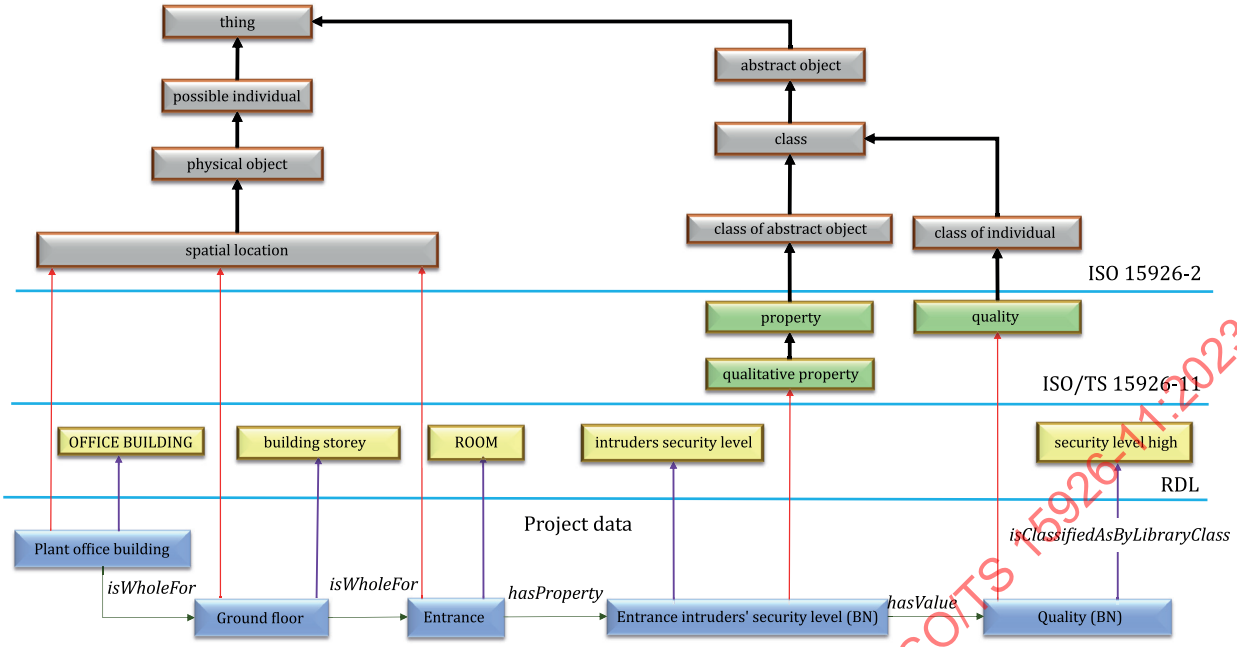


Figure 42 — Modelling pattern of a qualitative property

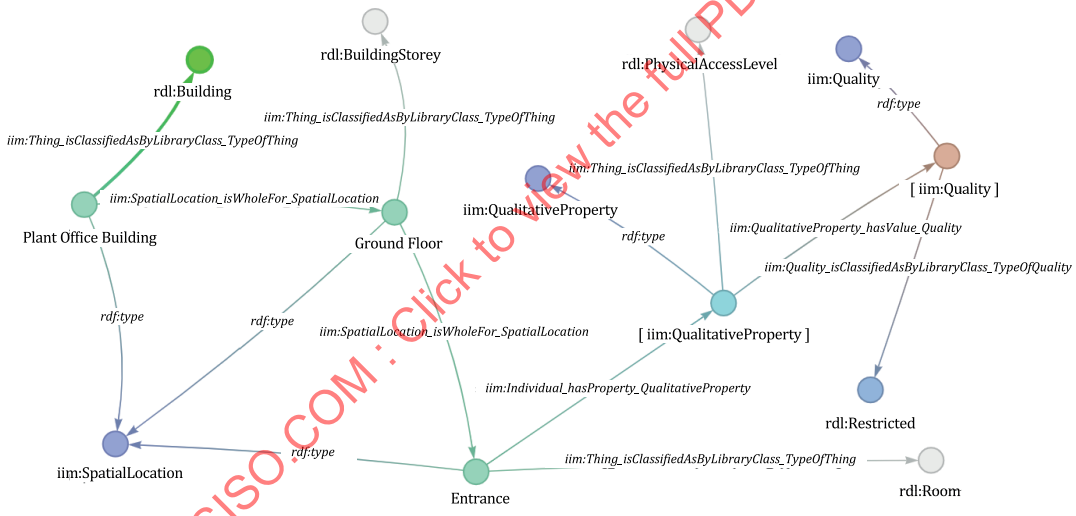


Figure 43 — Implementation of the model of Figure 42 in a graph database application

Quantity values of properties vary in practice over the life cycle. This is illustrated by Figure 44 which shows the volume flowrate of a pump in the life cycle where the pump is represented as a `designed functional physical object` and in the life cycle where the pump is represented as an `actual materialised physical object`. In this way the designed value of the pump derived from a process data sheet can be compared against the value of the installed pump derived from the instance of a site acceptance test (SAT). In this way automatic verification can be applied on life-cycle data.

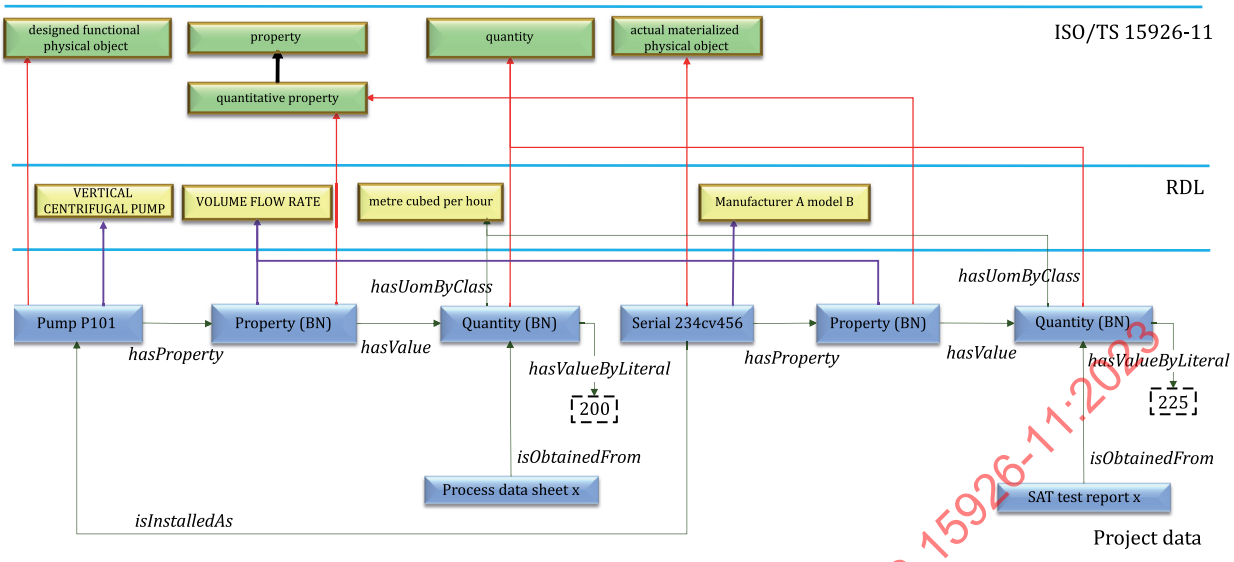


Figure 44 — Defining a specific property of an object within its life cycle

6.5 Modelling of states of individual

This document applies behaviours to systems based on the usage of states and transitions between states, driven by interactions or triggered by an output of a function driven by interactions. A system can be defined as 'A System is a set of elements in interaction.'

In Figure 45 a state of an actual reactor is modelled that represents operational data which can be integrated as configuration management data to support asset management.

In ISO 15926-2, states are treated as a status while in practice often a difference is being made between a state and the status. Therefore, `state of individual` is defined in the ontology in this document as a subclass of `part life individual`.

In this way any state of any instance of `possible individual` can be defined explicitly in this document.

A `state of individual` can be a blank node with a classification according to an RDL class (in this case hot full power). The start and begin events can be related to a point in time by instantiating `date property` and classifying these according to RDL classes.

In the same manner a state can also be defined by physical events, e.g. pushing a button and or a component system failure. This model pattern is also needed to describe behaviour of systems and systems elements

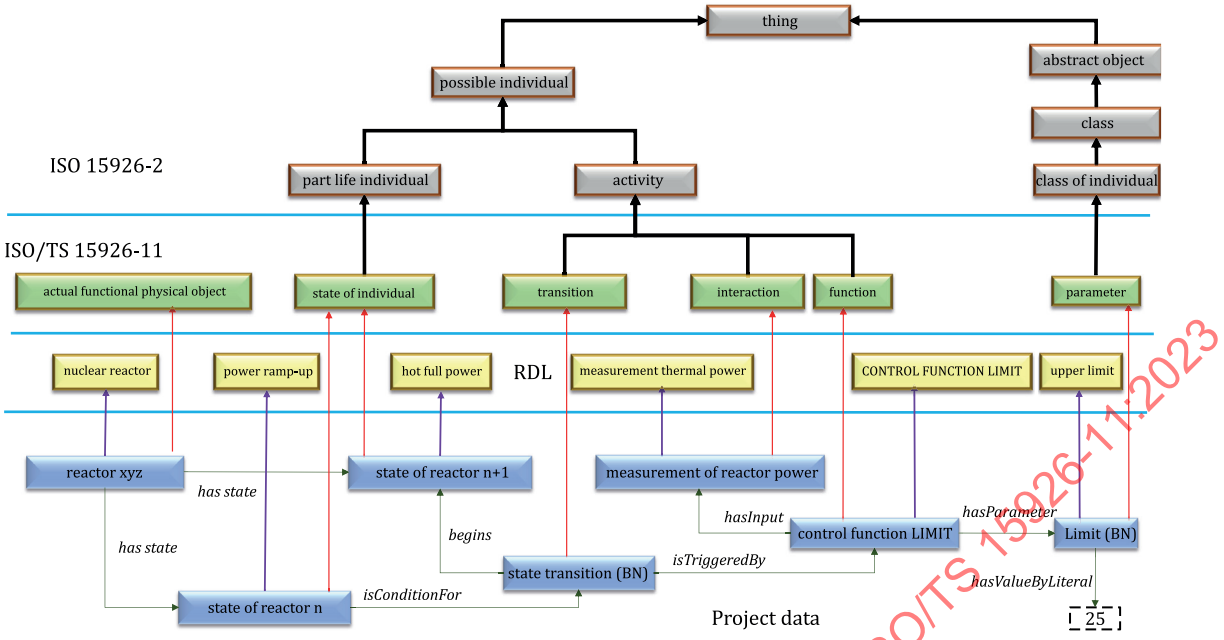


Figure 45 — Modelling of a state of a physical object

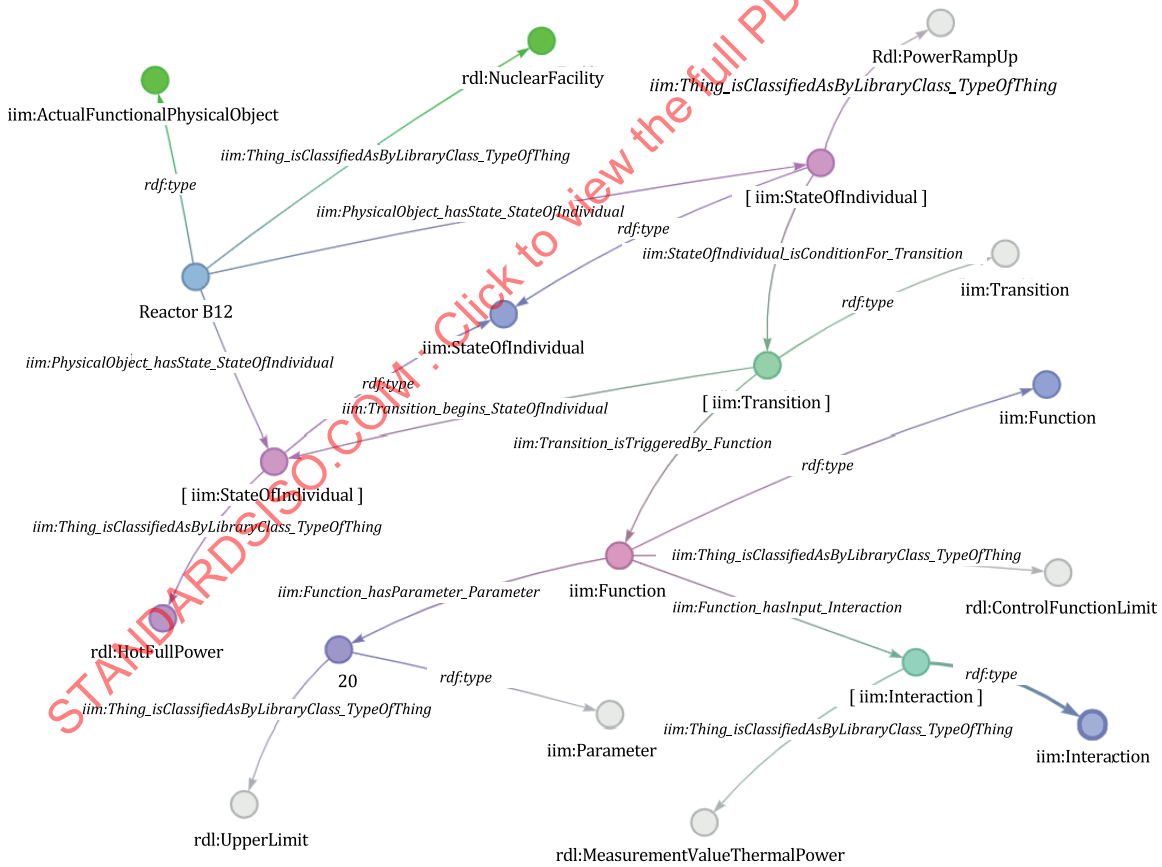


Figure 46 — Implementation of the model of Figure 45 in a graph database application

6.6 Modelling of interfaces and interactions

This document describes how to model interactions and interfaces explicitly to support the integration of both systems and systems of systems. For this purpose, `system` and `interface` are defined as a

subclass of `functional object`, and `interface port` as a subclass of `feature`. In Figure 47 applicable relationships show how these classes are related to each other.

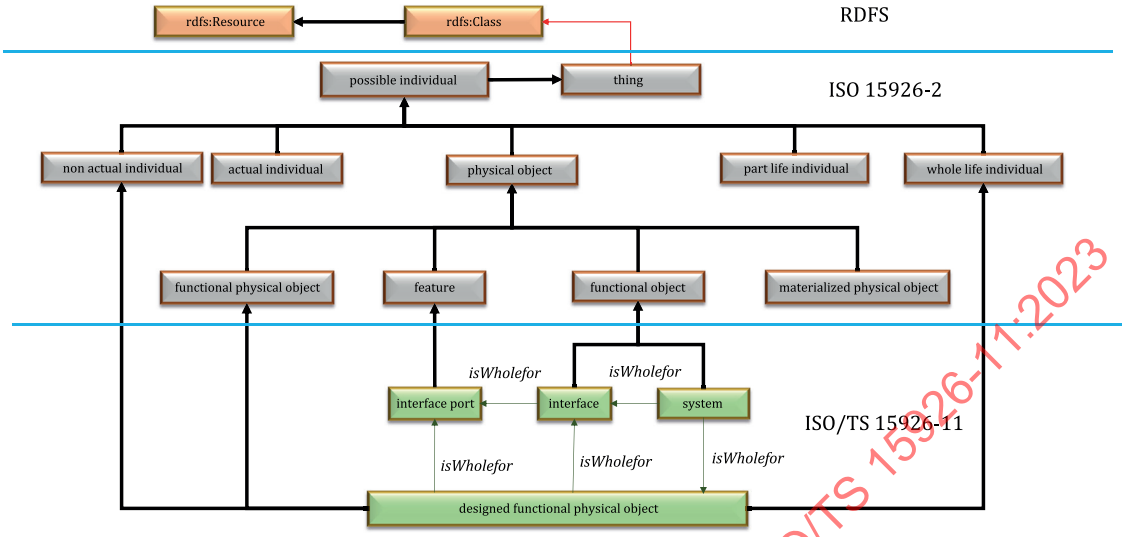


Figure 47 — Entities to model interface of systems in system elements

Interactions can occur across interfaces between the elements inside or outside the system and can be defined as exchanges of information (data), materials, forces, energy or the need for space. Interactions are supported by connections where the connecting points on the side of the system or system element are made by ports. Interactions result in a temporary or continuous flow of information, material, or energy between system elements within or between systems and or system elements. A `stream` is separate from interaction since the focus on a stream is more on the physical aspects of what is transferred from one place to another. It depends on the context whether one will model a `stream` or an `interaction`. This fragment of the ontology is shown in Figure 48.

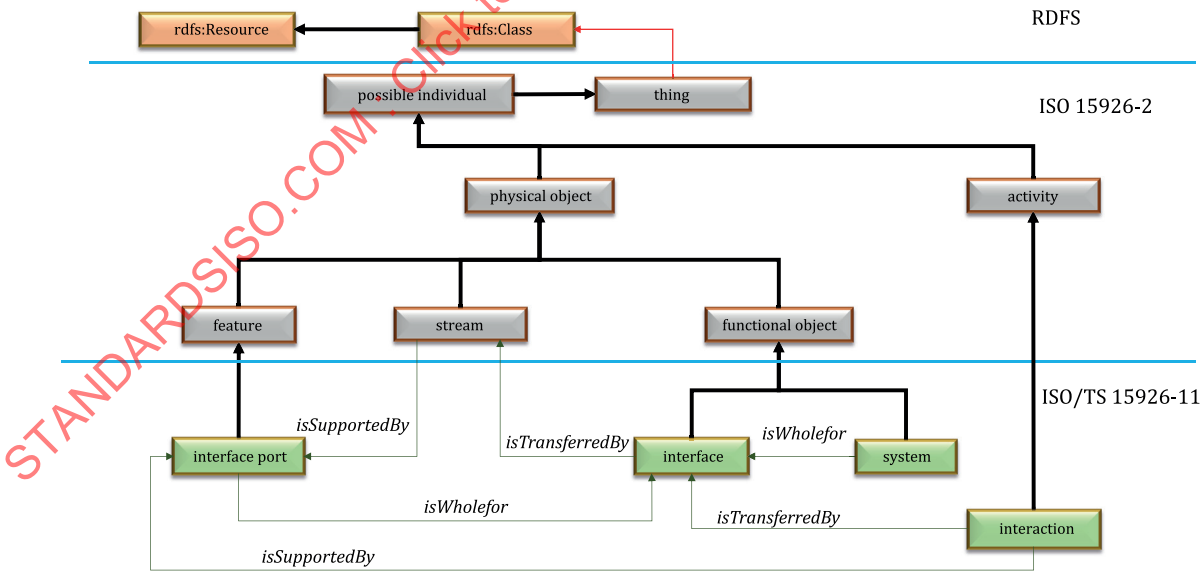


Figure 48 — Interactions supported or transferred by means of ports and interfaces

Figure 49 is an example of an `interaction` that represents the exchange of a measurement value between a measuring instrument and a control system. The instrument has a signal port feature that is part of a fieldbus interface which communicates via a network with the fieldbus interface of the control system. The specific signal of the measuring instrument is provided to the control system by a dedicated port in the fieldbus interface of the control system itself.

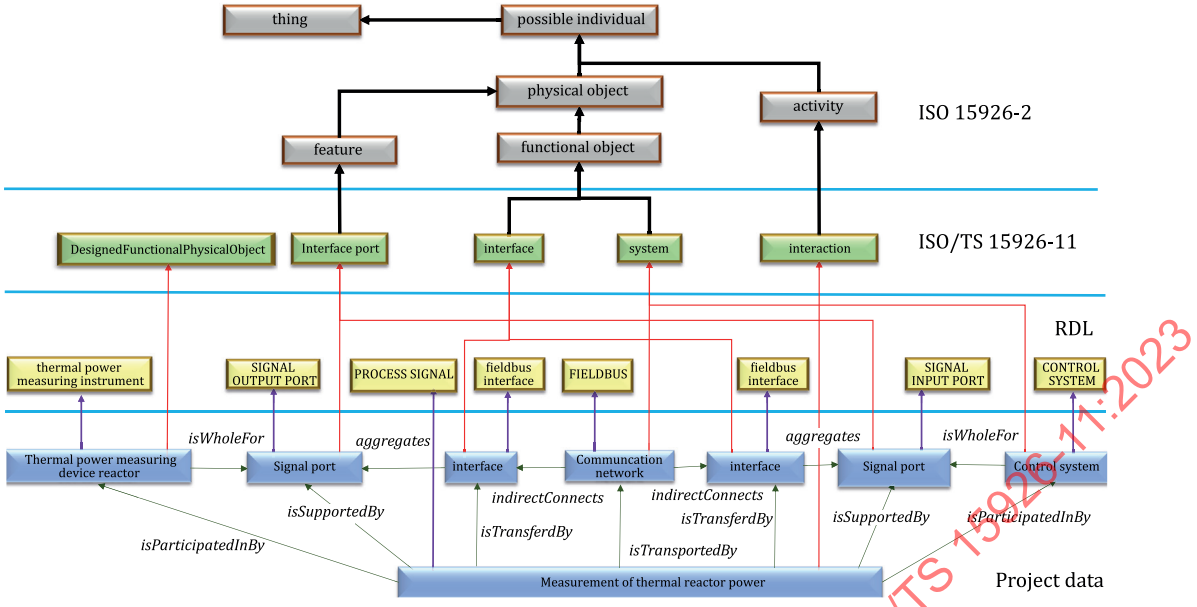


Figure 49 — Example of modelling an interaction being a measurement

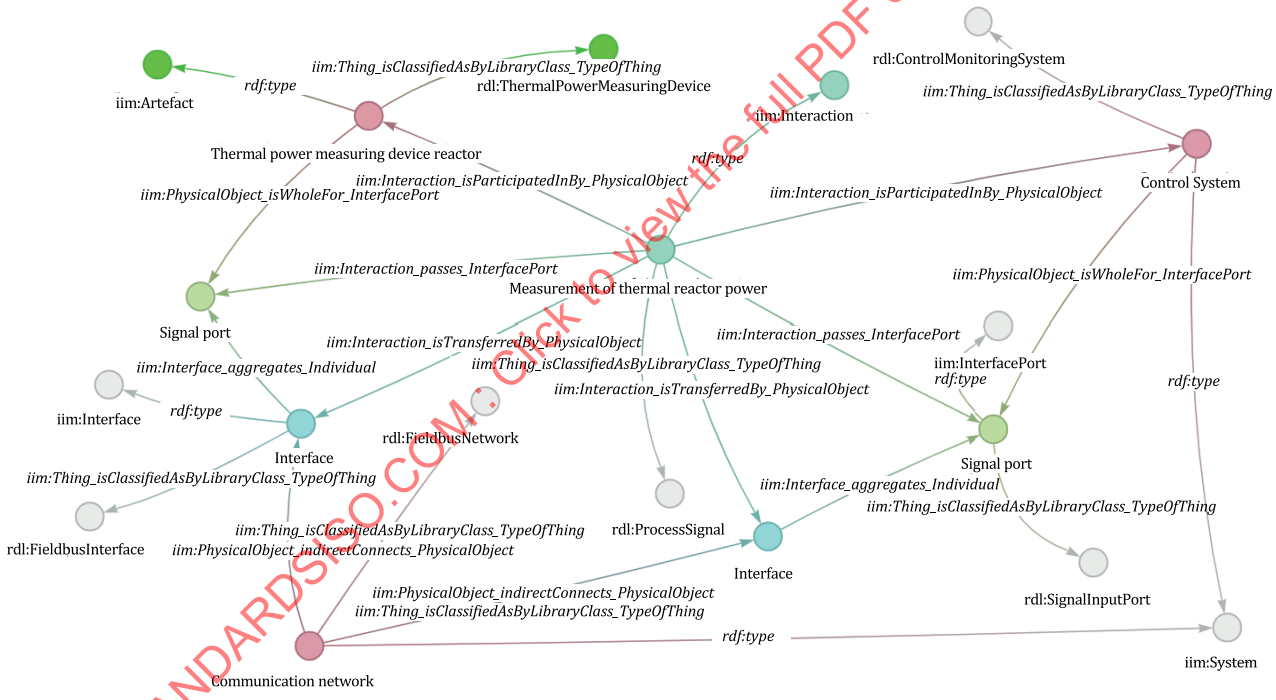


Figure 50 — Implementation of the model of Figure 49 in a graph database application

Figure 51 shows the modelling of a coolant stream as example of defining a `stream` as a counterpart to an `interaction`. The coolant stream is transferred by the interface between system A and system B which is shared as a common boundary between both systems. The coolant stream is supported by the fluid outlet port of the cooling pump of system A and supported by the fluid inlet port of the heat exchanger of system B. Both ports are indirectly connected (the way of connecting is not given) by the piping network of both system A and system B.

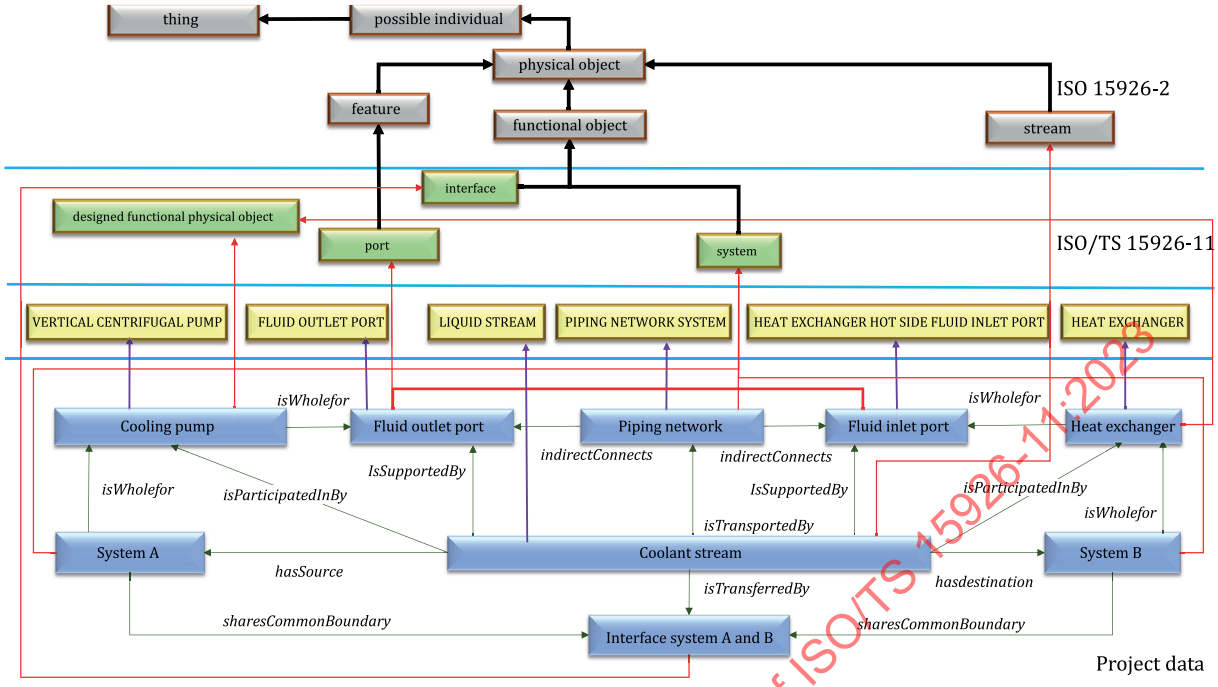


Figure 51 — Example of modelling a stream using ports and interface

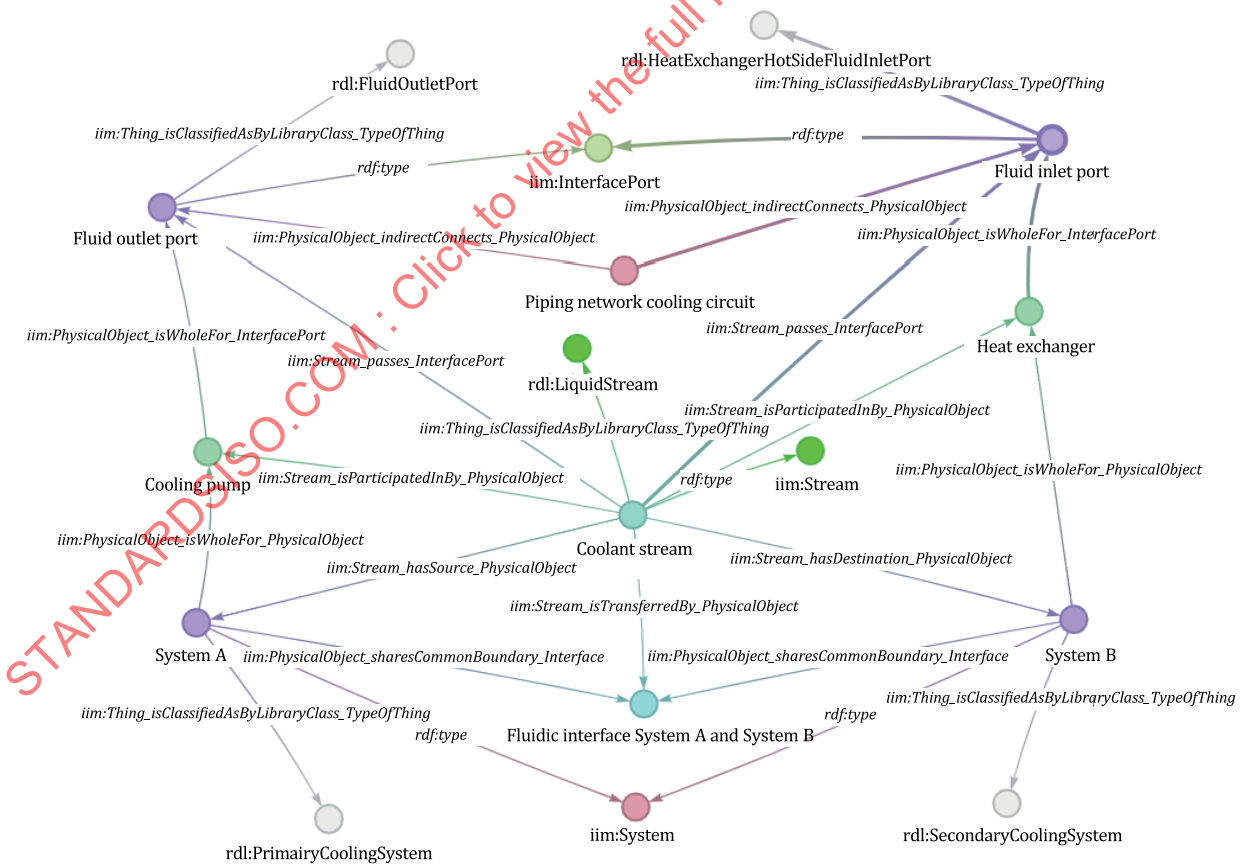


Figure 52 — Implementation of the model of Figure 51 in a graph database application

6.7 Modelling of risk information

Risk management requires control of information that characterizes a `risk` and information that comes with mitigating a specific `risk`.

The risk shown in Figure 53 concerns the risk of the cooling system not fitting within the corresponding building. This risk has as cause 'lack of integration", and the effect is defined as the redesign of the cooling system, the building or both. A `risk` is owned by a person or organization, in this case Company A.

The `risk` is qualified as high by means of a `qualitative property` which is directly classified the corresponding `quality` by the RDL class high.

The financial consequence is defined by means of a `quantitative property` with a `quantity` of 200 K Euro.

The risk is mitigated by using an integrated 3D model of the design. The work package representing the detailed design is involved in this risk.

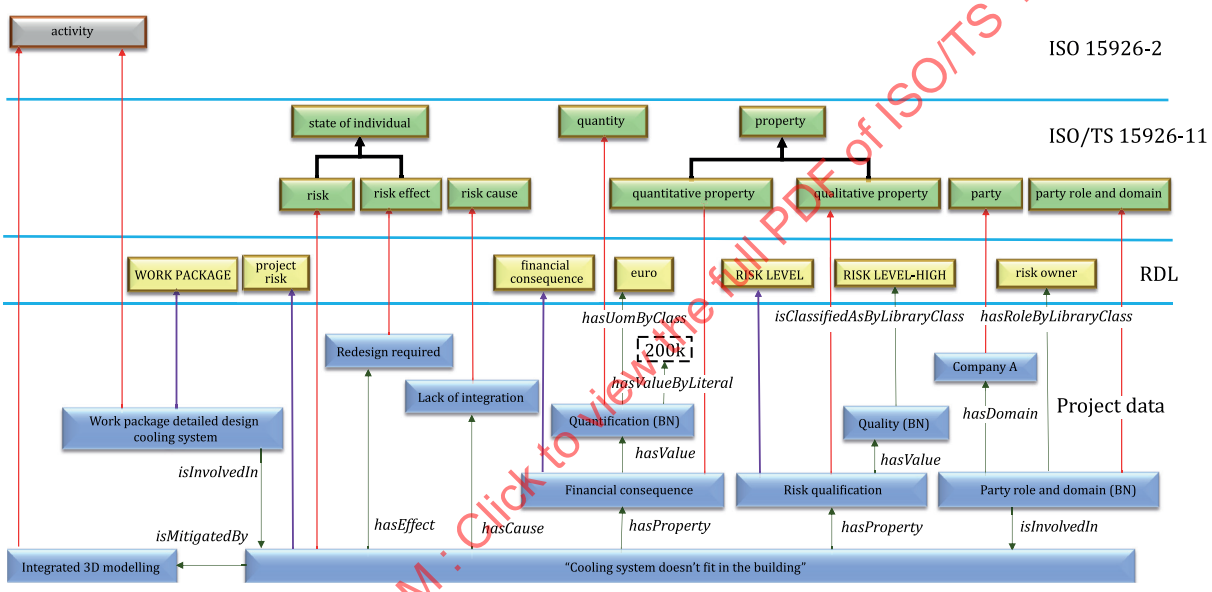


Figure 53 — Example of modelling a risk

Figure 54 — Implementation of the model of Figure 53 in a graph database application

6.8 Modelling of project change information

Figure 55 provides an example of how a project change can be modelled in the context of this document. The project change concerns "enlarging the cooling capacity" which affects both the corresponding building and the cooling system itself. This project change addresses the risk that the cooling system does not fit in the building. The project change includes the work package that enlarges the design of the cooling system. There is a `qualified property` defined representing a time consequence of more than six months. There is a `quantitative property` defined representing a financial cost of 500 K Euro. Company B is involved in this project change as a stakeholder (using the `PartyRoleAndDomain` construct).

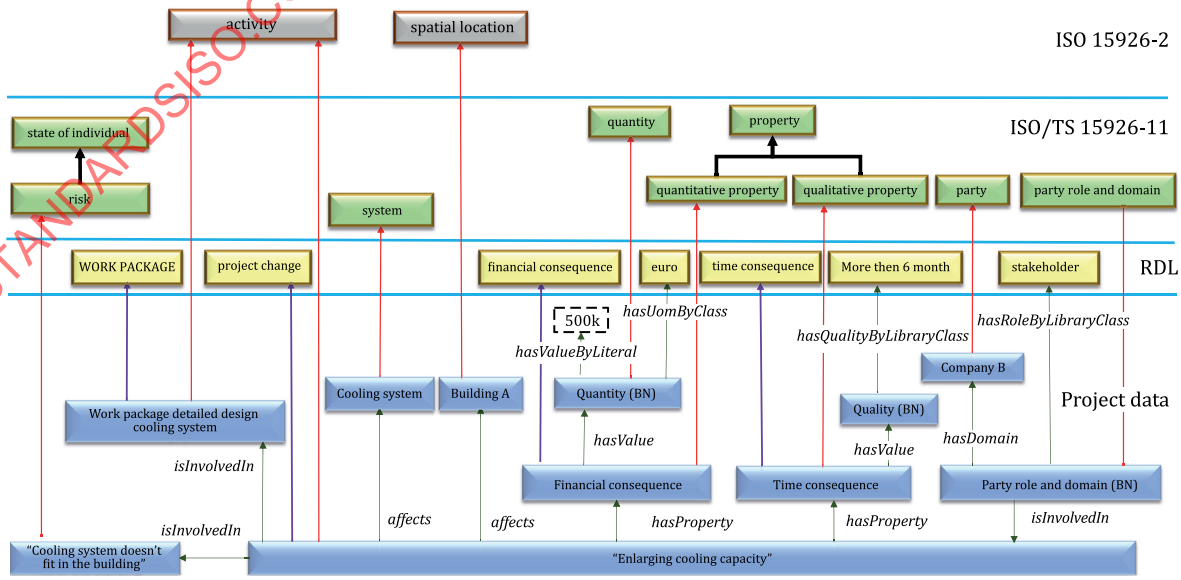


Figure 55 — Example of modelling a project change

Figure 56 — Implementation of the model of Figure 55 in a graph database application

6.9 Modelling of failure mode and effect analysis information

The methodology presented in this document can also be applied to failure mode and effect analysis information. An example of the modelling of a (limited) failure mode is shown in Figure 57.

A failure of the cooling pump is an event that in Figure 57 is caused by lack of lubrication of bearings. In Figure 57 the failure is characterized by the value of a duration property 'mean time to repair' (MTTR) of 8 h.

The failure cause is defined as an instance of a state and classified by the RDL class as such. The failure cause leads to a `failure mode`, which is defined and observable when the flow becomes less than 10 m3/h.

The effect of this failure mode is defined as an unexpected shutdown of the reactor.

The failure has an effect on the shut-down of the reactor and is the beginning of the state of the reactor 'loss of cooling capacity'. The shut-down state is characterized by a `duration property` 'mean down time reactor' with a value `duration` which is made explicit by means of a literal value of 24 and a unit of measure 'hour'.

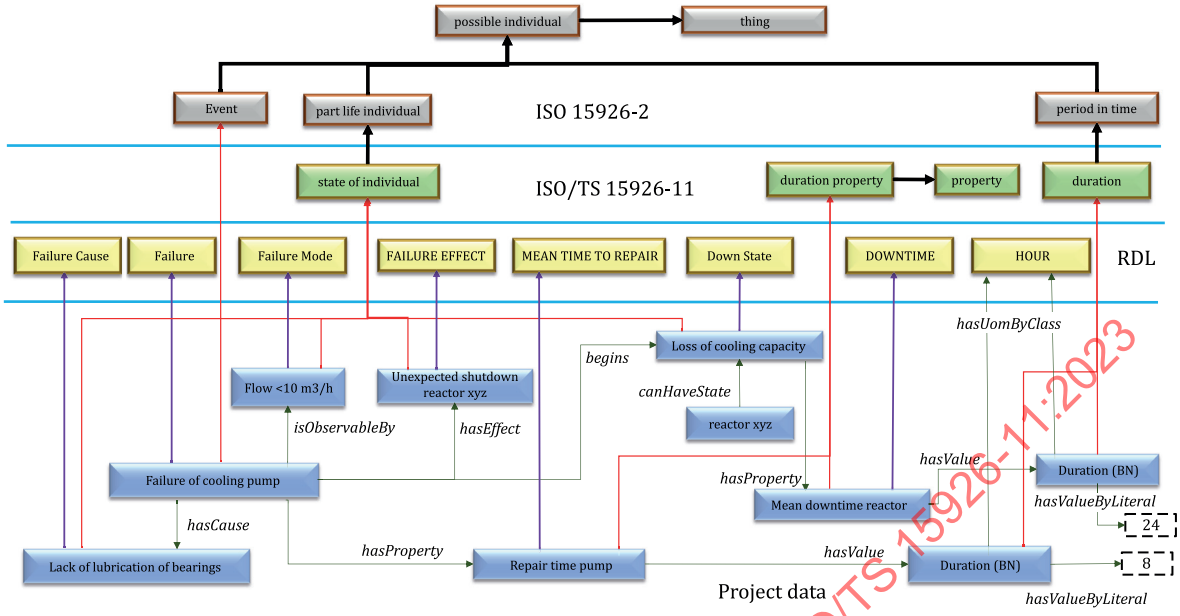


Figure 57 — Example of modelling failure mode and effect analysis (FMEA) information

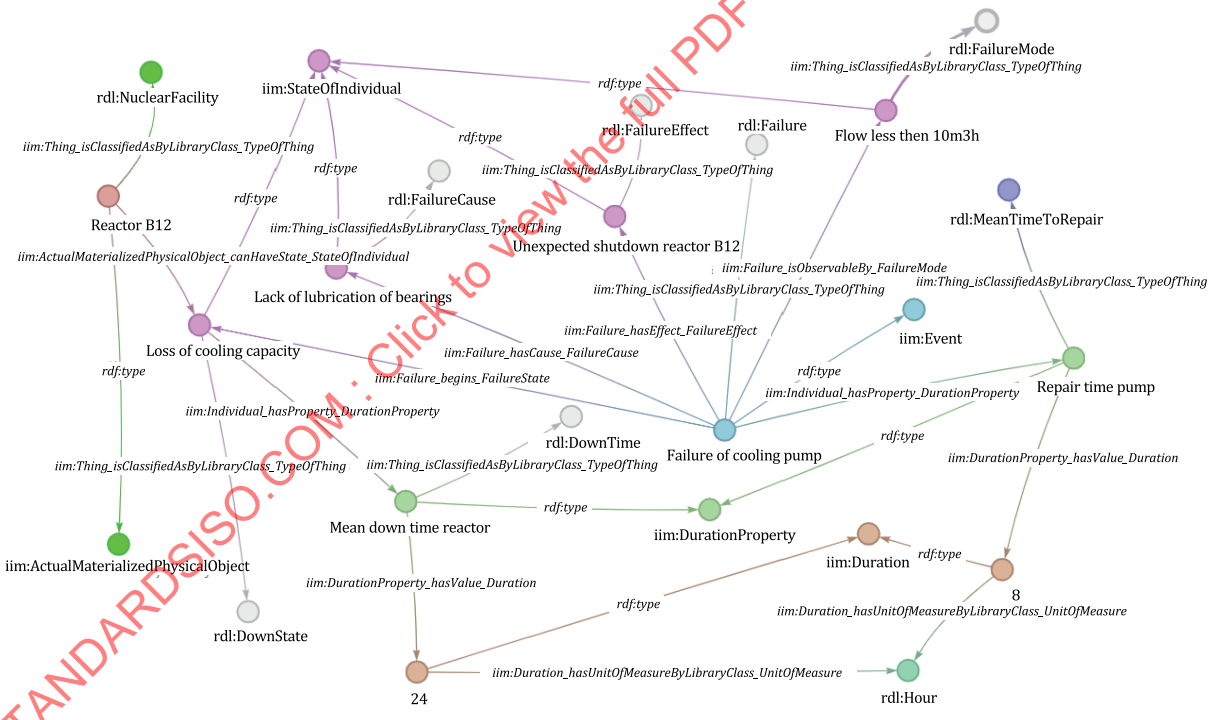


Figure 58 — Implementation of the model of Figure 57 in a graph database application

7 Integration and exchange of project data based on statements

7.1 Concept of a common data environment (CDE)

A possible implementation of this document in a concrete project is explained in this clause on the basis of a data-driven CDE, rather than a document-driven CDE as described in the ISO 19650 series. In the data-driven approach, between the multidisciplinary engineering tool environment and consuming entities of integrated plant data, a CDE is placed as shown in Figure 59. In a typical multidisciplinary engineering tool environment, in general, the engineering tools all have their own core model and data

export functionality and use a tool-specific set of terms and definitions of these terms. This leads to multiple data streams each having their own structure, syntax and semantics to be interpreted by the CDE. Therefore, all data streams are processed by a cleaning process in which a seamless integration of all data streams is realized by harmonizing all data stream content based on a common language defined by the IIM and RDL. When processing these data streams to integrate the data in the CDE, data quality principles are applied as defined in ISO 8000-1. The main data quality characteristics are defined by ISO 8000-8 as syntax quality, semantic quality, and pragmatic quality (usability of the data). Figure 59 shows the principle of receiving raw, baselined data sets from the engineering environment passing through the cleaning process. In this cleaning process, syntax of used coding identifiers is checked and eventually corrected, the meaning of the data (semantics) is interpreted and aligned with the origin of the data and all data is classified according the RDL. All successfully cleaned and imported data files are archived in the CDE for governance and traceability, supporting configuration management.

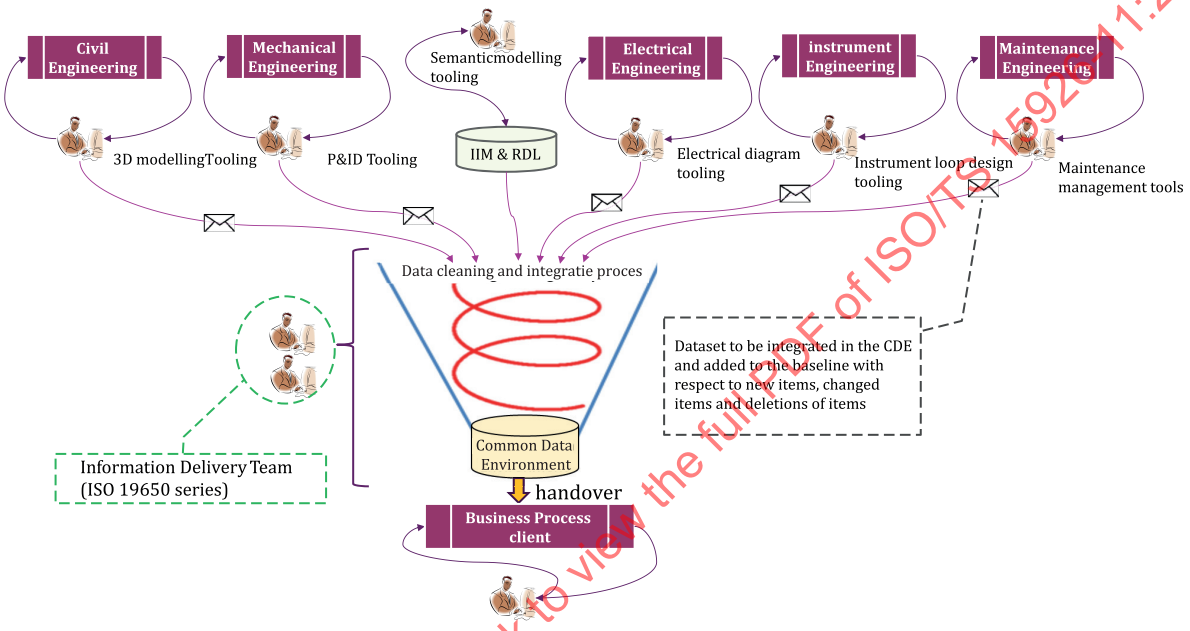


Figure 59 — Challenge to integrate data in a CDE obtained from various tools by which disciplines manage their work in progress

Figure 60 is an example shown of typically dynamic nature of project data resulting in additions, changes and deletions within the CDE. The guiding principle is that all deletions, changes or additions will be traceable in the CDE by means of adding signatures to all imported data sets that are imported into the CDE.

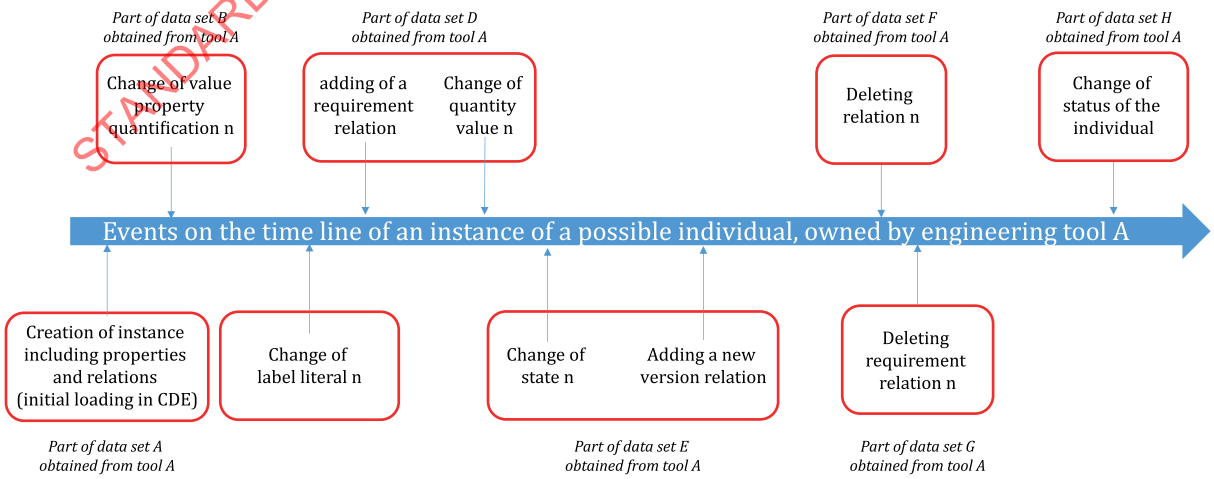


Figure 60 — Example of events on the time line of a possible individual

In this document a data set is seen as a coherent set of statements which have a common source and origin, and a common release date (defined by metadata), which are all collected in a signed named graph. Each signed named graph has a set of metadata for governance and traceability. The signature statements of a signed named graph are collected in a signing named graph. There can be more than one signing named graph related to a signed named graph, e.g. one for the cleaning and uploading activity and one for the validation activity of the signed named graph. A signing named graph holds, e.g. information about the performer and sign-off date of the activity with respect to the signed named graph. The metadata of both the signed named graph and signing named graph assures governance and traceability of history needed for configuration management. In summary, three mechanisms for configuration management are recognized:

— the use of reification of triples by means of rdf:statement to be able to make statement about statements (for replacement of statements and assigning a status to statements);

— the use of a signed named graph (with metadata) for holding a coherent set of statements together, representing an identified, individual data set as imported in the CDE;

— the use of a signing named graph, holding the signature of the party that has executed a task in relationship to a signed named graph.

A common principle in configuration management is the use of baselines, which represent the state of an instance or collection of instances in the design data at a certain point in time in the project. This enables the generation of the difference between two baselines in terms of changes, additions or deletions in the baselined data as represented in a signed named graph.

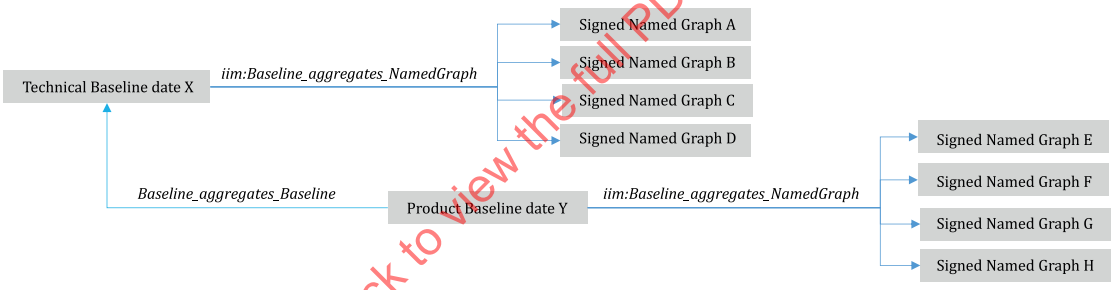


Figure 61 — Example of the sequence of two baselines composed of signed named graph

In Figure 61 the definitions of two baselines (e.g. the technical baseline and product baseline) are given by collecting signed named graphs. The second baseline (product baseline) inherits all signed named graphs of the first submission so the second is a cumulative collection of all previous signed named graphs. The results represent the actual state of a design, so all statements that are contained in one of the signed named graphs A to H. This means that all statements that have a statement that are classified as deleted are actually ignored and replaced by other statements. Only the statements that are not replaced by another statement stay valid.

Figure 62 expands the principle of building baselines as per Figure 61 with signing named graphs and the recognition of additions, deletions and changes with respect to the first loading of a data set obtained from, e.g. an engineering tool.

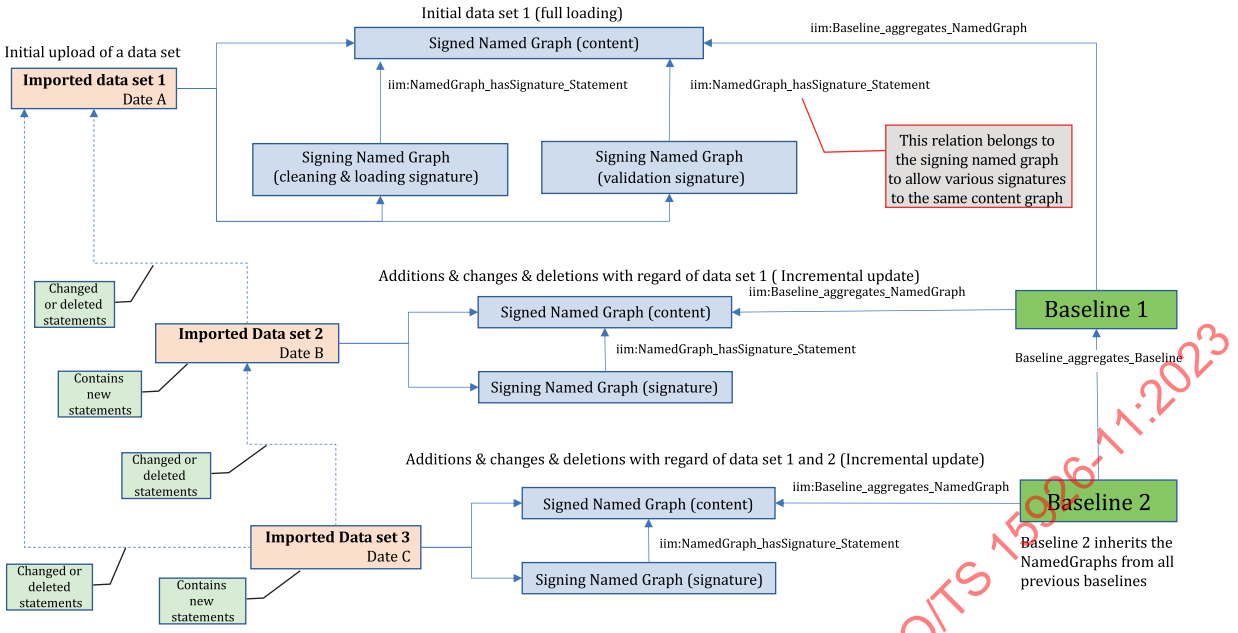


Figure 62 — Typical development of engineering content with respect to additions, deletions and changes, consolidated in baselines

To implement this functionality, use the signature mechanism to explicitly identify every new statement belonging to the total set of project data. None of the data will be permanently deleted, but will only be marked as deleted by means of a separate statement. To account for literal changes the dcterms: replaces as a forth triple in the statement with the new value will explicitly state the statement with the old value.

A statement that has not changed keeps its signature (via the signing named graph) from the very first import (initial loading), even when the same statement occurs in subsequent imports.

Named graphs that belong to the same subject/domain/view can be collected in an 'Aspect Model' that serves a purpose, such as a system break down structure or, e.g. geographical breakdown structure (GBS).

A typical workflow transforming 'raw data sets' from any engineering tool into a cleaned and harmonized import sheet into the CDE is shown in Figure 63.

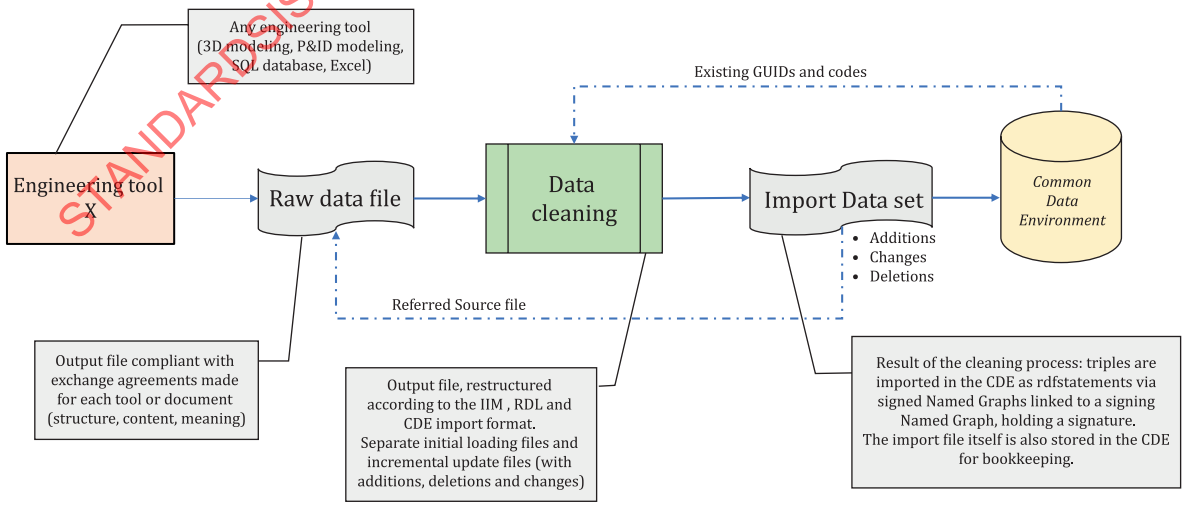


Figure 63 — Typical workflow of processing data files obtained from engineering tools

Figure 64 is an example of metadata attached to a signed named graph to characterize that named graph, e.g. security classification, status, owner and the source the data is obtained from. The relationship that points to the signature is captured in the signing named graph since there can be more than one signature attached to a signed named graph (e.g. one for loading and one for validation of the named graph). The principle is that a named graph will never be updated, e.g. by adding an extra signature: new data are always introduced in a new named graph.

The set of metadata is user definable by selecting relationships from the initial set or by making specializations of ones in the initial set. This should be defined before start of the project and can be captured in an IDM.

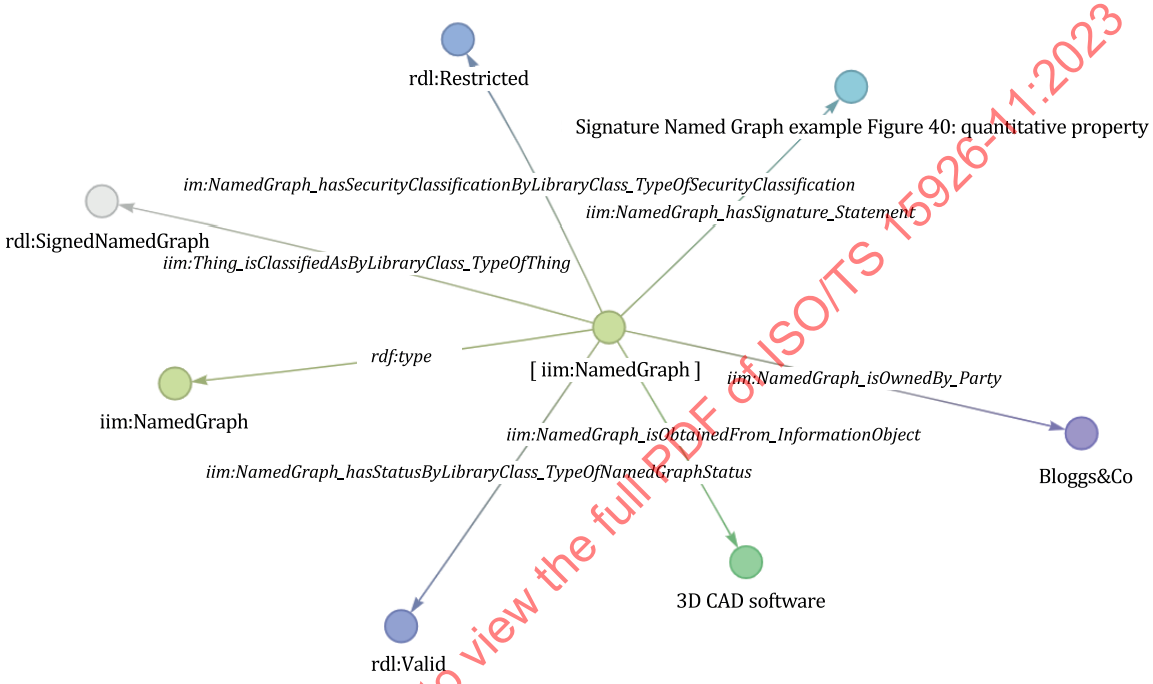


Figure 64 — Metadata describing a signed named graph and relationship to its signature

Figure 65 is an example of metadata defining a signing named graph and holding the signature statement that signs one or more signed named graph. This is, e.g. the case as a data set is holding several document versions which all are stored in separate signed named graphs each referring to the same signature since all named graphs are the result of all and the same cleaning and uploading activity. This is also the reason that the computer file reference is part of the signature. The principle is that every successfully uploaded import file in the CDE is also stored in a file system of the CDE for history and archiving purposes. Consequently, if this is done, the whole content of the CDE can be restored by importing all stored import files in the correct sequence.

A specific relationship is defined in Figure 65 stating that the signature statement concerns a specific configuration management activity (defined in the RDL) such as “initial loading” and update data”.

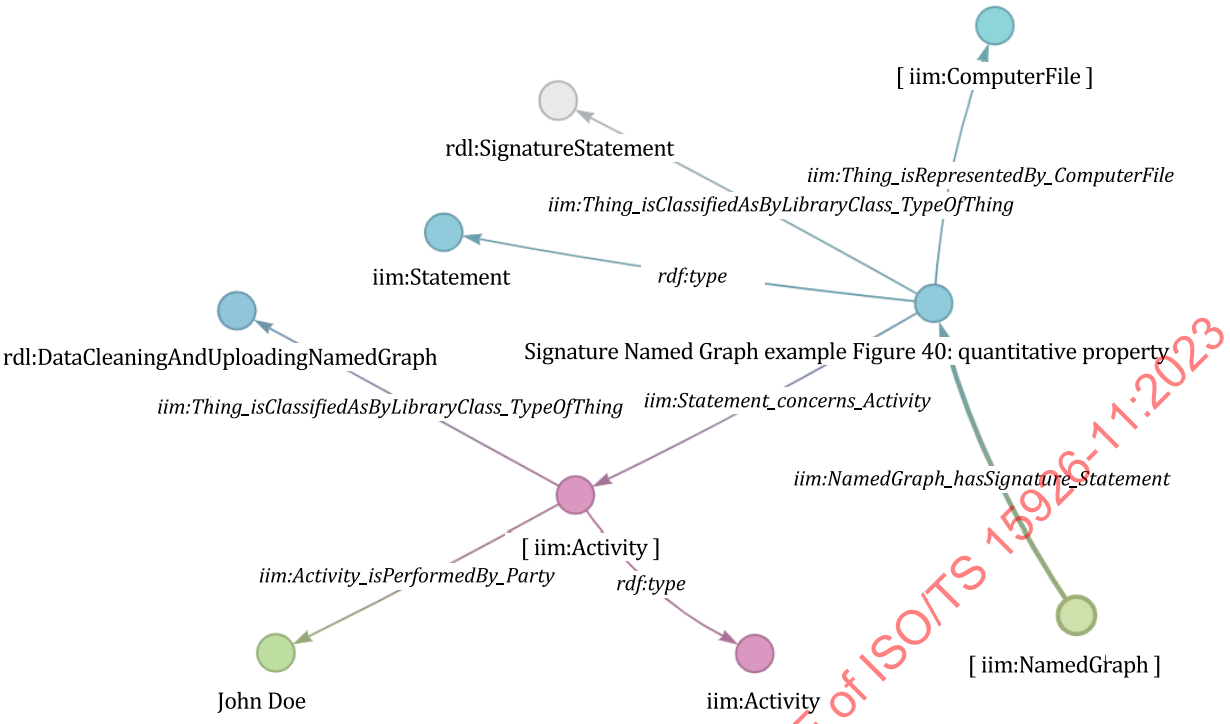


Figure 65 — Metadata of the signing named graph holding the signature of Figure 64

7.2 Reification of statements combined with named graphs

Since this document relies on both reified triples by means of rdf:statement and on collecting these statements consequently and consistent in named graphs, in this document TriG has been chosen as serialization mechanism. TriG allows an RDF dataset to be completely written in a compact and natural text form, with abbreviations for common usage patterns and datatypes. TriG is an extension of the Turtle format. TriG is intended to define an RDF syntax using multiple named graphs.

As stated in 7.1, the CDE, based on this document, supports additions, changes (by means of explicit replace relationships between statements), and soft deletions of data (by setting the status of a statement to deleted) to manage baselines in configuration and project data and reporting of differences between any baseline and another baseline (Figure 60). The ability to make a "statement about statements" in this document is accomplished by reification of the triples by means of rdf:statement. Each statement (holding a triple) is identified by a UID Uniform Resource Identifier (URI). This functionality relies on the rdf:statement mechanism to be able to explicitly identify every new statement belonging to the total set of project data. None of the data will be permanently deleted, but will only be marked as deleted by means of a specific statement. To account for changes in literals the dcterms:replaces function will explicitly state the old value and the new one as a forth triple in the corresponding rdf:statement.

In order to illustrate the complexity of configuration management, Figure 59 shows a sequence of examples of changes (events) on the time line of an instance of a possible individual (starting from left to right). Configuration management requires governance and the ability to trace the history of statements over time. In industry, parties will exchange statements in order to explicitly communicate configuration management information. These "statements about statements" support the workflow of processing industrial data along the supply chain and the life cycle. They become important when exchanging life-cycle data of a facility or plant.

The combination of RDF named graphs and rdf:statement offers a relatively simple and flexible mechanism to model products and processes and to express SE information during the whole life cycle. Both concepts can be queried using SPARQL, a W3C standard as well. Specifically the combination of named graph and SPARQL is quite powerful. By means of the SPARQL query language, all kinds of

questions can be asked to the CDE across about all imported named graphs. This can be utilised to create reports from the CDE for all kinds of engineering, verification and reporting purposes.

To illustrate how TriG can be used to capture a coherent set of serialized triples (by means of rdf:statement) as a result of instantiation of (part of) the ontology in this document, several types of named graphs are presented using the TriG serialization format below.

The named graph below is a signed named graph with the full content marked by '{' as start of the content of the named Graph and '}' as end of the content of the named graph. The example starts with classification and metadata of the named graph itself. After these statements are the triples, each represented by a rdf:statement, that form the model shown in Figure 41, which gives an example of a quantitative property (RoomArea) of a room on the ground floor of a plant office building. The value of the RoomArea property, by means of a quantity, is 20 m^2 which was obtained from the 3D CAD model.

The predicates in the statements are relationships that are part of the ontology of this document as defined in Annex A.

```
:136ee466-0107-4808-bd5b-30309f4b16ba {
        :24f611be-a68a-47cb-babd-cf65816b743a a rdf:Statement ;
                rdf:subject :136ee466-0107-4808-bd5b-30309f4b16ba ;
                rdf:predicate iim:Thing_isClassifiedAsByLibraryClass_TypeOfThing ;
                rdf:object rdl:SignedNamedGraph .
        :584a6f8b-18b4-4cc0-b93f-f512b7e91a06 a rdf:Statement ;
                rdf:subject :136ee466-0107-4808-bd5b-30309f4b16ba ;
                rdf:predicate rdf:type ;
                rdf:object iim:NamedGraph .
        :83210f9d-031b-4ed4-92ac-17d6ca8f617b a rdf:Statement ;
                rdf:subject :136ee466-0107-4808-bd5b-30309f4b16ba ;
                rdf:predicate iim:NamedGraph_isOwnedBy_Party ;
                rdf:object :95204dbc-77e6-4615-98f0-478898281c18 .
                # This object GUID represents a party defined in the CDE.
        :2df72fa4-086e-44c6-9335-ca172eb5f021 a rdf:Statement ;
                rdf:subject :136ee466-0107-4808-bd5b-30309f4b16ba ;
                rdf:predicate iim:NamedGraph_hasSecurityClassificationByLibraryClass_
                TypeOfSecurityClassification ;
                rdf:object rdl:Restricted .
        :213b9f57-579d-4b9b-9cfe-503b9e391859 a rdf:Statement ;
                rdf:subject :136ee466-0107-4808-bd5b-30309f4b16ba ;
                rdf:predicate iim:NamedGraph_hasStatusByLibraryClass_TypeOfNamed
                GraphStatus ;
                rdf:object rdl:Valid .
        :e37fbb47-4c8a-4443-b541-e2aaaed552f7 a rdf:Statement ;
```