Title:

The Facility Asset Information Modelling Framework Manual

Project:

Spine Project

Classification: **Internal**	Owner: **Magnus Oanes Knædal**
Expiry date: when Iteration 2 is issued	Status: Revision 1

Author(s)/Source(s):

Erlend Fjøsna, Torleif Saltvedt, Arild Waaler, Martin Georg Skjæveland, Rustam Mehmandarov, Magnus Knædal

Mission:

Develop documentation and specifications that together answer what a company must do to implement and use IMF. Frame and describe the IMF scope and relevant interfaces. Detailed description of all elements part of the IMF language. The result shall form the necessary basis to create a DNV Recommended Practice.

Remarks:

First draft.

Valid from: **01.07.2022**	Updated: **01.07.2022**

TABLE OF CONTENTS

TABLE OF FIGURES

PREFACE

The development of the facility asset Information Modelling Framework (IMF) was progressed through the READI JIP, resulting in an IMF Concept document issued in March 2021. Following this, the development continued as part of the Equinor Krafla project and then was extended to include the partners of the NOAKA cooperation, Equinor and AkerBP. The SIRIUS Centre at the University of Oslo has also contributed to the IMF work. In parallel with the continued development of the IMF framework and methodology, the implementation of IMF was pioneered by the Equinor Krafla project together with Aibel, being the engineering contractor. Learnings from this piloting has contributed significantly to enhancing IMF, which has now reached a maturity level when a Reference Manual can be produced comprising documentation and specifications that together answer what a company must do to implement and use IMF.
It is intended that the result shall form the necessary basis to create a DNV Recommended Practice for how to implement IMF in the industry. Responding to an increasing interest it is also an objective that the result serves as a basis for a wider involvement from the industry, possibly aiming at making it a Standard.

1 PURPOSE

1.1 TRANSITIONS TO INFORMATION MODELS OF FACILITY ASSETS

The purpose of the Facility Asset Information Modelling Framework (IMF) is to enable transitions to information models of facility assets from current documentation practice. In order to achieve its purpose, the IMF framework is comprised of methods and resources designed to support incremental and scalable implementation.

By an information model we mean a structure of objects and relations. Objects can be referred to by descriptors, i.e., descriptive references that are meaningful for human experts, or by identifiers used to access data stored in IT systems. Objects can be classified according to classification schemes and carry attribute information. Relations, like “is part of” and “is connected to”, have names and relate objects in a way that can be displayed in a diagram as arrows between objects. Unlike objects, relations do not carry any attribute information.

When designing an information model of a facility asset, the designer of the model provides answers to questions like this. “How shall the attribute at hand be named? Which object does it belong to, and how is that object identified?”. “How shall the newly created object be named? Which other objects is it related to, and through which relations?”

The answer to such questions depends on *who asks* and in *what context*. IMF addresses questions from different professional roles, including:

- The Subject Matter Expert that, e.g., develops a design of the main processing system for the FEED contractor,
- The Application IT expert that, e.g., exports data from an engineering authoring system into an engineering register,

- The knowledge engineer occupied with ensuring internal consistency of the information model,
- The multidiscipline engineer responsible for coherent integration of system designs from, e.g., different disciplines, different phases of the project lifecycle and different parties in the value chain.

The IMF approach to role and context is key for achieving incremental transition to an information model-based practice driven by scalable implementations of the modelling framework.

1.2 Context and Roles

IMF provides for incremental implementation where current ways of working are dominated by legacy systems, tools, and work processes, by creating value at each step of a gradual modelling exercise. Rather than aiming at developing one single model of an asset from the start, IMF identifies different types of models, where context models are the most basic ones. A context model is a model of a system, typically from one perspective, from one contributor in the value chain, and at one stage in the project lifecycle. A context model has clear interfaces to existing documentation, and thus provides incremental added value to the total body of documentation.

On the basis of context models, one can form composite integrated models by combining simpler models into more comprehensive ones whenever such comprehensive models are useful. In particular, this approach enhances collaboration between the client and multiple contractors and suppliers, by providing a common language for communicating facility asset design information and integrating various contributions along the value chain into a single coherent model.

The Information Modelling Framework critically meets some key requirements for it to be a viable and scalable solution:

- It accommodates the very different needs and expertise of Subject Matter Expert (expert on discipline engineering) and the Knowledge Engineer (expert on ontologies and semantic technologies), by providing different sets of vocabulary as well as the means to translate between them.
- It introduces and clearly defines the role and the required contribution of Digital Multidiscipline (expert on information model integration and data flow).
- It enables Application IT (expert on information technology systems and infrastructure, data repositories, and IT applications used by the Subject Matter Expert) to specify and maintain mappings between different identifier systems by exploiting context models.

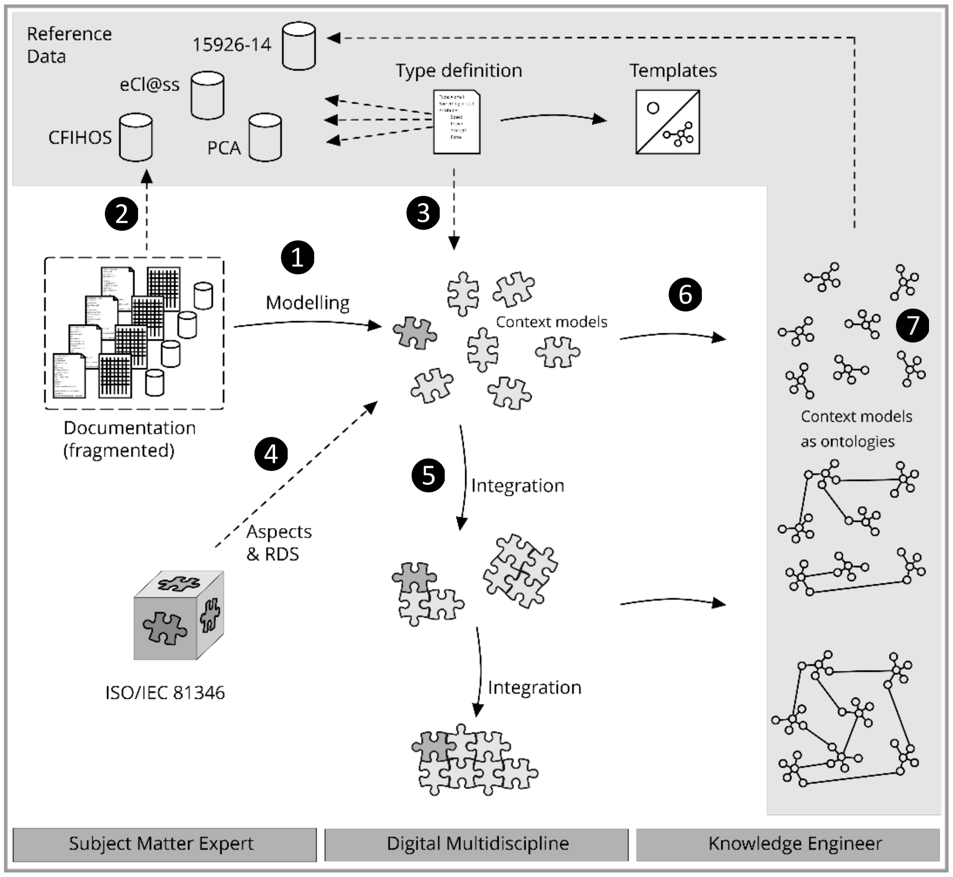


Figure 1: Different expertise and how they enable information modelling of facility assets.

Figure 1 illustrates how different expert domains work together to enable modelling of parts of the facility asset that then are integrated into a complete model where semantic technology and common industry reference data is utilized to verify and validate the integrity of the model.

The Subject Matter Expert models information that today is available only in fragments in current documentation ❶. This documentation is document centric. It refers to standards, possibly exploiting reference data, such as names of shared properties and classes, and initiatives to digitally enrich documentation format such as DEXPI ❷. In modelling in IMF the Subject Matter Expert instantiates Type definitions from an industry library ❸, enabling the re-use of well-proven design patterns (e.g., type of pump configuration). Instantiated types are inserted into a Context model using the structuring principles from ISO/IEC 81346 ❹.

As Figure 1 illustrates, IMF does not require all to work on one, single model, something which often is implied when referring to data-centric or model-centric ways of working. Instead, the modelling can be done as many smaller Context models, here shown as puzzle pieces, that Digital Multidiscipline later can bring together as a complete puzzle ❺. This integration is not only about smaller models being part of a larger model, but also about relations between elements in the various models, such as between different aspects or disciplines. Context models can be translated into ontology using ontology templates ❻. This enables the Knowledge Engineer to analyze and use the models, possibly exploiting powerful verification techniques such as automated reasoning ❼.

Part of the SME task is to relate engineering numbering used in documentation with IMF types. This is illustrated in Figure 2. The Subject Matter Expert maps, e.g., a property in an engineering register, linked to a tag number, to a Type definition ❽. Identifiers for the objects in the engineering register are managed by the IT Expert. It is normally a difficult task for an IT expert to establish and maintain identifier mappings from engineering registers to other applications or registers because this as a rule requires information from the SME. However, in IMF it is precisely this information that is captured in ❽. In consequence, the link from am engineering register to objects in a Context model can be established in two separate steps: The IT Expert generates a link to an IMF TAG object ❾, usually a query to the engineering register, while the link from the TAG object to the Context model is generated from the link that the SME created at a type level ❿. Thus IMF achieves a clear-cut division of labor between the SME and the Application IT Expert in establishing and maintaining identifier mappings.

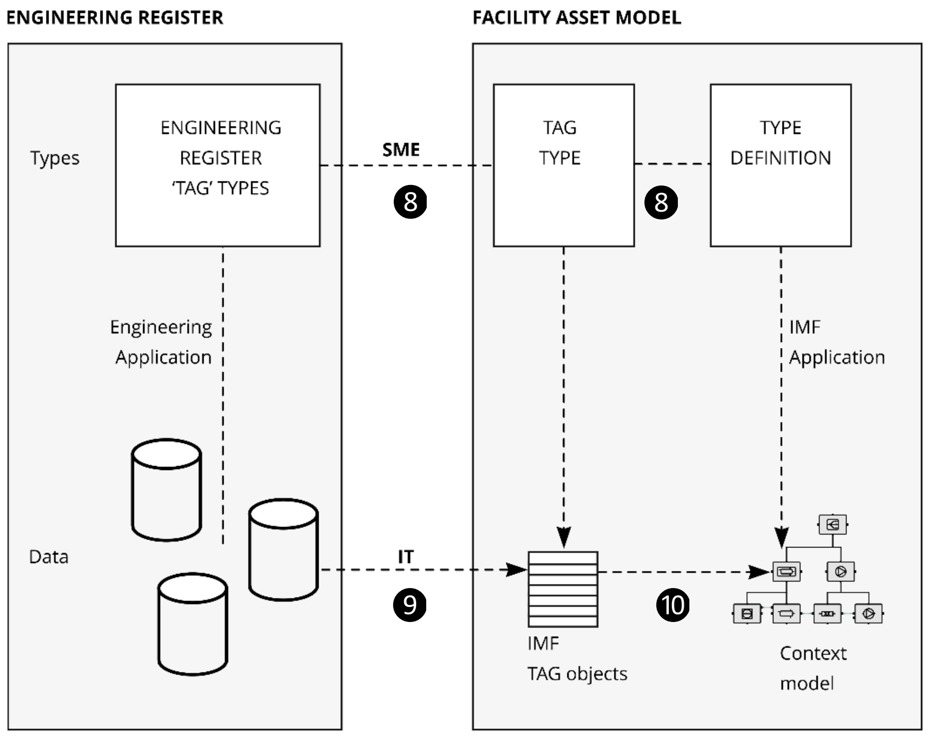


Figure 2. Division of labor between SME and IT expert in defining and maintaining identifier mappings.

1.3 Current Documentation Practice Versus Information Models of Facility Assets

The purpose of IMF can be realized in many ways. One case in point is discussed here, contrasting the current practice of project documentation with the development of a context model. At present, the way of working is suffering from a loss of information along the value chain. There is also a great potential for improving the efficiency of the design process, in particular by depending less on documents, and instead by utilizing information modelling.

Figure 3 illustrates the problem of today's way of working, as well as it is indicating a model-based approach. The figure illustrates the *logical* flow of value creation during a facility asset project, whereas the actual execution schedule will have many overlaps and iterations that are intentionally left out. The figure is intended to provide a conceptual overview; details should therefore be disregarded here, as they will be discussed throughout this document.

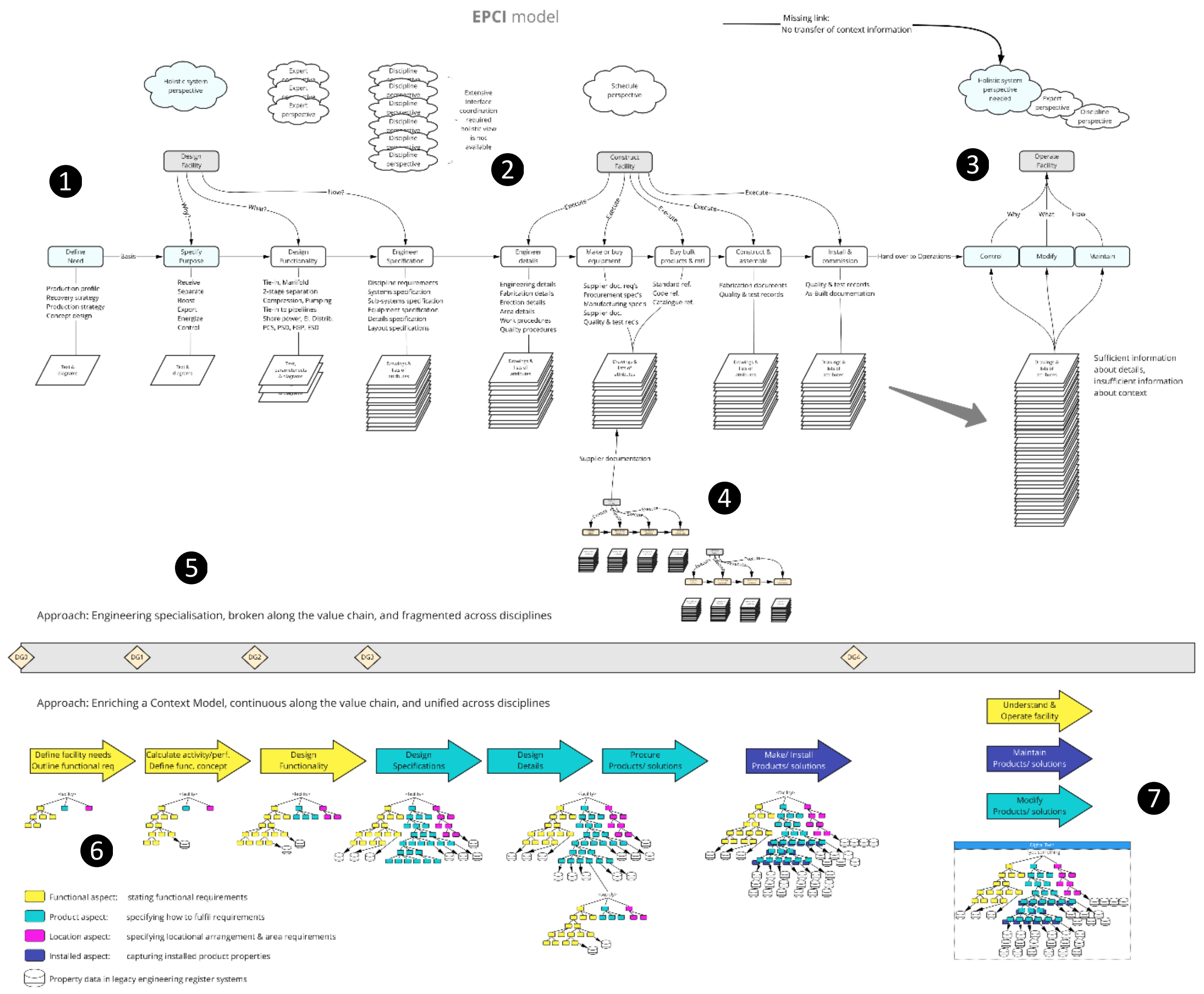


Figure 3: Current documentation practice versus new way of working utilizing information models of facility assets.

When a facility asset is developed, the work begins by defining the overall requirements and functionality. The result is typically contained in one or a few documents, which means that at this stage a holistic description is feasible ❶. As the work progresses, more specialization is necessary in order to develop the details. Since the way of working is document-based, the result is an increasing number of documents. Because of this fragmentation, it becomes gradually impossible to keep up a holistic description ❷. When at the end of the execution ❸, where the handover of facility asset information to the operator takes place, the fragmentation has grown to the extreme. Information about individual parts of the facility asset is provided, but very little about how they relate and are part of a whole. This situation is further exacerbated by even more fragmented information that will have been provided by subcontractors and suppliers ❹.

Along the project schedule, there are normally a few main decision gates ❺. Lack of overview at decision gates will in many cases decrease the quality of decisions, as an overview is hard to attain when most information resides in unrelated fragments.

When instead a model-based approach is chosen ❻ the information about the facility asset can be modelled as one context throughout the project, continuously enriching the model with increasing detail. The individual contributors may work on separate context models that are later integrated. When the handover of facility asset information to the operator takes place ❼ there is no loss of information or context, and the information is available at a holistic level, accessible at any level of detail. Furthermore, information is not restricted by document formatting, but can be navigated freely. Significant savings are made during project execution since time and resources spent on document creation and management are dramatically reduced. During the operational life of the facility asset the value gained could be huge, because information for decision support is much richer and accessible.

The observant reader may have noted that in the example above a single coherent model was described in the model-based approach which, as the observant reader has noted, is a simplification of the message in Figure 1. In a real-world scenario different context models will be produced. The facility asset information model is not one single model, but rather a model of models, from which a single coherent model can be derived by successive integration effort by Digital Multidiscipline.

2 Scope

2.1 Framing of the IMF

Figure 4 illustrates dependencies between information sources in current industry practice as well as in a scenario in which IMF has been implemented.

In the current practice, an engineering design is stored in various engineering registers ❶. Data in these registers is created using specialist applications, which means that there will normally be a complex application infrastructure surrounding the engineering register data. As a result, the engineering data will usually contain duplicates in the sense that the same data is stored in different places with different application identifiers, possibly without any “single point of truth” ❷. In order to enable consistent naming across the value chain, different project phases, and disciplines, elements used in project data and documentation are named from a centrally managed engineering numbering register ❸. A prominent case in point is the tag master used to manage the project’s tag numbers. Engineering numbers are descriptors; they serve to identify an artifact by describing, for instance, its function in a way that is meaningful for the SME. The relationship between engineering numbers and application identifiers ❹ is managed by the Application IT Expert. This includes managing the information that links, for instance, a tag number with all the attributes and their values in the engineering registers that are associated with that tag number. Engineering numbers are used in project documentation ❺ comprised of various information (e.g., figures, tables, drawings, data sheets, property lists) shared across the project. It is this body of documentation that IMF targets and aims to recast into a Facility Asset Information Model ❻. The Facility Asset Information Model will capture relationships that are explicit in the project documentation, but it will also capture relationships that are only implicitly present like breakdown structures that are part of SME's background knowledge and used by them to relate pieces of information at a different scale.

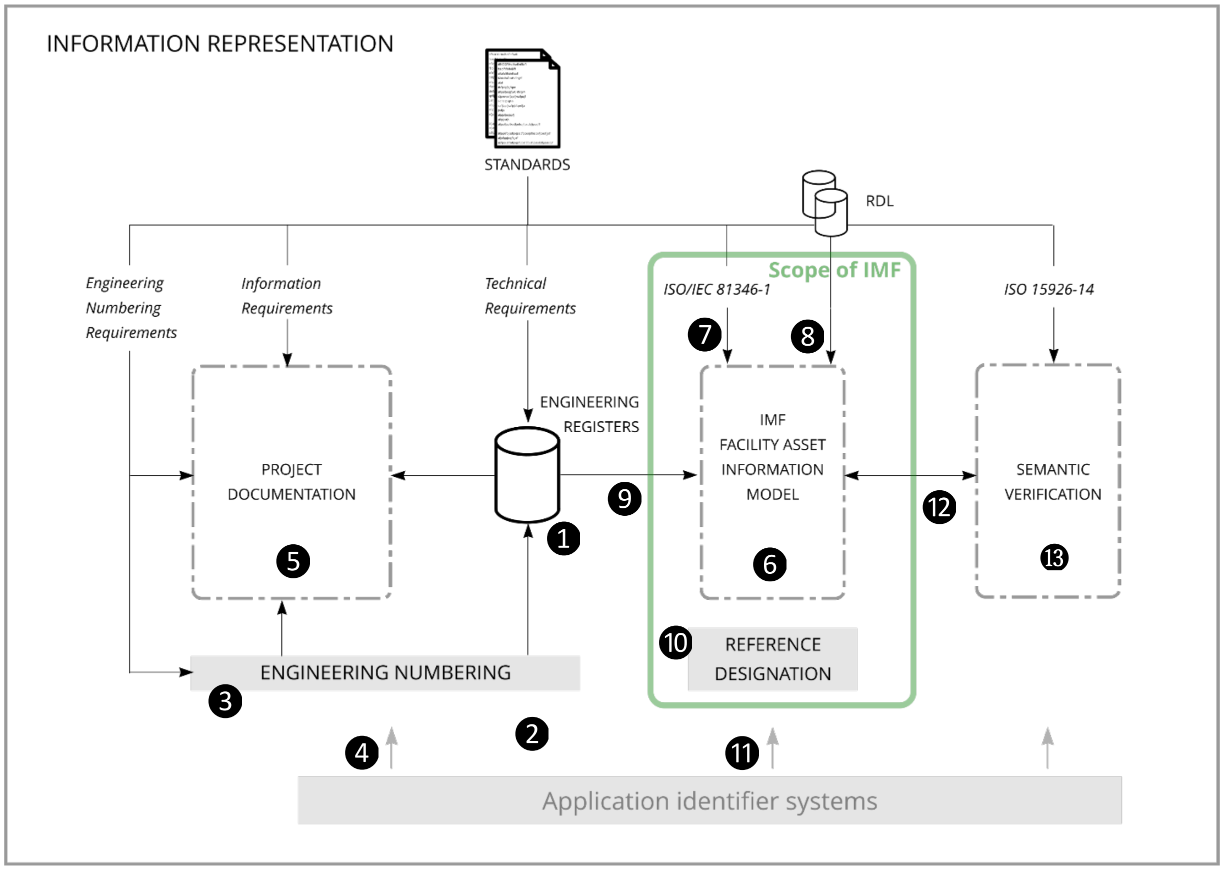


Figure 4: Framing and scoping of the IMF.

Structuring principles from 81346 Part 1 ❼ are used in the modelling Facility Asset Information Model. Reference data is used to build type definitions ❽. Reference data includes RDS codes from 81346, Part Oil & Gas, which are used to build reference designations ❿, a system for descriptors that is in part self-managed and in part co-managed. Data values from engineering registers are attached to Context model objects ❾. The Application IT expert and the SME both contribute to mappings between identifier systems ⓫. Conceptual frameworks from ontology are applied to guide the selection of modelling concepts and ensure that they are applied consistently ⓭, where the ontology is generated from the Facility Asset Information Model using ontology templates ⓬.

2.2 Scope of the IMF

Creation of IMF facility asset information models, including:

- Concepts used to create context models, including system, type definition, aspect.
- Language for representing context models, including references to reference data libraries.
- Language for representing facility asset information models.
- Descriptors, including reference designations from 81346.

2.3 Interfaces to the IMF Scope

Interfaces to IMF facility asset information models includes:

- Concepts for specifying mappings to engineering registers.
- Concepts for specifying relationships between descriptors, i.e. between engineering numbering and reference designations.
- Templates mechanism for translating context models to ontology.
- Mechanism for using RDL resources.
- Utilization of concepts from 81346-1.

2.4 Examples of Use by Target Groups

Examples of using IMF facility asset information models, including:

- Use of IMF facility asset information models to explicate requirements
- Use of IMF facility asset information models to provide a context service by establishing and maintaining mappings between application identifiers
- Use of ontology reasoning for semantic verification
- Use of IMF to exchange asset information across the value chain using open exchange protocols
- Use of IMF as specification for application developers

2.5 Not in Scope of the IMF

Software tools have been developed that have been instrumental to the demonstration of IMF for facility asset modelling, and that will keep being essential for first movers wanting to implement IMF in their value chain work processes. These will continue to be developed in the form of Open Source projects, and the intention is that this effort will result in branches of development also with providers of engineering tools and facility operation information tools, to further accelerate the uptake of IMF. These software tools however, are not part of the IMF scope.

3 Conceptual Introduction & Background Knowledge

This chapter intends to give a conceptual introduction, and therefore some terms will be introduced but only briefly described without going into details. The details and precise definitions are provided mainly in Chapter 4.

3.1 Concept

IMF is an information modelling framework for facility asset modelling. A main pillar of the IMF concept is Systems Thinking which in this context is a way of managing the complexity of an intended facility asset by looking at it in terms of a system description repeatedly being broken down into more narrowed down system descriptions, and so on until a satisfactory level of granularity is reached. A second main pillar is the description of relationships between systems. This relationship information is enriched by the concept of aspects, allowing the description of relationships between different *aspects* of the systems. A facility asset model in the IMF format can be translated into an ontology to enable semantic verification.

A model is built by using templates from a common library ❶ to create the objects that represent the individual system descriptions, then connect ❷ these objects into the hierarchy and to the relations they have. To obtain the values of attributes of the objects ❸ a mapping to the source of these attribute values is established.

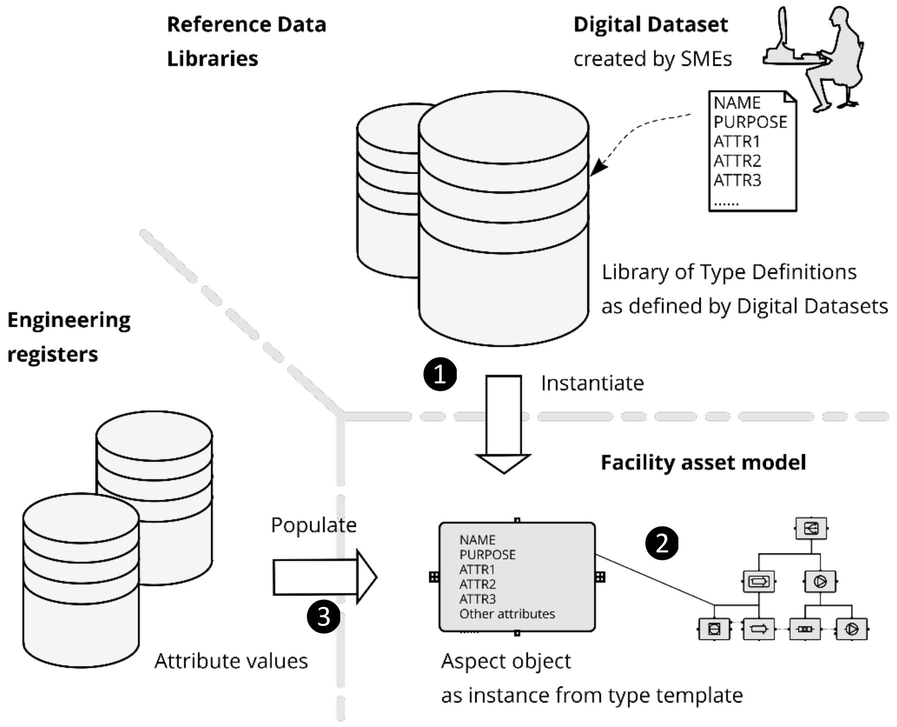


Figure 5: A conceptual illustration of the IMF and its link to Reference Data Libraries, Engineering registers, and Digital datasets.

3.2 VOCABULARY

Figure 6 shows the individual modelling elements available to the SME when modelling a facility asset in accordance with the IMF.

An Aspect object is characterized by its Aspect, Purpose, Attributes, and Terminals when applicable. To create an Aspect Object, a template is fetched from a Type Definition library, and used to instantiate the Aspect Object. To create a facility asset model, Aspect objects are created and put in a structure to specify how the Aspect objects are related to each other. This structure has two dimensions – the hierarchy dimension stating Aspect objects being *part of* higher-level Aspect objects, and the topology dimensions stating how Aspect objects are *connected to* each other. To allow connections where Aspect objects have input streams or output streams, input- and output terminals are connected by means of interface points. A terminal is defined by the Media it conveys, which can be fluids, energy, force, or information.

Every Aspect object has descriptors and identifiers, and the facility asset model as a whole has a model context descriptor. There can be many sub-models with individual model context descriptors.
The descriptors for Aspect objects are based on the IEC/ISO81346 standard. Using these descriptors (RDS), which are 1-, 2-, and 3-letter codes plus a sequence number, a description of the Aspect object Purpose is given, and a concatenation of these codes traversing from the Aspect object and to the top of

the hierarchy gives the location in the hierarchy.
Yellow, cyan, magenta, and blue are by convention used to signify aspect, as are prefixes to the Reference Designation System (RDS) codes, as is shown in the figure.

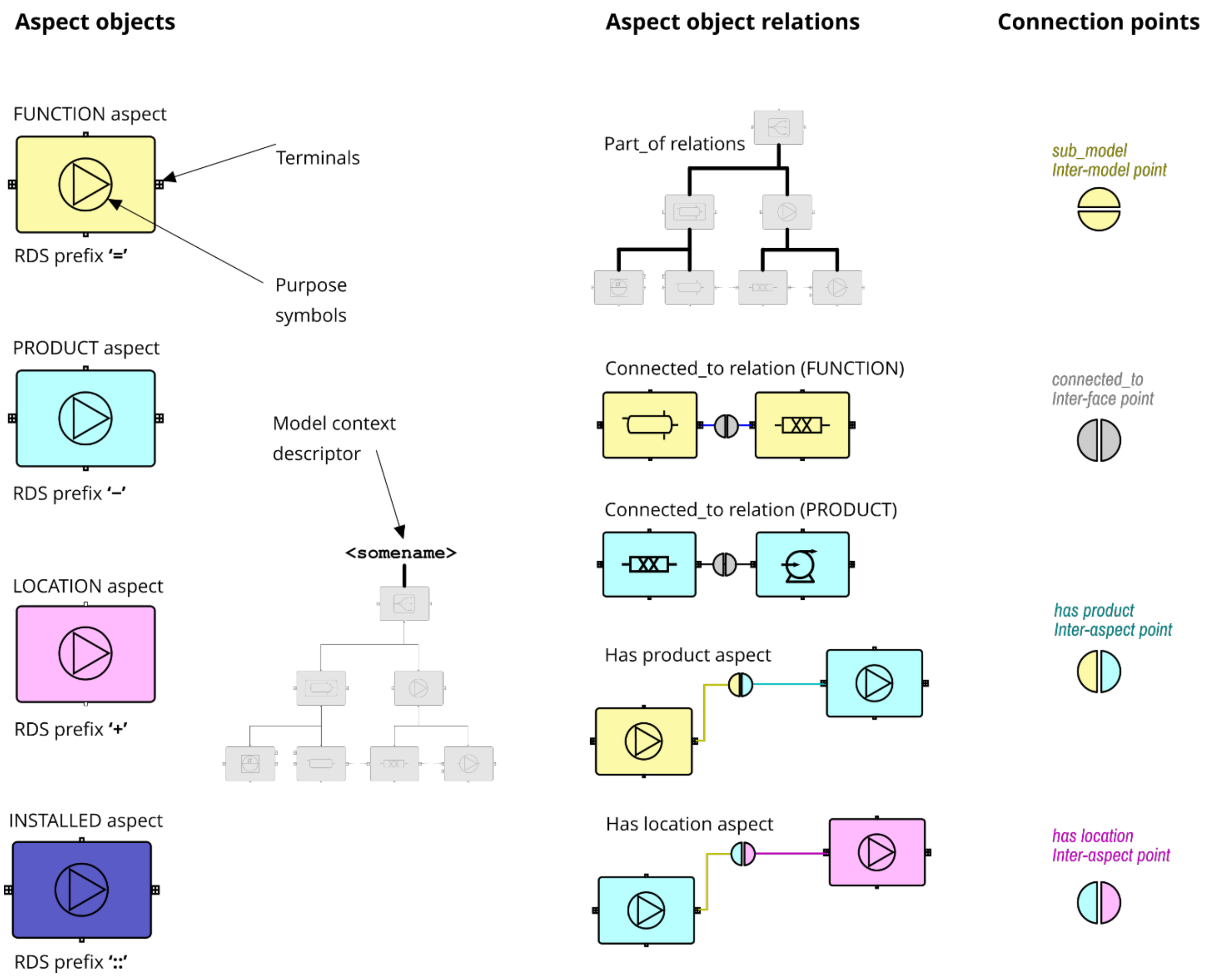


Figure 6: Individual modelling elements available to the SME when modelling a facility asset in accordance with the IMF.

3.3 The Concept of Aspects

The concept of aspects is that when viewing something from different perspectives the resulting information will be different. The IEC/ISO 81346 standard formalizes this concept by introducing a few defined aspects. The concept of aspects is embedded in the IMF vocabulary. In theory, there is no limit to how many aspects can be applied, but for the purpose of facility asset modelling, understood as the description of an *intended* facility asset, the aspects Function, Product and Location as a minimum are needed. To extend the facility asset model with information about the *actual* facility asset, the Installed aspect is also needed. Figure 7 illustrates these aspects as being different perspectives on the information about an intended pumping activity. The Functional aspect ❶ is about the intended activity, in this example: to pump, providing information about required activity, performance, and function. The Product aspect ❷ is about the specification of a solution that is intended to perform the activity, in this example the specification of a particular type of pump. The Location aspect ❸ is about the spatial envelope – the size and shape – of the specified pump. The Installed aspect ❹ is about information about the actual pump, with information such as serial number, run hours, and status.

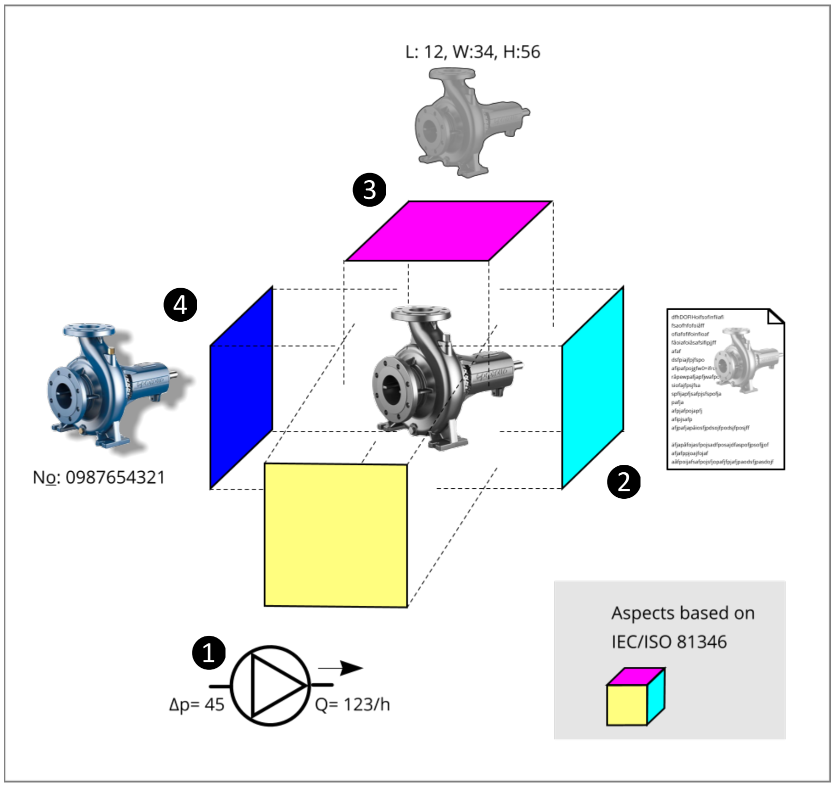


Figure 7: The concept of aspects.

It is valuable alone to structure the information about something into aspects, but more importantly, this enables relating different aspects of something into different break-down structures that each represent different perspectives on the facility asset as a whole.

How an object relates to the different aspect break-down structures is illustrated in Figure 8.

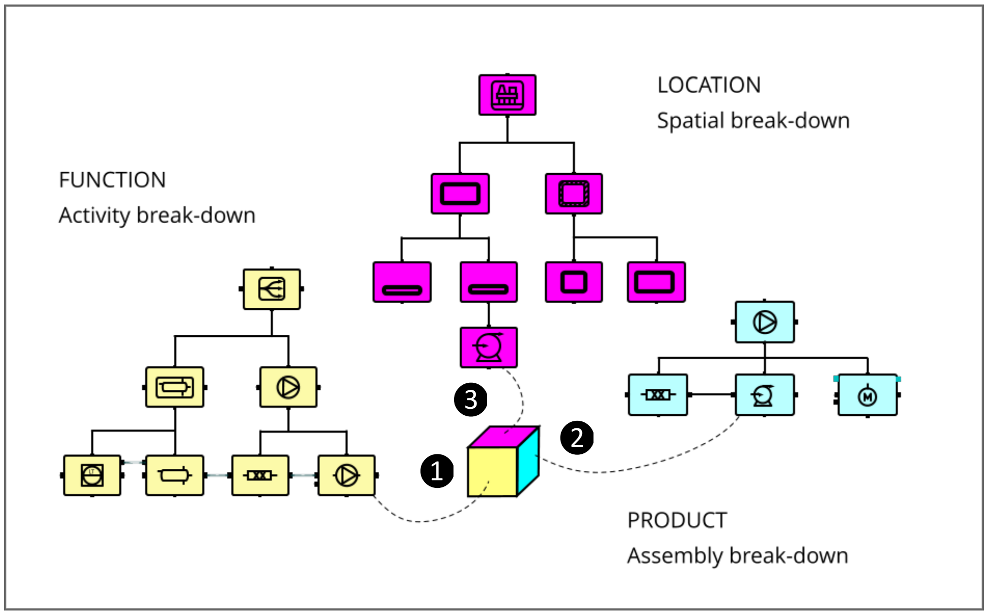


Figure 8: Aspect break-down structures.

Relations between aspects may arise when something (the cube) has several aspects that each relate to different aspect of the model. This example revolves around the intended pumping, where ❶ is the functional aspect of the pumping, which is *part of* a pumping system, which again is *part of* a separation system. Note that the break-down structure in the Function aspect is about breaking down a main activity (separating) into sub-activities. The Product aspect ❷ gives the specification of the pump as a thing and being a thing the break-down reflect how this thing is *part of* a larger assembly of things. The Location aspect ❸ is information of the thing in the space dimension, and by this space being part of a break-down of spaces, its relative location is specified, also allowing requirements specific to this location to be specified.

3.4 Aspect Object Types

Figure 9 illustrates how – as part of facility asset modelling – one asset facility building block is modelled by means of Aspect objects. In this case, the building block is something that ultimately will materialize as an actual pump ❶. First, the need for a pumping activity is identified and specified as a pumping function, i.e. by an Aspect object in the Function aspect ❷. The Aspect object in the Product aspect ❸ is used to specify the solution intended to perform the activity and fulfill the functional requirements. The specified product has spatial attributes (e.g. size) that are defined by an Aspect object in the Location aspect ❹. At some point, the specified product is installed as an actual piece of equipment ❺ in which case it will have attributes that are actual, such as serial number, running hours, etc.
Other aspects ❻ may be introduced as needed, e.g. to define maintenance attributes and relations, to define tagging/labeling and relations, etc.

There may be relations between objects of different aspects, such as ❼ to state a relation between a specified activity or function and a specified product - or a relation ❽ between a product and a location. Also, a specified product can be related to ❾ the actual information about the installed product.
This regime can be extended ❿ as needed to specify relations to further aspects.

A wide range of Aspect objects are needed to fully model a facility asset model. In this example it is about a type of pump, but it could also be about a type of valve, a type of processing activity, etc. To accommodate this a common library provides templates for the Aspect objects needed to be created (instantiated) ⓫. Such a template comprises ⓬ specific types of attributes, codes, symbol references, etc. available from libraries of standardized content ⓭.

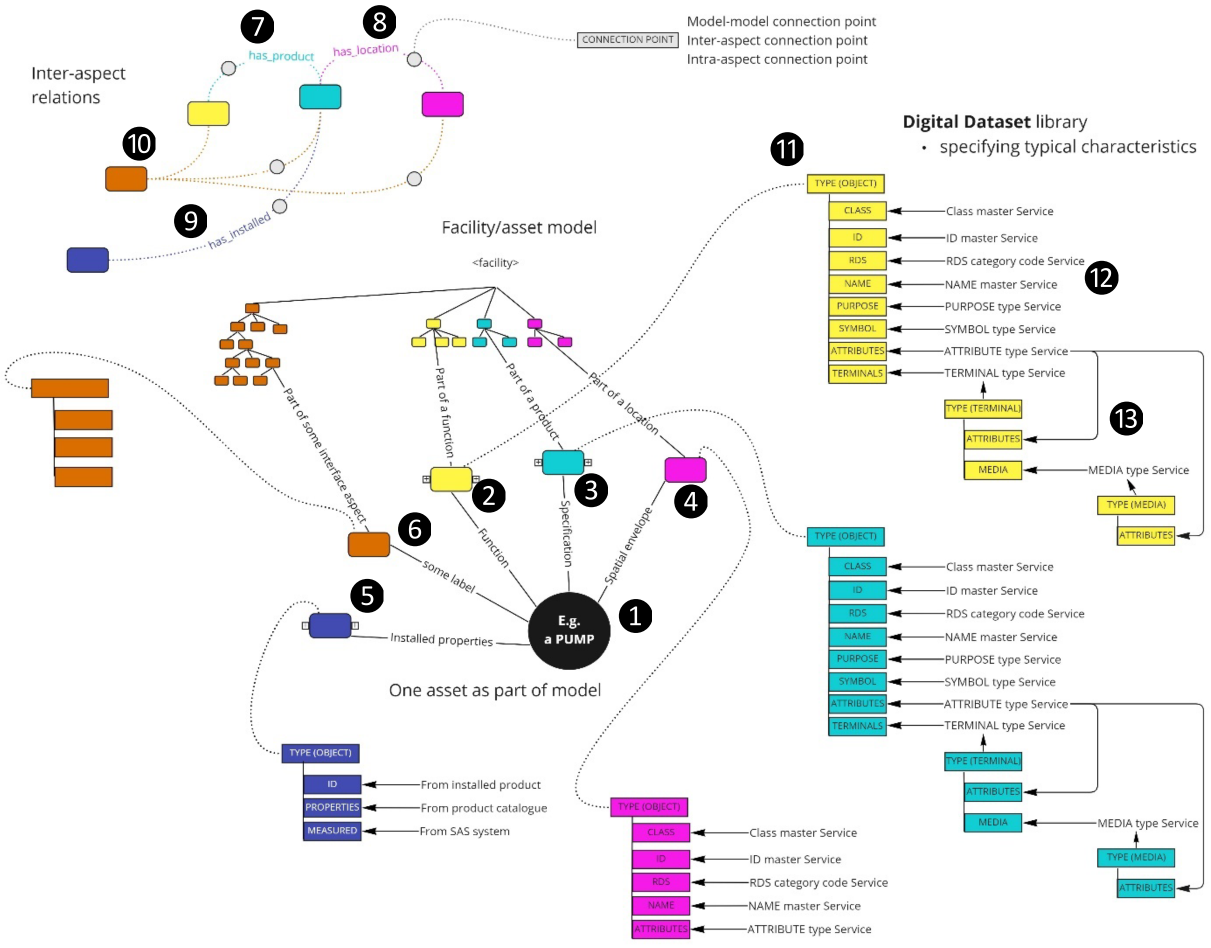


Figure 9: A facility building block modelled by means of Aspect objects.

3.5 Classes

Class hierarchies are not part of IMF, however they are relevant because the *Type Definitions* used to instantiate Aspect objects are fetched from the *Type Definition library*, where the templates are ordered in a class hierarchy.

Figure 10 illustrates the concept of classes and class hierarchy in this context. Such a structure can greatly reduce and simplify the work to define new templates because it enables a new type to inherit the properties of parent classes. Referring to the Figure 10, a ❶ Centrifugal Pump may inherit all the properties of a Rotary Pump ❷, since it is a specialization of a Rotary Pump. Likewise, a Rotary Pump may inherit all the properties of a Pump ❸, of which it is a sub-class. Thus, the re-use of former and higher-level definitions is made possible.

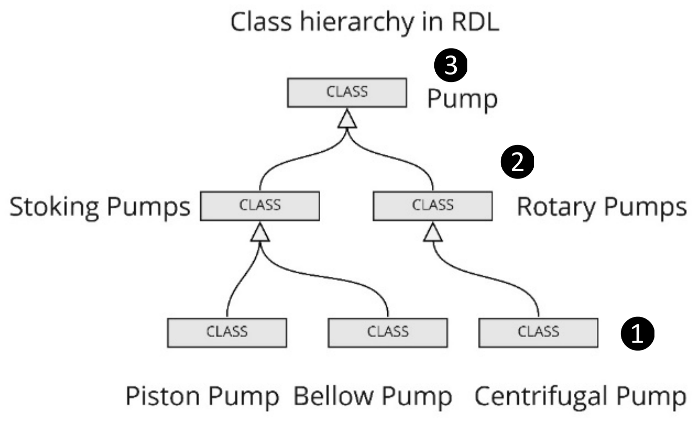


Figure 10: A class hiarchy in the Reference Data Library.

3.6 Subject Matter Expert Modelling

Utilizing the vocabulary of IMF and the resources of the Type Definition library, a facility asset model can be made. The system thinking of Subject Matter Experts (SMEs) is at the core of how a model is developed, beginning with defining the main system(s) description and progressively breaking systems into smaller system descriptions with an increasing level of detail.
Which aspect logically comes first will depend on which SME discipline does the modelling. Typically, the Process discipline, as illustrated in Figure 11, will begin by defining the process activities the facility is intended to perform, and the corresponding required functions. The Architectural or Safety disciplines however will typically begin with defining and classifying the main areas and their subdivision into smaller areas. Figure 11 takes a separation system as an example ❶ and illustrates how the separation system is broken down into its subsystems ❷ and so on. Only a very small subset of the attributes is shown here. Adding to the system-of-system dimension, the IMF also provides a vocabulary for defining interfaces between subsystems, e.g., how separation is connected to filtering ❸ which again is connected to pumping.

Figure 11 also illustrates how the IMF allows the SME to work within a sub-scope of the complete facility asset model, and the SME in this example focuses entirely on the separation system and the process discipline. IMF provides a vocabulary and methods for integrating several sub-models into a model of models at the point in time when required. Likewise, IMF provides for splitting out sub-models from a model whenever the SME work process would benefit from this.

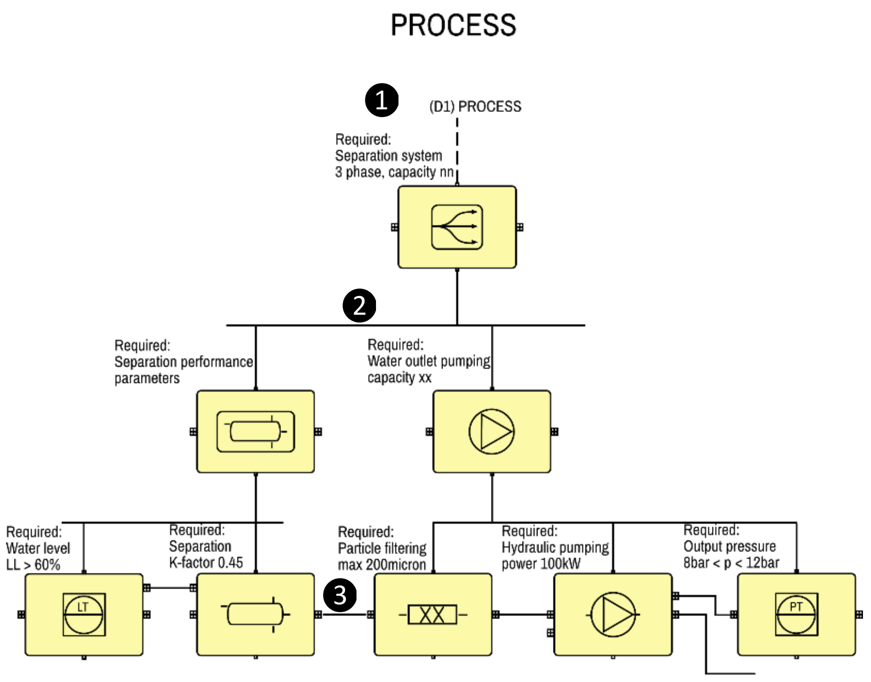


Figure 11: Subject Matter Expert modelling of a separation system.

3.7 Multi-Discipline Model Integration

Figure 12 illustrates how IMF supports integration across disciplines by modelling the design thread and requirements thread from its origin in one discipline to its destination in another discipline. Two examples are shown: ❶ How the pumping activity (Process) needs power input, and how there must be a deliver-power activity (Electro), including how the prerequisite for this activity cascades up the electrical systems. ❷ How the separation and pumping activity (Process) needs to be controlled and protected, and how there must be control and shutdown activity (Automation), including how this depends on there being an automation system.

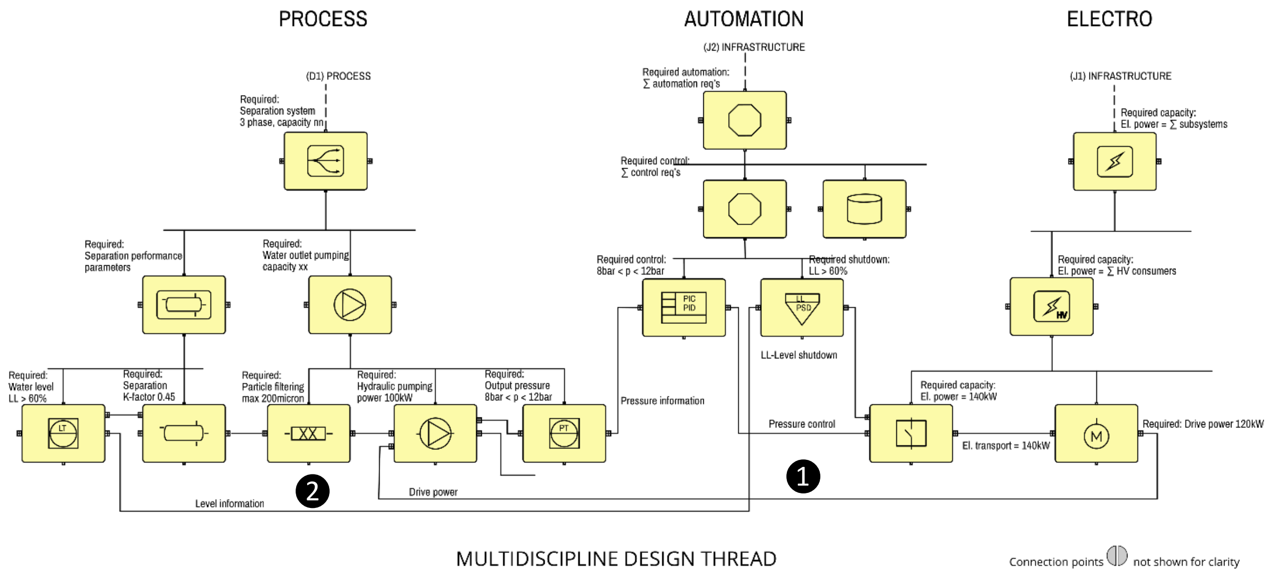


Figure 12: Integration across disciplines by modelling the design thread and requirements thread from its origin in one discipline to its destination in another discipline.

To model and specify such connections as given in Figure 12, the IMF vocabulary includes Interface Points. These contain a set of attributes that specifies what is required for there to be a successful connection. For example, the interface between the driving part and the driven part must be capable of transferring a specific amount of power.

3.8 Designation and Identifications

IMF provides for two main identification regimes: Descriptors and Identifiers. Descriptors are codes or names that are meaningful to humans. The Reference Designation System (RDS) codes from IEC/ISO 81346 is an example. Depending on the purpose, different classes of descriptors and identifiers are used. They are also different with regards to what governance is required, and by what role (user) they are managed. These include RDS codes, Context names, and IRI/UUID. By utilizing a Tag aspect, TAG numbers may also be included.

3.9 Semantic Verification

A facility asset model built in accordance with IMF can be translated into a format which enables semantic verification by means of rules and reference data. This format is referred to as an ontology, and the verification method is referred to as reasoning.

4 The Information Modelling Language

This chapter defines and gives a detailed description of the IMF language. The language is also defined as an OWL ontology, which is listed in Appendix C.

4.1 System and System Descriptions

IMF adopts a systems approach to asset information modelling. This means that literally everything is viewed as a system. This section aims to provide the reader with some intuition prior to the formal definitions based on a standard concept of system.

We shall by an *engineered system* mean *a group of interacting elements where the interaction is designed so that the elements together deliver the required functionality*. The functionality reflects its *purpose*, understood as the intended activity that the system is designed to bring about. This activity is to transform various sorts of input to various sorts of output, where such input and output can be energy, material, force, or information. The points of input and output of systems, called terminals, lie at the system boundary. Interaction between systems is brought about by establishing connections between terminals through interface points. A system is realized by physical objects that have a spatial extension. Interface points do not have a spatial extension.

The IMF language is a language for *system descriptions* intended to form the specification of an engineered system, i.e., the specification of how the system delivers its *required* functionality. Below is an informal introduction to how IMF formally articulates a system description. In Section 4.2 and Section 4.3, the formal IMF language is specified.

By a *system element description* we shall simply mean a property list, or, in other words, a set of attributes with attribute values. It is clear from our concept of system that a system cannot be described by just one flat property list. Note that our concept of system includes references to both systems, terminals and interface points. Hence a system description must contain more than just a system element description.

The formal IMF language introduces formal concepts of system element:

- A property list without data values is captured by a Type Definition
- System, terminal and interface point are captured by the system element categories System Block, Terminal, and Interface Point.

A key point in IMF is to clearly distinguish different kinds of property lists, distinguished by shared categories of properties. These categories are called aspects. Three such aspects fall directly out of our concept of system:

- Function: Properties describing the intended activity that the system is designed to bring about
- Product: Properties describing the product types of the physical objects realizing the system
- Location: Properties describing spatial extension of the physical objects realizing the system

Some examples of properties that belong to each of these categories are given in Appendix A. In addition to the Function, Product and Location aspects, IMF identifies the aspects Installed and Tag, and allows for introduction of others on demand.

In IMF descriptions are hence divided into aspects [exception: TAG, explain]. When we ignore property values, a description is captured by a Type Definition and is of one of the system element categories System Block, Terminal, or Interface Point.

An *instance* of a Type Definition is called an Aspect Object. An Aspect Object thus captures a property list, potentially with property values inserted, and is of one of the categories System Block, Terminal, or Interface Point.

IMF introduces formal relations between aspect objects:

- hasTerminal is used to relate a system block (i.e., an aspect object of category System Block) to its terminals
- hasPart is used to build up a breakdown structure within one aspect. Note that the form of the breakdown structure in one aspect will as a rule be different from the form it has in another aspect.
- connectedTo is used to connect two terminals through an interface point.
- Inter-aspect relations are used to relate two aspect objects that capture different descriptions of the same system element, for instance a function description of a pumping system (a Function aspect object) and a product description of its physical realization (a Product aspect object). Note that an aspect object in one aspect need not have any inter-aspect counterparts in other aspects.

We can now return to our starting point. IMF is a language for system descriptions. An IMF system description consists of:

- A group of aspect objects, each expressing a list of properties of the same category
- A set of specific relations, defined in the IMF language, between the aspect objects

At a superficial level this may seem as the definition of a relational database if we view a property list as a table. However, in a relational database the relations are not between tables, but between columns in tables. The natural internal structure of an IMF system description is not a tabular structure, but rather a graph, even though this can (as any graph) also be encoded in a relational database.

4.2 Concepts in the IMF Language

This section summarizes definitions of the formal concepts in the formal IMF language.

Aspect: A principle used to form collections of properties of interest in one particular view. Instances of Aspect are Function, Product, Location, Installed, and Tag. Other instances may be added.

Prime Type Definition: Collection of properties without value. If the Prime Type Definition has an aspect attribute, this aspect is unique and its properties conform to the principle expressed by that aspect. A Prime Type Definition is of one of the categories: System Block, Terminal, Interface Point.

Type Definition: Either a Prime Type Definition or a System Block related to one or several Terminals of the same aspect through hasTerminal relationship. In the latter case the Type Definition is of category System Block.

<u>Aspect Object</u>: Instance of a Type Definition. If the Type Definition has aspect, say Function, the Aspect Object that instantiate the Type Definition is said to be of that aspect (i.e. Function).

<u>Function System Block</u>: Aspect object that is an instance of a System Block Type Definition of Function Aspect.

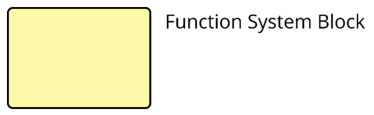


Figure 13: Symbol of a Function System block.

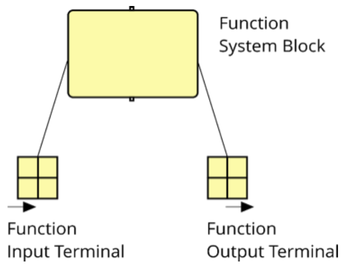


Figure 14: A Function System Block with an input and an output terminal.

Example: A collection of properties that defines a pumping activity.

<u>Product System Block</u>: Aspect object that is an instance of a System Block Type Definition of Product Aspect

Example: A collection of properties that specifies a solution that performs pumping activity

<u>Location System Block:</u> Aspect object that is an instance of a System Block Type Definition of Location Aspect

Example: A collection of properties that defines spatial form and dimension of a solution that performs pumping activity

<u>Function Terminal</u>: Aspect object that is an instance of a Terminal Type Definition of Function Aspect. A Function Terminal can be either an Input or an Output Terminal.

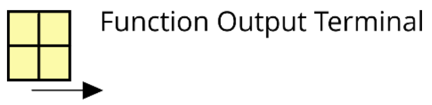


Figure 15: The symbol of an Function Output Terminal.

Example: A collection of properties that defines the output resulting from a pumping activity

<u>Product Terminal</u>: Aspect object that is an instance of a Terminal Type Definition of Product Aspect. A Product Terminal can be either an Input or an Output Terminal.

Example: A collection of properties that specify a solution that provide containment to the output of a pumping activity

<u>Function Interface Point:</u> Aspect object that is an instance of an Interface Point Type Definition of Function Aspect.

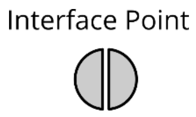


Figure 16: The symbol of an Interface Point.

Example: A collection of properties that defines the mutual requirements to an interface between a pumping activity output and a separation activity input.

<u>Product Interface Point:</u> Aspect object that is an instance of an Interface Point Type Definition of Product Aspect

Example: A collection of properties that defines the mutual requirements to an interface between a containment of a pumping activity output and a containment of separation activity input.

4.3 Relations in the IMF Language

4.3.1 System of System Relations

<u>hasPart:</u> Relation between two (Function/Product/Location) System Blocks or between two (Function/Product) Terminals.

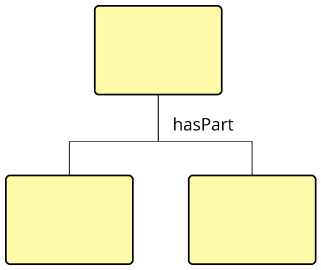


Figure 17: hasPart relations between System Blocks.

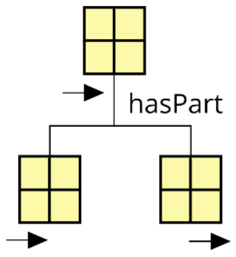


Figure 18: hasPart relations between Terminals.

Example: Relation that expresses that a pumping activity is a part of a separation process activity.

4.3.2 System to System Relations

<u>hasTerminal:</u> Relation between a (Function/Product) System Block and a (Function/Product) Terminal inherited from the level of Type Definition

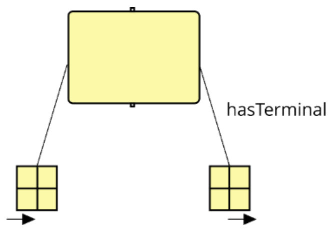


Figure 19: hasTerminal relation between a System Block and a Terminal.

<u>connectedTo:</u> Relation between a (Function/Product) Terminal and a (Function/Product) Interface Point.

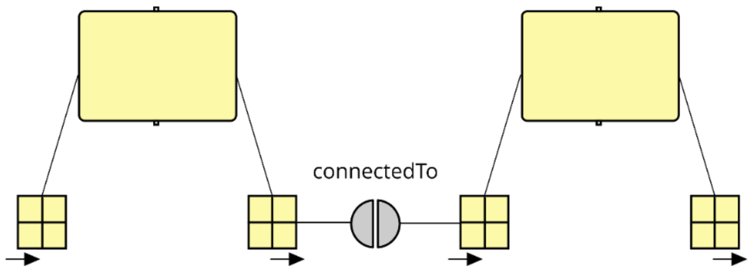


Figure 20: connectedTo relation between a Terminal and an Interface Point.

4.3.3 Inter-aspect Relations

<u>Inter-aspect Point:</u> A collection of properties that holds information about the joining of two aspect objects

Example: A collection of properties that defines how a product specification fulfills a function.

<u>hasInteraspectPoint</u><u>:</u> Relation between an aspect object and an Inter-aspect Point.

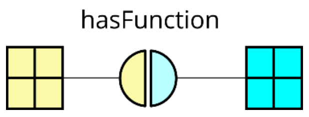


Figure 21: hasFunction relation between a Function Terminal and a Product Terminal.

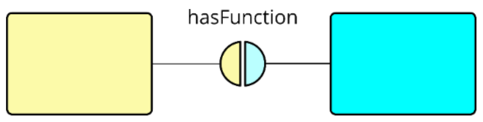


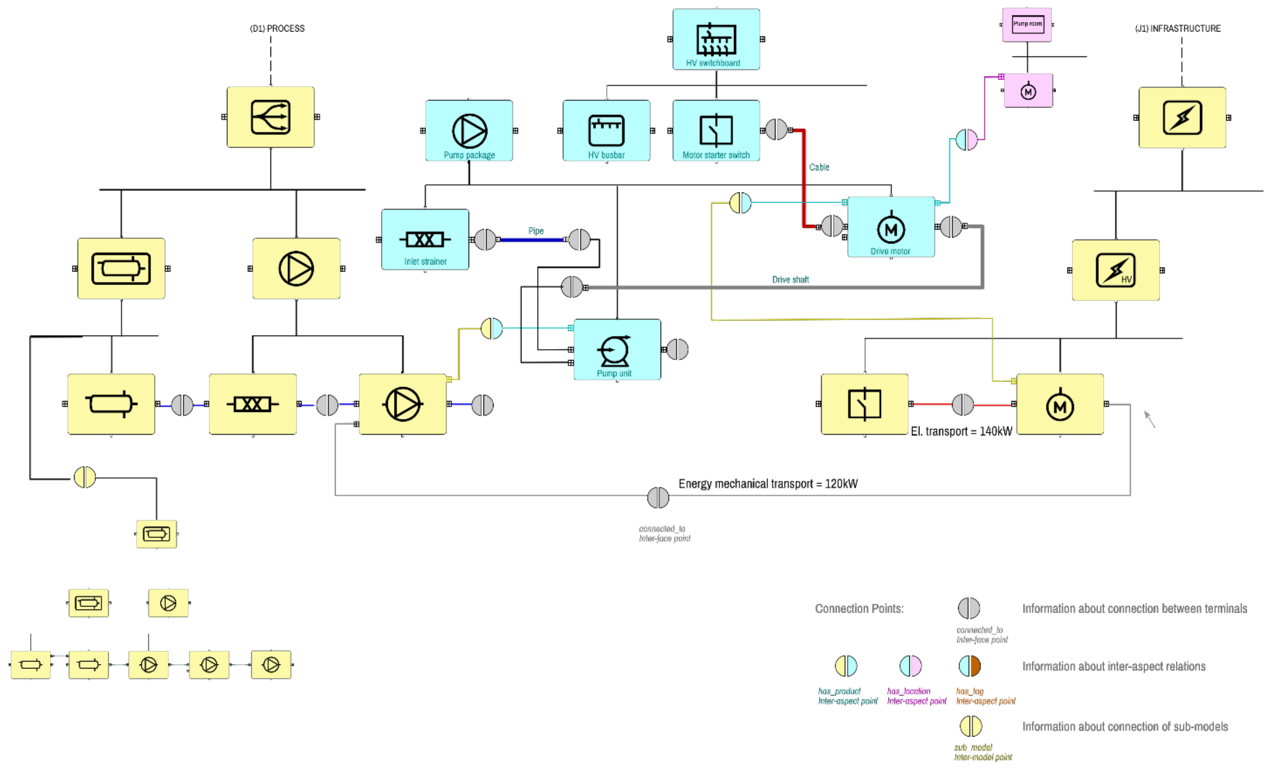
Figure 22: hasFunction relation between a Function System Block and a Product System Block.

4.4 DESIGNATION AND IDENTIFICATION

[To be spelled out generalizing the principles of reference designations in ISO/IEC 81346 Part 1]

5 The IMF Language in Use

This chapter illustrates by use of Figure 23 and hotspots of the same figure how IMF elements are used to form context models demonstrating Requirements threads, Design threads, Cross Discipline work, and inter-aspect relations.



(D1) PROCESS
HV switchboard
Pump room
(J1) INFRASTRUCTURE
Pump package
HV busbar
Motor starter switch
Cable
Inlet strainer
Pipe
Drive motor
Drive shaft
Pump unit
HV
El. transport = 140kW
Energy mechanical transport = 120kW
connected_to inter-face point
Connection Points:
connected_to inter-face point
Information about connection between terminals
has_product inter-aspect point
has_location inter-aspect point
has_tag inter-aspect point
Information about inter-aspect relations
sub_model inter-model point
Information about connection of sub-models

Figure 23: An example of a comprehensive IMF asset model using different Aspects.

5.1 Requirement Threads

IMF supports the flow of requirements – from high-level systems down to more detailed sub-systems, and from discipline to discipline. An example of this is given in Figure 24.

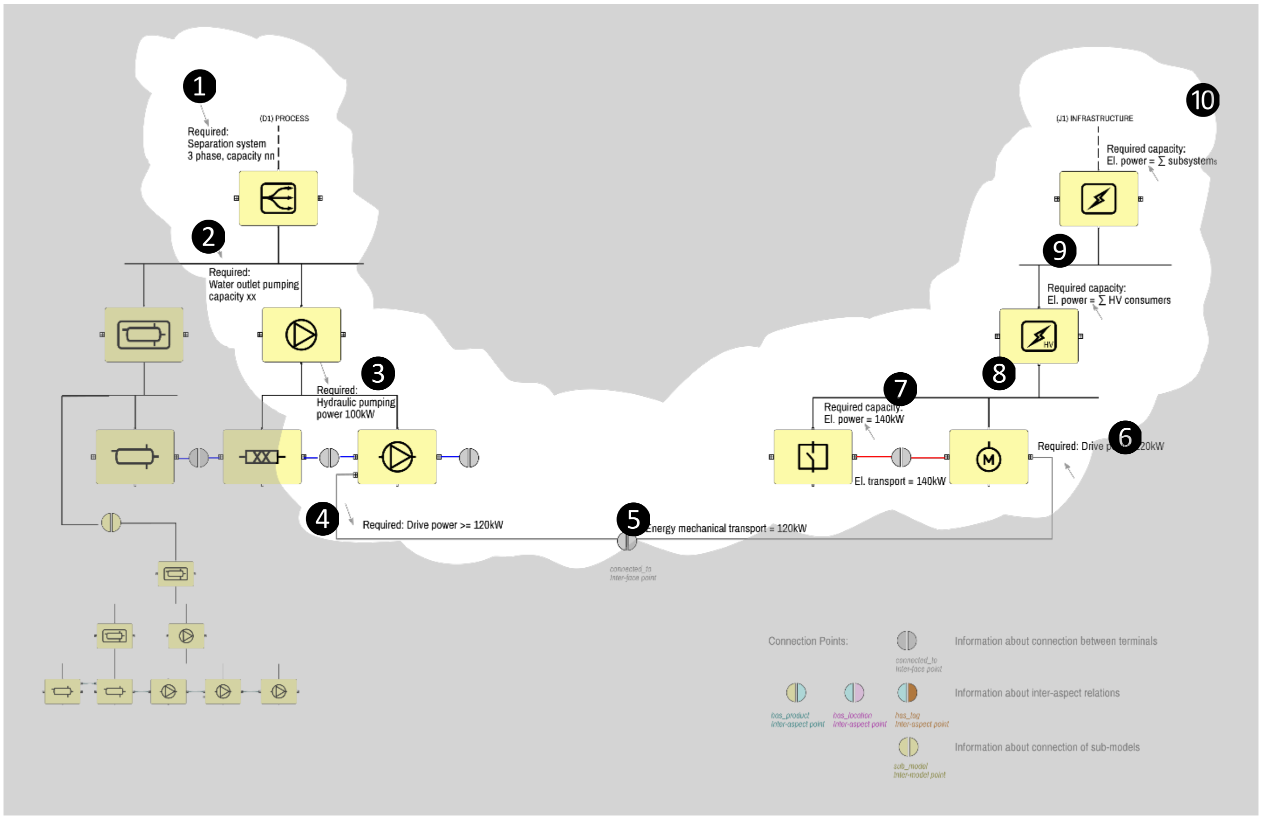


Figure 24: The flow of requirements.

Early in the design phase ❶ the overall separation requirement is established. As the design progresses, this overall requirement is broken down into requirements of the individual sub-systems, such as ❷ the required capacity for discharging water. Further detailing of this requirement results in the required hydraulic pumping power ❸. To achieve such hydraulic pumping capacity, an input of energy in some form is needed, i.e., the required drive power – including efficiency loss ❹. This drive power is required to be transported ❺ from a source of such drive power, in this case, an electrical motor which therefore is required to output this power ❻. To energize this motor, controlled electrical power is required, including efficiency losses ❼, and this power needs to be supplied ❽ by a distribution system – which also is required to supply other consumers ❾. This distribution system, as well as other distribution systems, need to be supplied with electrical power, which results in a total required electrical power demand capacity ❿. Thus, the modelling facilitates a requirement thread running from systems to subsystems, as well as from discipline to discipline.

5.2 DESIGN THREADS AND DISCIPLINE INTEGRATION

Extending on the concept of *Requirements Threads* from 5.1, the flow of design decisions, in general, is accommodated by the model, as illustrated in Figure 25. Here the design threads across the Process, Automation, and Electro disciplines are given by example.

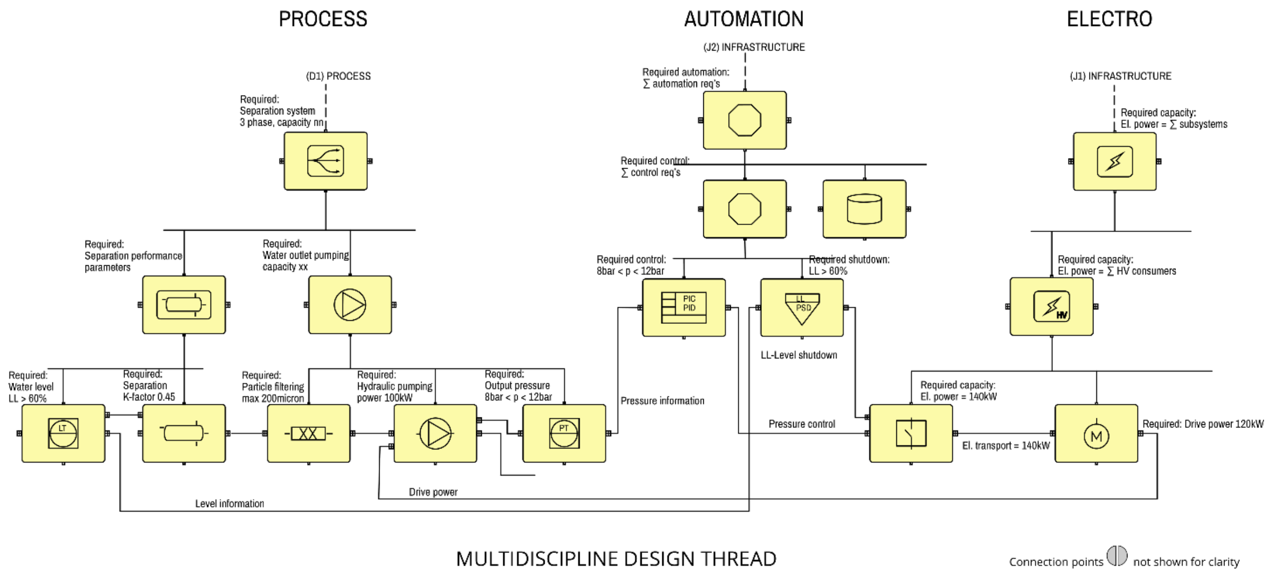


Figure 25: Multidicipline design threads.

5.3 ASPECTS

As development of the facility takes place, catering for information objects holding data about physical objects and how they are connected becomes important. ❶

As illustrated in Figure 26 Information objects ❷ representing the physical objects and their inter-aspect relations are catering for in the model. This provides identification of objects used as references for physical objects. The inter-aspect relation is identified through the Inter-aspect interface point. The physical connection between the electrical motor and the pump is a Product Aspect Block illustrated as ❸ the Shaft in the figure.

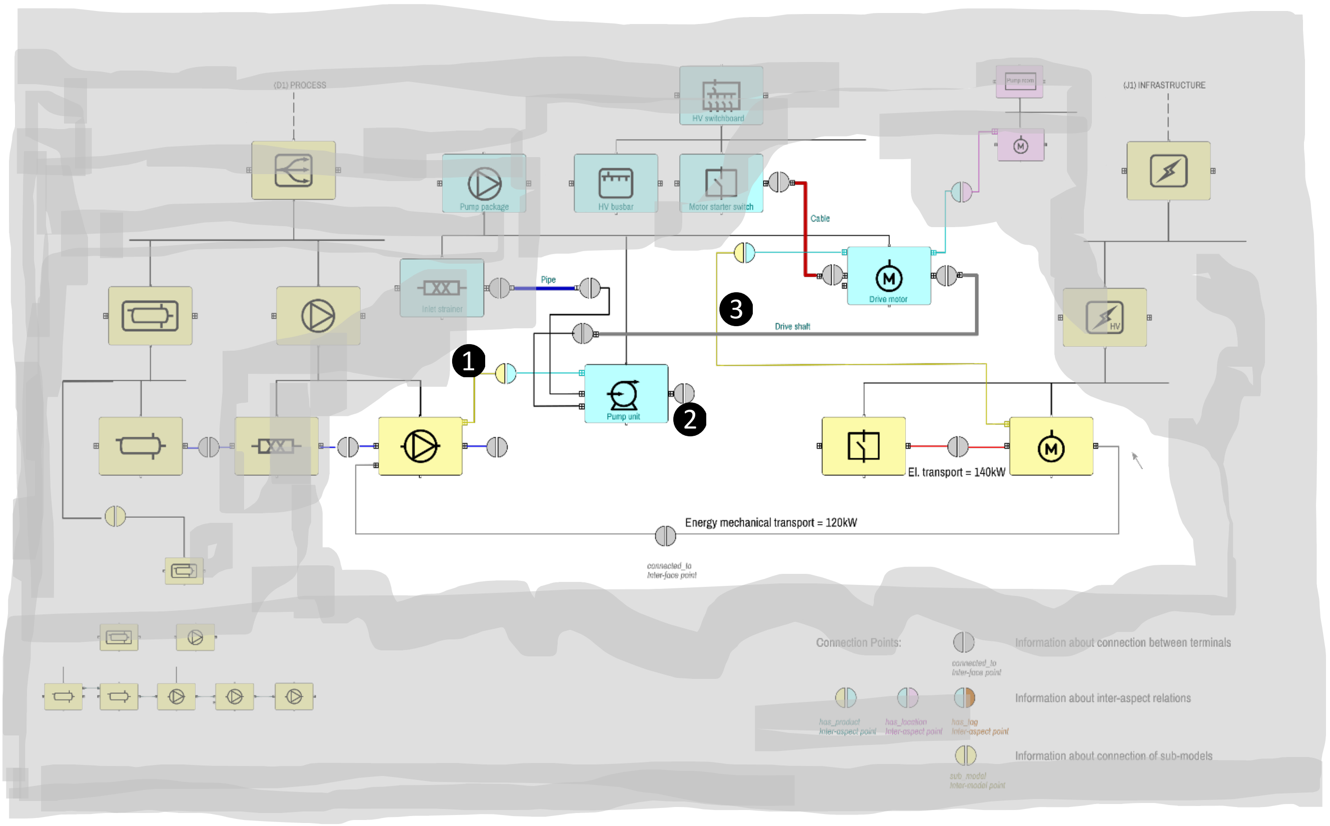


Figure 26: Inter-aspect relations.

5.4 Using Relations

[This section shall describe using an example of how to properly use the different types of relations.]

5.4.1 Part-Of Relations

[This section shall describe using an example of how to properly use the different types of relations.]

5.4.2 Topologies

[This section shall describe using an example of how to properly use the different types of relations.]

5.5 Creating a Type Definition

[This section shall describe how to create a full-fledged type definition that can be used in a context model.]

6 Interfaces to the IMF Language

This chapter gives explanations of important interfaces to the IMF language. See Figure 4 In 2.1 for a an overview.

6.1 The Technology Stack

6.1.1 W3C Standards for Data and Knowledge Representation and Verification

The IMF language uses W3C standards, in particular the Resource Description Framework (RDF), the Web Ontology Language (OWL), and Shapes Constraint Language (SHACL), as means to representing, sharing and validating IMF data and models.

OWL is the de-facto standard ontology language for practical use. It is an open standard backed by the World Wide Web Consortium (W3C), it is founded on well-known and studied concepts and principles from formal logic and builds on existing well-proven web technologies (such as Unicode, HTTP, IRI, XML) for its representation format and implementation. These fundamentals and technologies are actively supported and further developed by an innovative community comprising researchers, technology vendors and users. The following features motivate the use of RDF, OWL and SHACL.

Open and extendable (schema-less) model: As its underlying representation format, OWL uses the graph-like data model format RDF. A fundamental and inherent feature of RDF is data, since it is based on a simple graph structure, may be merged even if their vocabulary schemas differ. These features also carry over to OWL ontologies, making OWL ontologies easy, on a technical level, to extend and merge with other technologies. This contrasts with for example relational database schemas were extending the schema with new tables or columns often require more elaborate redesign.

Global identification scheme: OWL uses the Internationalized Resource Identifier (IRI) internet protocol standard as its identification scheme, i.e., using the format which is best known as the format for webpage addresses also as the format to designate model objects. This brings with it the benefit that existing web architecture can be exploited for distributing and consuming information about the identified objects, also known as Linked Data.

Support for multiple identifiers: OWL does not abide by the unique name assumption, an assumption where different names, i.e., identifiers, by definition refer to different real-world entities. OWL provides explicit constructs for specifying that different model objects refer to the same real-world object. The schema-lessness of OWL also supports annotating any model object dynamically with arbitrarily many identifying names.

Shared and distributed vocabularies and data: Using existing well-proven web architecture and technology, OWL ontologies are easily shared in a distributed and de-centralized manner. Declarative knowledge representation format: The OWL is based on formal logic that provides support for rigorous semantic descriptions of the model objects. Using declarative specifications, powerful relationships between the model artefacts, such as inheritance of attributes between objects, may be succinctly expressed. Both RDF data, OWL ontologies and SHACL expressions may be serialized in different widely accepted formats such as XML and JSON.

Verification and consistency: The formal logic that underlies SHACL and OWL and allows datasets ontology models to be formally checked by tools for correctness and inconsistencies. This is a powerful mechanism for detecting missing and erroneous data, and, e.g., duplicate classes, the existence of which causes huge challenges for data quality and correct answers to queries.

6.1.2 Reasonable Ontology Templates (OTTR) as Model Construction and Mapping Language

The Reasonable Ontology Templates (OTTR) framework is used in expressing and translating to and from the IMF models.

The OTTR framework is designed to improve the efficiency and quality of constructing and maintaining ontologies and knowledge bases, and it is built to fit with existing W3C languages and tools. The framework allows complex modelling patterns to be represented as reusable and instantiable templates, following many best-practice modelling practices and techniques, such as uniform modelling, modular patterns that encapsulate complexity, separation of concerns, and simple input formats.

OTTR templates for a particular domain or purpose are intended to be collected in well-designed libraries that are published for reuse using the same techniques and mechanisms as for OWL ontologies. Such template libraries play a similar role in the construction of ontologies as programming APIs do in software development. Capturing modelling patterns as ontology templates prepared by ontology experts in cooperation with domain pattern experts is expected to significantly lower the time to construct and maintain ontologies, while increasing the quality of the produced ontology. This is because a few ontology experts can, by building a limited, but for practical purposes, complete set of high quality and carefully aligned modelling patterns, put domain experts in the position to actively contribute in constructing complex ontologies without the need for understanding the intricacies of the underlying logical languages. The ontology experts are responsible for maintaining the template library (in cooperation with domain experts), while ontologies and data is generated from instantiations of these templates using data that is programmatically collected from different source data systems.

In IMF, OTTR templates are used for different data and model representation and transformation tasks. OTTR templates are used for representing and generating type definitions (see [ref to other section]); and for representing and generating model instance data, such as aspect objects descriptions, and also for translating IMF model data to OWL ontologies expressed using external ontological vocabularies such as ISO 15926 to perform semantic validation (see [ref other section]).

6.2 The Reference Data Library

[This section will address how IMF can make use of different RDLs and align them as part of the model integration process.]

6.3 Engineering Systems (Applications)

[This section describes IMF Context models link up to data in engineering systems, following the principles illustrated in Figure 2 in Section 1.]

6.4 Identification Systems Within the IMF Scope

6.4.1 Different needs for identification

Identifiers and descriptors are used to manage the individual entities of the IMF model and how they relate. Figure 16 illustrates different needs for identification and how IMF provides mechanisms to support them.

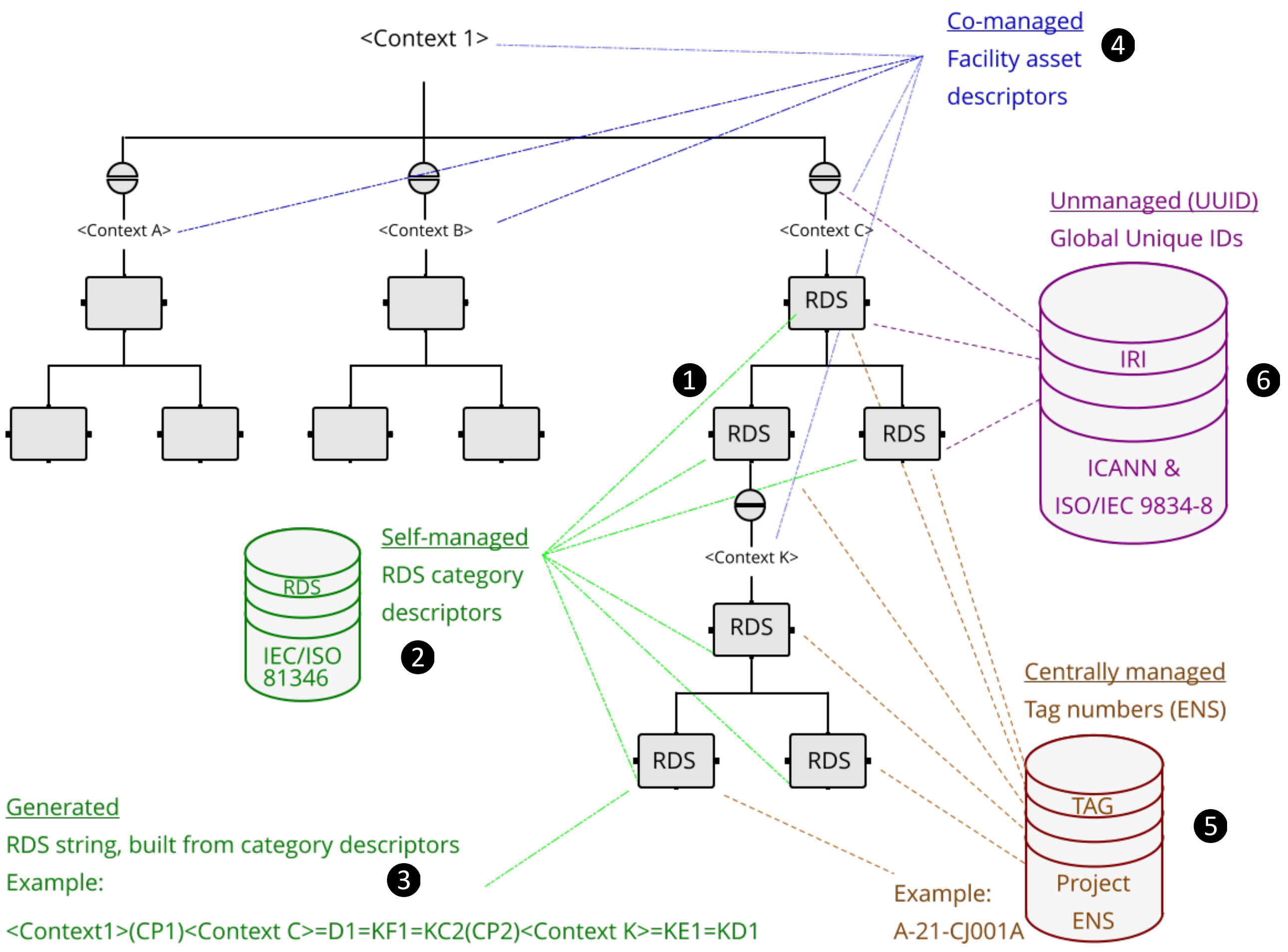


Figure 27: Indetification systems within the IMF scope.

When the SME develops a Context model, there is a need to establish unambiguous descriptors within a context that easily can be integrated into bigger context ❶. These descriptors are self-managed, meaning that they are not governed by any central register, being built up dynamically from RDS classification codes ❷, a numbering system, and the aspect objects above in the context ❸. This means, whenever the SME decides to instantiate a Type definition and insert it into the model, the new aspect object will get a self managed descriptor that serves to identify it. The descriptors generalize RDS reference designations:

- Reference Designation System based on ISO/IEC 81346 Part 1 and 2 and RDS-O&G.
- Purpose of RDS is to give Unambiguous identification of objects and relations within a context. RDS based on 81346 provide a common language across domains within facility development. RDS is in IMF classified as a Descriptive identification system.

- RDS is by nature Self-managed. Top node is defined by context and exchange with a RDS object when integrated into a part of relation.
  Example: =A1=JD1
- The IMF language introduces terminals and connection points as new entities not covered by the scope of the 81346 standards. The 81346 standards provide a method for allocation of RDS identifiers by use of RDS category codes and resulting RDS strings. This method is expanded to also cover terminals and connection points.
- The purpose and benefits of such a system is to provide an application and context neutral way of creating unambiguous identification of entity within a context.

The individual Context models need to be distinguished by descriptors. These descriptors must be co-managed among the parties that develop context models that need to be directly integrated ❹. The context descriptors is aligned with the system of self-managed descriptors to give a unique descriptor ❸.

TAG labels are centrally managed descriptors used to identify physical objects within a facility ❺.

- TAG is a method for identification of objects in a facility by use of Engineering Numbering System providing classes and syntax. TAG identifications are used both as KEY identificatory across IT applications and for physical marking of equipment in the field.
- TAG identification is Central managed, to avoid duplicates.
- Identification of objects part of facility development projects for all disciplines

Example: 21-PA-101A

There is a need to identify global resources like Type definitions, templates, names that are used across contexts and models, as well as the individual aspect objects. For this purpose IMF adopts globally unique identifiers that can be used across context and application for accessing and use of resources ❻.

6.4.2 Mapping to Application IDs Outside of the IMF

Engineering tools and engineering registers in legacy systems need to work alongside the IMF model(s). This requires mapping and linking mechanisms, as illustrated in Figure 28, defining how relative identifiers within a context are related to applications key identifications and how objects part of a context can be exchanged along and across value chains.

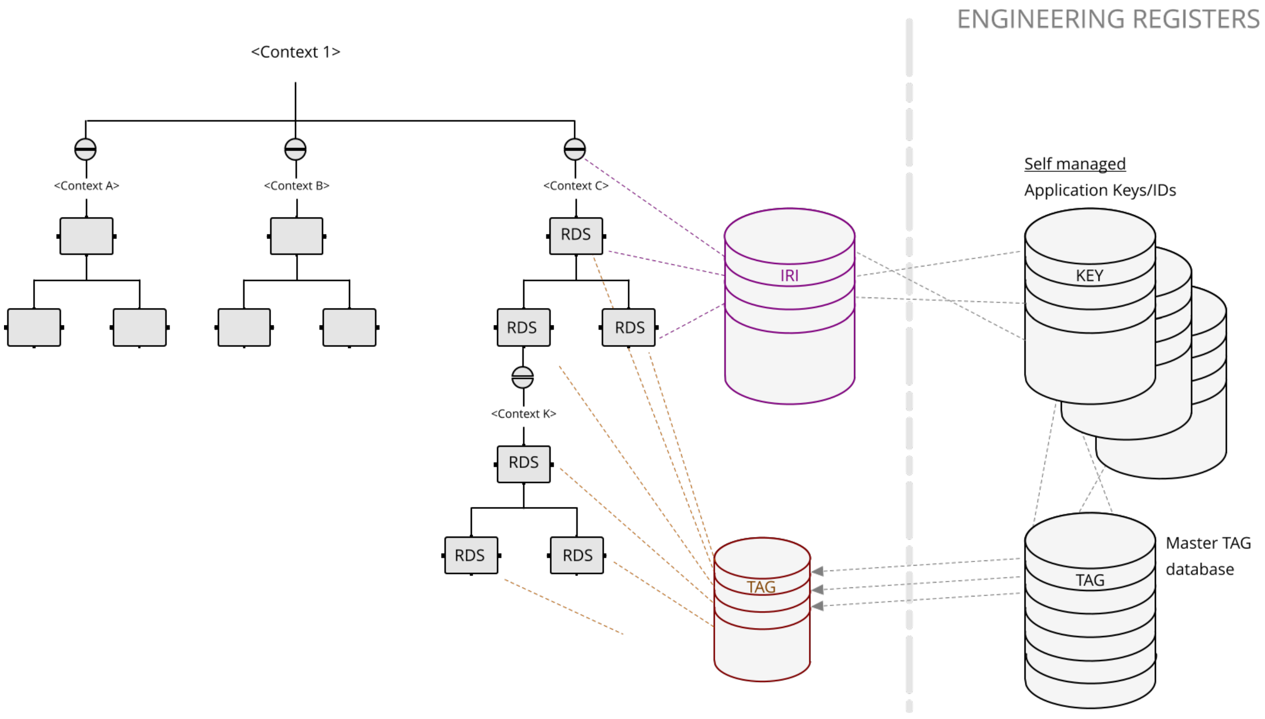


Figure 28: Mapping to application IDs outside of the IMF.

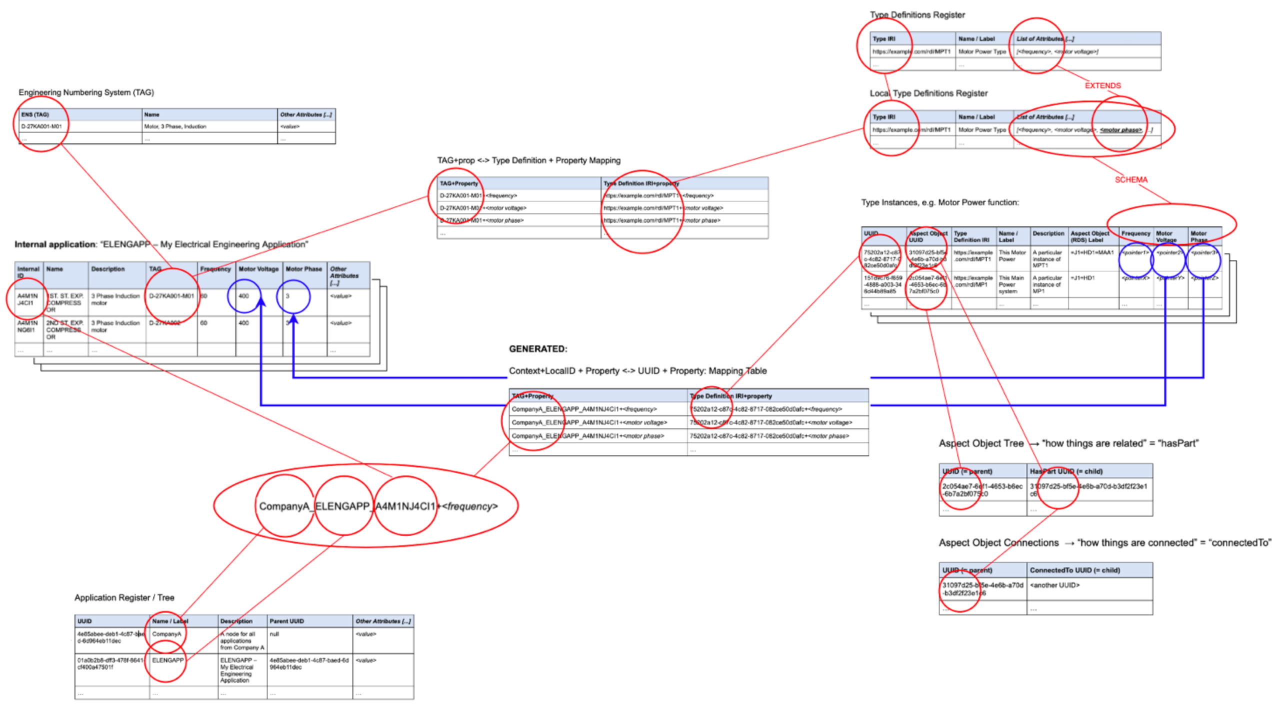
6.4.3 Characterization of the various descriptors and identifiers

To better understand the need for identifiers and their mapping, let's start with an example based on a simple data set.

On the left side, we have a simple example of an internal application containing date using internal IDs (local application keys) as well as a references to engineering numbering systems (TAGs in our case). The idea here is to create a mapping to the values from the vast number of internal applications to one model using IMF techniques. The important aspect of this work is to create a method for generating mapping to the data in internal applications (as opposed to copying the data from those sources) and maintaining the mapping over time by being able to generate and regenerate those mappings based on the IMF model.

This example shows how fragmented information across many applications and using various types of identifiers can be mapped together to provide a unified view to the existing data across domains and the supply chain.

The way we use, exchange, and expose data poses requirements for different classes of descriptors and identifiers.



Type Definitions Register
EXTENDS
Local Type Definitions Register
SCHEMA
Engineering Numbering System (TAG)
TAG+prop <-> Type Definition + Property Mapping
Type Instances, e.g. Motor Power function:
Internal application: "ELENGAPP – My Electrical Engineering Application"
GENERATED:
Context+LocalID + Property <-> UUID + Property: Mapping Table
Aspect Object Tree → "how things are related" = "hasPart"
Aspect Object Connections → "how things are connected" = "connectedTo"
CompanyA_ELENGAPP_A4M1NJ4CI1+<frequency>
Application Register / Tree

Figure 29: A simple example showing generation of mappings between the data in various internal systems and the model based on the IMF approach.

Table 1 gives an overview of those relevant to IMF. Note that an IRI can be constructed as a meaningful path as show in the table example, or it can incorporate a UUID, in which case it is less meaningful, but since the generation of the UUID is by algorithm this allows unmanaged governance.

Table 1: Identifiers and descriptors.

Name	**Purpose**	**Role**	**Governance**	**Classification**	**Example**
TAG number	Describe function	Operator	Centrally managed	Descriptor	21-PA001A
RDS	Describe function and part-of relation	Engineer	Self-managed	Descriptor	=KF4 for one system, and whole path as <facility>=D1=KC2=KF4
Application Key	Unique reference within context of application	IT department	Self-managed	Application ID	IPX-2B-000044
IRI	Unique reference, support ontologies, enable publishing/sharing	IT department	Centrally managed	Global ID	http://data.posccaesar.org/rdl/RDS327239
UUID	Practically unique ID code	IT department	Unmanaged	ID code	123e4567-e89b-12d3-a456-426614174000
Context name	Name of the context of a model, as root	Operators and contractors	Co-managed	Descriptor	“Krafla1201C1”

	name for top node of the model				

6.5 Semantic Verification of Models

Semantic validation of IMF models is performed with respect to different semantic descriptions:

1. IMF OWL ontology and SHACL constraints.
2. Type definitions and their attribute specifications.
3. External ontologies and constraints.

The IMF language is formalized in an OWL ontology and SHACL constraints. The ontology and constraints are used by reasoner and constraint checker (open source) tools to ensure that IMF models are correct and complete with respect to the IMF language meta model, e.g., that System Blocks are related to Terminals using the correct relationships, and that Function System Block may only have other Function System Blocks (and not System Block of other aspects) as parts. This basic and important validation step ensures that IMF models may be correctly parsed and understood as in fact IMF models.

Type definitions allow for an additional level of semantic validation, checking that type definitions are related in a permissible manner, and that instantiations of type definitions are correct. This validation step ensures that the type definitions are correctly interpreted and used by the IMF model, which is also crucial for preparing for the next level of validation.

The final level of semantic validation is performed by translating IMF models to being represented using external semantic descriptions such as the ISO 15926-14 and CHIFOS ontology. The translation from the IMF ontology language to external ontological language is specified by a shared library of OTTR templates and mappings. This validation step ensures that the IMF model is valid with respect to the external ontology and may be understood and processed as represented in the external ontologies' vocabularies.

6.6 Interface to Requirement Management Systems

Digitalization of Requirement management is on a journey where requirements are moving from requirements as text in documents, via sentence-based requirements that can be managed as collections of requirements. The goal is a setting where requirements are managed as data sets that are related to design codes, objects, terminals and interface points in a context model or RDL. Requirement management is based on context, scope, condition and demands. IMF provides context and scope for a facility that enable instantiating of requirements towards scope and condition.

6.7 Mechanisms for Management of Change

As part of SPINE and the Equinor Krafla project was needs, principles and functional requirements for how to do Management of Change developed. The core needs for facility development and operation are.

- **Scoping** – to do precise scoping there is a need to connect models, sub-models, aspect-objects towards project and operational activities.

- **Containment -** There is a need to know when and where physical connections in a facility is planned to be connected, disconnected or reconnected to other parts. To secure containment of fluid, energy, forces and information.
- **Battery limits and boundaries** – in operation and during design of facilities, there is a need to manage technical, contractual and other responsibility interfaces between facilities in a value chain, between technical systems as part of a facility and between components within a product.
- **Lock structures** – as a consequence of the sequencing of engineering, procurement, construction, commissioning, operational and maintenance work there is a need to be able to lock models, objects and data from being updated outside agreed work processes and ownership.
- **Distribution and exchange –** the nature of design, build and operate industry facilities is complex interactions between many stakeholders and this leads to a need for controlled publishing and distribution of models, partial models and belonging objects.
- **CRUD-** the administration of a Facility as an Asset reveal needs with respect to Create, Read, Update and Delete (CRUD) models, parts of models and belonging objects.

The needs were translated into 5 requirement groupings for models and belonging objects.

- **Part of the whole:** It shall be possible to relate the whole or part of a facility model to activities to be able to control what is scope of the activity. Examples of activities are modifications projects, maintenance tasks, turn around stops.
- **Grouping of models**: It shall be possible to group objects into sub models that control the objects part of that sub model. (Models that replace existing diagrams, documents, specifications and datasheets)
- **Persistent identification:** It shall have a protocol for handling persistent identification of the objects.
- **Change distribution and exchange:** It shall define a concept for how changes are made and synchronized across the parties and related to different project phases.
- **Breaking changes**: It shall be possible to implement a lock structure, that control changes (which changes are allowed within one data set, and which changes require changes to the parent - model above (the home of the object) and/or associated objects, like systems, locations, or areas.

The structure of a context model made by use of the IMF method enable the possibility of fulfilling these requirements, since the language provides those elements needed to identify scope and interfaces between context models. The red markup ❶ – identifies the physical motor and connection interfaces. The blue markup ❷ illustrates how to scope a modification to a driving function.

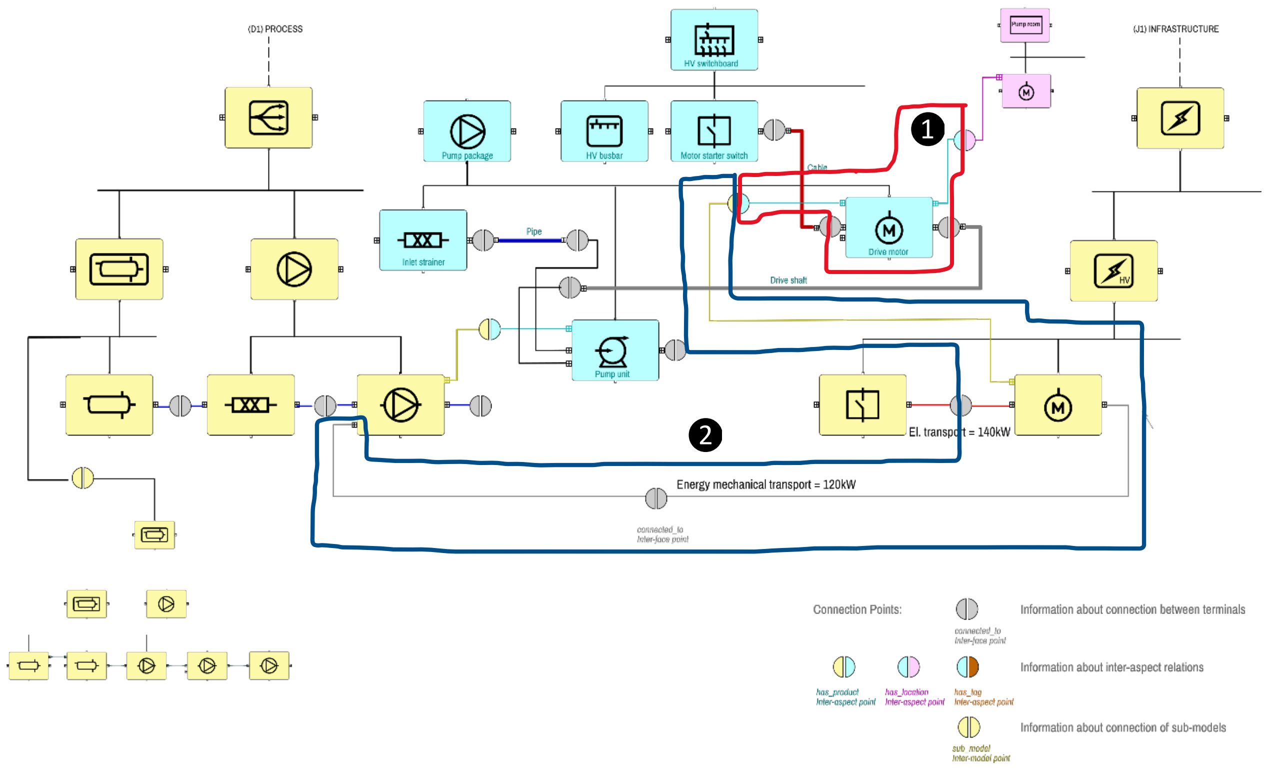

(D1) PROCESS
HV switchboard
Pump room
(J1) INFRASTRUCTURE
Pump package
HV busbar
Motor starter switch
Cable
Drive motor
Inlet strainer
Pipe
Drive shaft
Pump unit
HV
El. transport = 140kW
Energy mechanical transport = 120kW
connected_to inter-face point
Connection Points:
Information about connection between terminals
Information about inter-aspect relations
has_product inter-aspect point
has_location inter-aspect point
has_tag inter-aspect point
Information about connection of sub-models
sub_model inter-model point
1
2

Figure 30: Scope and interfaces between models.

7 THE IMF ECO-SYSTEM

[IMF will be implemented in the context of applications that serve different parts of an eco-system. This chapter will address central component of the eco-system.]

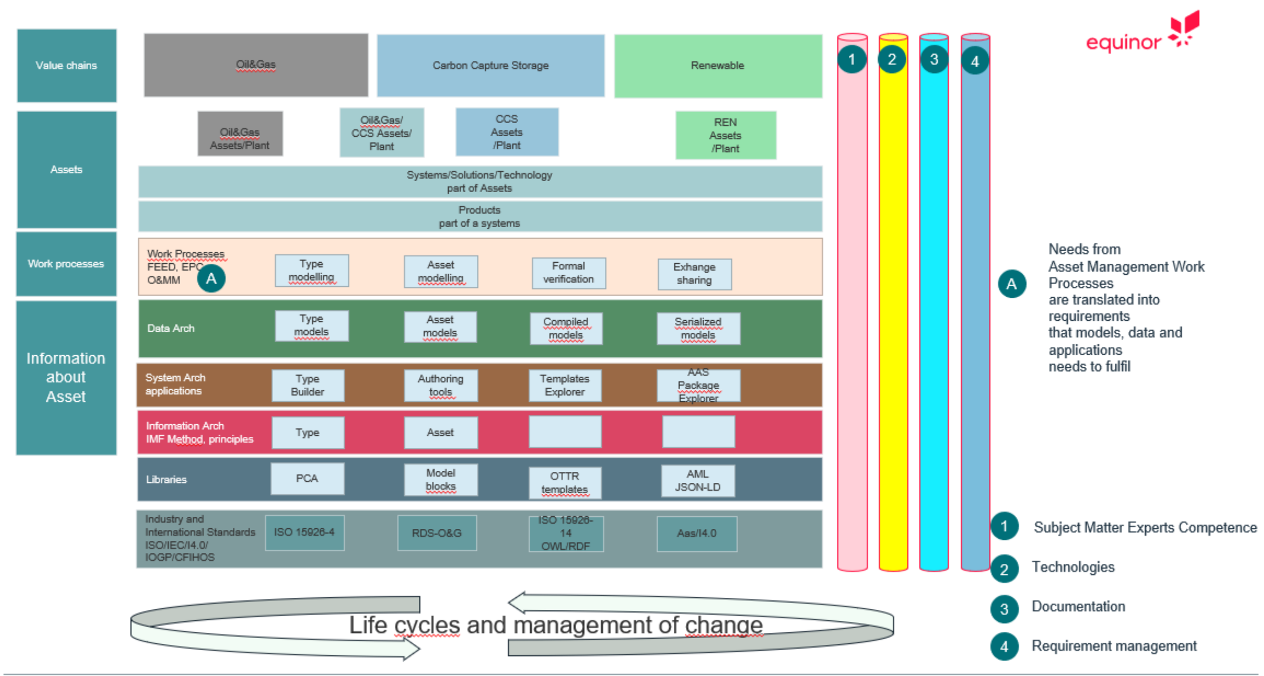


Figure 31: The IMF eco-system.

7.1 Authoring Tools

[This section will cover the learnings from the development of Mimir.]

7.2 Representation Formats

[This section specifies representation formats that exploit well-proven W3C technologies and support the sharing-by-publishing paradigm.]

7.3 Query and Analyses

[Context models and ontologies can be utilized by querying and analyzing query answers. This insight is explained in this section.]

7.4 Serialization and Data Exchange

[Exchange of data must follow strict formats and protocols. This section specifies exchange formats that exploit the W3C protocols.]

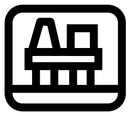
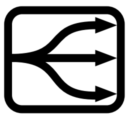
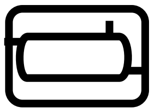
Appendix A – Example of Attributes Divided by Aspect

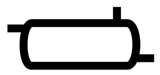
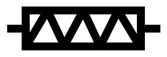
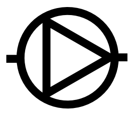
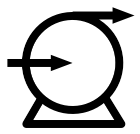
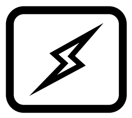
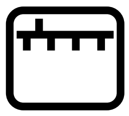
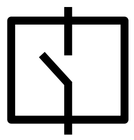
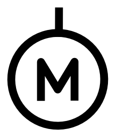
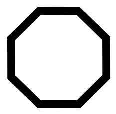
Table 2: Attributes for a pump unit 40PA001A.

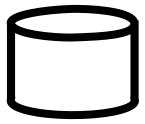
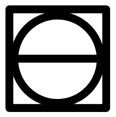
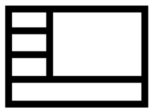
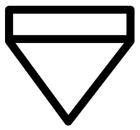
Function	Product	Location	Installed
Operating temp.= -3/50/55 C	Design temp.= -10/50/85 C	Length= 2.34m	Manufacturer= ABB
(Flow)Capacity= 1444 m3/h	Design Capacity= 1800m3/h	Width= 0.89m	Weight, pump= 3651 kg
Differential pressure= 7.5 bar	Differential pressure= 8.99 bar	Height= 0.78m	Weight, motor= 3552 kg
Discharge pressure= 18.2 barg	Max pressure= 41barg@85C	[part of= AC120]	Serial nº motor= ABB-12-124-555-005
Viscosity= 1.3 cP	Speed= 1740 rpm	[Site temp.= -3/22 C]	Serial nº pump= Flowserve-33-46526-35-006
SpGravity= 1.05	Power, drive= 460 kW	[Location= Outdoor]	
	Ex.protection= Ex e px	[Area class= Zone 2 T3]	
	Power, motor= 500 kW Voltage, motor= 6 kV		
	Design code, pump= TR 2910-04 to API 610		

Appendix B - Symbols

Table 3: An explanation of the illustrative symbols used in this document to indicate the purpose/activity of aspect objects shown in model figures.

Symbol	Purpose / activity	Description
	n/a	Complex offshore facility
	Separate	System for separating a stream
	Separate	System for gravity-based separation of a stream, including auxiliary systems

	Separate	System for gravity-based three-phase separation
	Mixing	System for mixing a non-homogenous stream
	Pumping	System for increasing flow and/or pressure of a stream
	Pumping	System for increasing flow and/or pressure of a stream by centrifugal principle
	n/a	System for managing electrical energy
	Distributing	System for distributing a stream of electrical energy
	Switching	System for switching/controlling a stream of electrical energy
	Driving	System for driving a system by means of converting electrical energy to mechanical energy
	n/a	System for managing logic solving and control

	Storing	System for storing
	Measuring	System for measuring a quality
	Control	System for applying logic on inputs to result in a controlling output
	Shutdown	System for applying logic on inputs to result in a shutdown output

APPENDIX C – LIST OF STANDARDS

[TODO]

APPENDIX D – IMF OWL ONTOLOGY

The IMF OWL ontology is specified by three ontologies that contain import statements between each other. These three ontologies are called the IMF top ontology, the IMF metamodel ontology and IMF aspects ontology. These are collected in a central ontology called the IMF ontology. Figure 32 displays the different ontology documents and the import relations that exist between these.

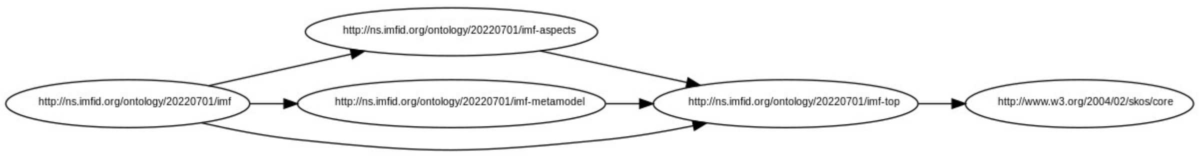

http://ns.imfid.org/ontology/20220701/imf-aspects
http://ns.imfid.org/ontology/20220701/imf
http://ns.imfid.org/ontology/20220701/imf-metamodel
http://ns.imfid.org/ontology/20220701/imf-top
http://www.w3.org/2004/02/skos/core

Figure 32. IMF ontology import hierarchy

The ontologies are listed below in RDF Turtle serialization format. The prefixes used are listed once for brevity in a separate section.

IMF Ontology

This is the main IMF ontology and its main point of entry. This ontology document imports all other IMF ontology documents.

```
<http://ns.imfid.org/imf> a owl:Ontology ;

    owl:versionIRI <http://ns.imfid.org/ontology/20220701/imf> ;

    owl:imports

      <http://ns.imfid.org/ontology/20220701/imf-top> ,

      <http://ns.imfid.org/ontology/20220701/imf-metamodel> ,

      <http://ns.imfid.org/ontology/20220701/imf-aspects> ;

    skos:prefLabel "Information Modelling Framework Ontology " ;

    skos:altLabel "IMF ontology" ;

    skos:scopeNote """


      This is the main IMF ontology and its main point of entry.  This

      ontology document imports all other IMF ontology documents, each

      of which focus on one particular part of the ontology:


      1. IMF Top defines the very high-level classes and properties of

      the IMF ontology, such as AspectObject.


      2. IMF Metamodel defines the metamodel for aspect objects and

      their permissible properties.


      3. IMF Aspects defines the aspects that are available to aspect

      objects.


    """;


    .
```

IMF Top Ontology

This ontology defines the very core classes and relations of the IMF ontology to provide a structure for ontologies that extend this ontology.

```
imf:Aspect

    a owl:Class ;

    skos:example "Function, Location, Product are examples of aspects." ;

    skos:prefLabel "Aspect" .


imf:AspectObject
```

```
    a owl:Class ;
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:onClass imf:Aspect ;
        owl:onProperty imf:hasAspect ;
        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger
    ] ;
    skos:definition """

    An aspect object describes an asset from a specific viewpoint (or
    aspect).

    An aspect object has a single aspect, we say that the aspect
    object \"is of\" this aspect.

    """ ;
    skos:prefLabel "Aspect Object" .

imf:MultiAspectObject
    a owl:Class ;
    skos:prefLabel "Multi-Aspect Object" ;
    skos:scopeNote """

  A multi-aspect object is a collection of aspect objects that
  describe an asset from different viewpoints (aspects).

  There are currently no formal restrictions on the aspect objects
  that a multi-aspect object groups together. Hence, a multi-aspect
  object may collect aspect objects of the same aspect, or of
  different aspects.

  """ .

imf:associativeRelation
    a owl:ObjectProperty ;
    rdfs:domain imf:AspectObject ;
    rdfs:range imf:AspectObject ;
    rdfs:subPropertyOf skos:related ;
    owl:propertyDisjointWith imf:hierachicalRelation ;
    skos:definition "A generic associative relation." ;
    skos:prefLabel "associative relation" ;
    skos:scopeNote """
```

```
      This relation is used to enforce that subproperties respect this

      relation's definition. This must be enforced by introducing

      class axioms that locally further restrict the domain and range

      of the relation.


    """ .


imf:hasAspect

    a owl:ObjectProperty ;

    rdfs:domain imf:AspectObject ;

    rdfs:range imf:Aspect ;

    skos:definition "Relates an aspect object to its aspect." ;

    skos:prefLabel "has aspect" .


imf:hasAspectObject

    a owl:ObjectProperty ;

    rdfs:domain imf:MultiAspectObject ;

    rdfs:range imf:AspectObject ;

    skos:definition """


    Relates the multi-aspect object to the aspect object(s) it

    collects.


    """ ;

    skos:prefLabel "has aspect object" .


imf:hierarchicalRelation

    a owl:IrreflexiveProperty, owl:ObjectProperty ;

    rdfs:domain imf:AspectObject ;

    rdfs:range imf:AspectObject ;

    rdfs:subPropertyOf skos:semanticRelation ;

    skos:definition "A generic hierachical relation that may be used to represent a breakdown structure." ;

    skos:prefLabel "hierarchical relation" ;

    skos:scopeNote """


      This relation is used to enforce that subproperties respect this

      relation's definition. This must be enforced by introducing

      class axioms that locally further restrict the domain and range

      of the relation.


    """ .
```

```
imf:intraAspectRelation
    a owl:ObjectProperty ;
    rdfs:domain imf:AspectObject ;
    rdfs:range imf:AspectObject ;
    rdfs:subPropertyOf skos:semanticRelation ;
    skos:definition "A generic relation between aspect objects of the same aspect" ;
    skos:prefLabel "intra-aspect relation" ;
    skos:scopeNote """

      This relation is used to enforce that subproperties respect this
      relation's definition. This must be enforced by introducing
      class axioms that locally further restrict the domain and range
      of the relation.

    """ .

<http://ns.imfid.org/imf-top>
    a owl:Ontology ;
    owl:imports <http://www.w3.org/2004/02/skos/core> ;
    owl:versionIRI <http://ns.imfid.org/ontology/20220701/imf-top> ;
    skos:altLabel "IMF top ontology" ;
    skos:prefLabel "Information Modelling Framework Ontology: Top Ontology " ;
    skos:scopeNote """

      This ontology defines the very core classes and relations of the
      Information Modelling Framework (IMF) to provide a structure for
      ontologies that extend this ontology.

    """ .

[]
    a owl:AllDisjointClasses ;
    owl:members (imf:Aspect
        imf:AspectObject
        imf:MultiAspectObject
    ) .
```

IMF Metamodel Ontology

This ontology defines IMF's meta model which defines how IMF models are represented.

```
imf:AspectObject
    a owl:Class ;
    owl:equivalentClass [
        a owl:Class ;
        owl:unionOf (imf:SystemBlock
            imf:Terminal
            imf:InterfacePoint
        )
    ] ;
    skos:scopeNote """

    Each aspect object is either a system block or a terminal.""" .

imf:InputTerminal
    a owl:Class ;
    rdfs:subClassOf imf:Terminal, [
        a owl:Restriction ;
        owl:allValuesFrom imf:InputTerminal ;
        owl:onProperty imf:isPartOf
    ] ;
    skos:altLabel "Input" ;
    skos:definition """

    A terminal whose default function is to recieve input for its
    system.

  """ ;
    skos:prefLabel "Input Terminal" .

imf:InterfacePoint
    a owl:Class ;
    rdfs:subClassOf imf:AspectObject, [
        a owl:Restriction ;
        owl:onClass imf:OutputTerminal ;
        owl:onProperty imf:hasOutputTerminal ;
        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger
    ], [
        a owl:Restriction ;
        owl:onClass imf:InputTerminal ;
        owl:onProperty imf:hasInputTerminal ;
        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger
    ] ;
    skos:altLabel "Connection", "Junction", "Transport" ;
    skos:definition """
```

```
     A simple type of system block that has exactly one input and one
     output. No transformation of the input is performed.

     The connection, in effect, merges the two connected
     terminals. The connection itself has no function or spatial
     significance.

  """ ;
    skos:prefLabel "Interface Point" .

imf:OutputTerminal
    a owl:Class ;
    rdfs:subClassOf imf:Terminal, [
        a owl:Restriction ;
        owl:allValuesFrom imf:OutputTerminal ;
        owl:onProperty imf:isPartOf
    ] ;
    skos:altLabel "Output" ;
    skos:definition """

    A terminal whose default function is to give output for its
    system.

  """ ;
    skos:prefLabel "Output Terminal" .

imf:SystemBlock
    a owl:Class ;
    rdfs:subClassOf imf:AspectObject, [
        a owl:Restriction ;
        owl:allValuesFrom imf:SystemBlock ;
        owl:onProperty imf:isPartOf
    ] ;
    skos:definition """

    A system is a processing (black) box. It processes the input to
    output, possibly changing the state of what is processed.

    Transforms the input from its input terminals to the output to its
    output terminals. The relation between the input and output
    terminals may be complex and may be further described by
    subsystems which are related by hasPart.


    A system may have multiple input and output terminals. A system

    with zero terminals is uncommon.


  """ ;

    skos:prefLabel "System Block" .


imf:Terminal

    a owl:Class ;

    rdfs:subClassOf imf:AspectObject ;

    owl:equivalentClass [

        a owl:Class ;

        owl:unionOf (imf:InputTerminal

            imf:OutputTerminal

        )

    ] ;

    skos:altLabel "Channel", "Input/Output", "Port" ;

    skos:definition """


    A port or boundry point through which a system block can interact

    and communicate with the world outside the system, receiving input

    and giving output.


  """ ;

    skos:editorialNote """


     [2022-03-11 Fri] Need to represent the type of a terminal, and

     constraints on connections between typed terminals: material,

     information, energy, structural.


   """ ;

    skos:prefLabel "Terminal" .


imf:directlyPrecedes

    a owl:ObjectProperty ;

    rdfs:domain imf:InputTerminal ;

    rdfs:range imf:OutputTerminal ;

    rdfs:subPropertyOf imf:precedes ;

    skos:definition """


    An immediate, non-transitive, precedes relation between input
    terminals and output terminals *of the same system* such that

    (parts of) the input to the input terminal is processed by the

    system to produce (parts of) the output of the output terminal.


  """ ;

    skos:prefLabel "directly precedes" .


imf:hasInputTerminal

    a owl:ObjectProperty ;

    rdfs:range imf:InputTerminal ;

    rdfs:subPropertyOf imf:hasTerminal ;

    skos:definition "The relation between a system and its input terminals." ;

    skos:prefLabel "has input terminal" .


imf:hasOutputTerminal

    a owl:ObjectProperty ;

    rdfs:range imf:OutputTerminal ;

    rdfs:subPropertyOf imf:hasTerminal ;

    skos:definition "The relation between a system and its output terminals." ;

    skos:prefLabel "has output terminal" .


imf:hasPart

    a owl:ObjectProperty ;

    rdfs:subPropertyOf imf:hierarchicalRelation, imf:intraAspectRelation, skos:narrower ;

    owl:inverseOf imf:isPartOf ;

    skos:altLabel "has part" ;

    skos:prefLabel "has child" .


imf:hasTerminal

    a owl:ObjectProperty ;

    rdfs:domain imf:SystemBlock ;

    rdfs:range imf:Terminal ;

    rdfs:subPropertyOf imf:associativeRelation, imf:intraAspectRelation ;

    skos:definition "The relation between a system and its terminals." ;

    skos:prefLabel "has terminal" .


imf:isConnectedTo

    a owl:ObjectProperty ;

    rdfs:domain imf:OutputTerminal ;

    rdfs:range imf:InputTerminal ;

    rdfs:subPropertyOf imf:associativeRelation, imf:intraAspectRelation, imf:precedes ;

    skos:definition """
    The relation between two terminals that are connected. The output

    of the output terminal is given as input to the input terminal.


  """ ;

    skos:prefLabel "is connected to" .


imf:isPartOf

    a owl:FunctionalProperty, owl:ObjectProperty ;

    rdfs:domain imf:AspectObject ;

    rdfs:range imf:AspectObject ;

    rdfs:subPropertyOf imf:hierarchicalRelation, imf:intraAspectRelation, skos:broader ;

    skos:altLabel "is part of" ;

    skos:definition """


    An aspect object is placed in a tree-shaped breakdown structure

    using isPartOf/hasPart relationships. An aspect object may have

    a single parent (the root object has no parent) and possibly

    multiple children. This is an abstraction mechanism. The children

    of an aspect object provide a more detailed description of (parts

    of) its parent.


  """ ;

    skos:prefLabel "has parent" .


imf:precedes

    a owl:ObjectProperty, owl:TransitiveProperty ;

    rdfs:domain imf:Terminal ;

    rdfs:range imf:Terminal ;

    rdfs:subPropertyOf imf:associativeRelation, imf:intraAspectRelation ;

    skos:definition """


    A relation between terminals that represents the \"flow\" of

    input/output between terminals, both terminals of the same system

    block (represented by the subproperty directlyPrecedes) and of

    different system blocks (represented by the subproperty

    isConnectedTo).


  """ ;

    skos:prefLabel "precedes" .


<http://ns.imfid.org/imf-metamodel>
```

```
    a owl:Ontology ;

    owl:imports <http://ns.imfid.org/ontology/20220701/imf-top> ;

    owl:versionIRI <http://ns.imfid.org/ontology/20220701/imf-metamodel> ;

    skos:altLabel "IMF metamodel ontology" ;

    skos:prefLabel "Information Modelling Framework Ontology: Metamodel Ontology " ;

    skos:scopeNote """


      This ontology defines IMF's meta model which defines how

      IMF models are represented.


    """ .


[]

    a owl:AllDisjointClasses ;

    owl:members (imf:InputTerminal

        imf:OutputTerminal

    ) .


[]

    a owl:AllDisjointClasses ;

    owl:members (imf:SystemBlock

        imf:Terminal

        imf:InterfacePoint

    ) .
```

IMF Aspect Ontology

This ontology defines IMF's aspects.

```
imf:Aspect

    a owl:Class ;

    skos:scopeNote """


    Each aspect is associated with a class of the aspect objects that

    have that aspect, e.g,. imf:FunctionAspectObject is the class of

    aspect objects with the aspect imf:FunctionAspect. These classes

    are used to specify permissible relationships between aspect

    objects according to their aspect.""" .


imf:FunctionAspect

    imf:color "#FFFF00" ;

    imf:prefix "=" ;

    a imf:Aspect .
```

```
imf:FunctionAspectObject
    a owl:Class ;
    rdfs:subClassOf imf:AspectObject, [
        a owl:Restriction ;
        owl:allValuesFrom imf:FunctionAspectObject ;
        owl:onProperty imf:intraAspectRelation
    ] ;
    owl:equivalentClass [
        a owl:Class, owl:Restriction ;
        owl:hasValue imf:FunctionAspect ;
        owl:onProperty imf:hasAspect
    ] .

imf:InstalledAspect
    imf:color "#FFFFFF" ;
    imf:prefix "::" ;
    a imf:Aspect .

imf:InstalledAspectObject
    a owl:Class ;
    rdfs:subClassOf imf:AspectObject, [
        a owl:Restriction ;
        owl:allValuesFrom imf:InstalledAspectObject ;
        owl:onProperty imf:intraAspectRelation
    ] ;
    owl:equivalentClass [
        a owl:Class, owl:Restriction ;
        owl:hasValue imf:InstalledAspect ;
        owl:onProperty imf:hasAspect
    ] .

imf:LocationAspect
    imf:color "#FF00FF" ;
    imf:prefix "+" ;
    a imf:Aspect .

imf:LocationAspectObject
    a owl:Class ;
    rdfs:subClassOf imf:AspectObject, [
        a owl:Restriction ;
        owl:allValuesFrom imf:LocationAspectObject ;
        owl:onProperty imf:intraAspectRelation
    ] ;
    owl:equivalentClass [
        a owl:Class, owl:Restriction ;
        owl:hasValue imf:LocationAspect ;
        owl:onProperty imf:hasAspect
    ] .

imf:ProductAspect
    imf:color "#00FFFF" ;
    imf:prefix "-" ;
    a imf:Aspect .

imf:ProductAspectObject
    a owl:Class ;
    rdfs:subClassOf imf:AspectObject, [
        a owl:Restriction ;
        owl:allValuesFrom imf:ProductAspectObject ;
        owl:onProperty imf:intraAspectRelation
    ] ;
    owl:equivalentClass [
        a owl:Class, owl:Restriction ;
        owl:hasValue imf:ProductAspect ;
        owl:onProperty imf:hasAspect
    ] .

imf:hasAspect
    a owl:ObjectProperty .

imf:interAspectRelation
    a owl:ObjectProperty ;
    rdfs:domain imf:AspectObject ;
    rdfs:range imf:AspectObject ;
    rdfs:subPropertyOf skos:related ;
    skos:definition """

      Relates aspects objects of different aspects.

    """ .

imf:intraAspectRelation
    a owl:ObjectProperty .
```

```
<http://ns.imfid.org/imf-aspects>

    a owl:Ontology ;

    owl:imports <http://ns.imfid.org/ontology/20220701/imf-top> ;

    owl:versionIRI <http://ns.imfid.org/ontology/20220701/imf-aspects> ;

    skos:altLabel "IMF aspects ontology" ;

    skos:prefLabel "Information Modelling Framework Ontology: Aspects Ontology " ;

    skos:scopeNote """


      This ontology defines IMF's central aspects.


    """ .


[]

    a owl:AllDifferent ;

    owl:members (imf:FunctionAspect

        imf:LocationAspect

        imf:ProductAspect

        imf:InstalledAspect

    ) .
```

PREFIXES

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix imf: <http://ns.imfid.org/imf#> .

@prefix pav: <http://purl.org/pav/> .

@prefix o-rdf: <http://tpl.ottr.xyz/rdf/0.1/> .

@prefix o-owl-rstr: <http://tpl.ottr.xyz/owl/restriction/0.1/> .

@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix skos: <http://www.w3.org/2004/02/skos/core#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix ex: <http://example.com#> .

@prefix sh: <http://www.w3.org/ns/shacl#> .

@prefix o-imf: <http://ns.imfid.org/templates/> .

@prefix ottr: <http://ns.ottr.xyz/0.4/> .

@prefix o-owl-ma: <http://tpl.ottr.xyz/owl/macro/0.1/> .

@prefix o-owl-ax: <http://tpl.ottr.xyz/owl/axiom/0.1/> .

@prefix shsh: <http://www.w3.org/ns/shacl-shacl#> .
```