Title**:**

The Information Modelling Framework Manual

Project:

Equinor Spine Project
SIRIUS project 4102 Asset Information Modelling
NOAKA Digital Collaboration

Classification:	Owner:
Open	Magnus Knædal

Status: Revision 2.0. Ready for beginning implementation and development of DNV RP.

Author(s)/Source(s):

Erlend Fjøsna, Torleif Saltvedt, Arild Waaler, Magnus Knædal, Vetle Koppergård, Lillian Hella

Martin Georg Skjæveland, Rustam Mehmandarov, Mihaly Fekete, Baifan Zhou

Mission:

Develop documentation in form of a manual and an IT specification that together answer what a company must do to implement and use the IMF concept. The result shall form the necessary basis to create a DNV Recommended Practice.

Remarks:

This document is the second draft version of The Information Modelling Framework Manual. The development of the manual will continue after this release.

Valid from:	Updated:
21.11.2022	21.11.2022

Preface

The development of the Information Modelling Framework (IMF) was progressed through the READI[1] Joint Industry Project, resulting in an IMF Concept document issued in March 2021. Following this, the development continued as part of the Equinor's Krafla and Wisting project, and then was extended to include the partners of the NOAKA Digital Cooperation, Equinor and Aker BP. The SIRIUS Centre at the University of Oslo has also contributed to the IMF work. In parallel with the continued development of the IMF, the implementation of IMF was pioneered by the Krafla project together with Aibel, being the engineering contractor. Learnings from this piloting has contributed significantly to enhancing IMF and documenting IMF in the form of a Reference Manual that comprises 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] https://readi-jip.org/

Contents

List of Figures

List of Tables

Terms and Abbreviations

Terms and their explanations are given in Table 1, while abbreviations and their meaning are given in Table 2**Error! Reference source not found.**. Terms related to the formalization of IMF can be found in Chapter 4.

Table 1: Terms and explanations.

Term	**Explanations**
Artifact	Physical objects or software, with functions realized in activities, and situated in spatial locations
Aspect	An aspect is a particular way of viewing a Facility Asset. Different aspects have no overlap in information.
Attribute	A quality or characteristic that someone or something has. Synonym with property.
Capital Value Process	Decision process for investment projects.
Data Engineer	An expert on the flow of data between IT systems and applications.
Design Code	Requirements to standardized engineering solution.
Engineering Register	IT application holding attribute data.
Facility Asset	An industrial plant, or part of it. A Facility Asset can be intended during design and actual when it has been installed.
Facility Asset Information Model	A digital representation of a Facility Asset, also called an IMF Model. IMF Models can be integrated into a larger IMF Model. In addition, IMF Models may be extracted from a larger IMF Model.
Hierarchy	The hierarchy dimension stating elements being *part of* higher-level elements, i.e. a breakdown.
IMF Purpose	An intended activity that describes the essence of an IMF Type.
IMF Type	A library resource used as starting point for instantiation of Facility Asset Information Model objects.
IMF Type Library	A library of IMF Types that is a shared resource in the industry.
Information Model	An Information Model is a structure of information objects and relations between them that is designed to capture knowledge.
Knowledge Engineer	An expert on knowledge representation including semantic technology.
Media	The type of content of a stream, e.g. information, energy, material or force.
Ontology	Technology that is used to represent knowledge in a formal way.
Reference Data Library	A managed collection of reference data [1], e.g. accessible through an endpoint. Reference Data Libraries provide industry shared resources, such as vocabularies, symbols, units.
Reference Designation	Identifier of a specific object with respect to the system of which the object is a constituent, based on one or more aspects of that system. [2]
Stream	The flow of media between systems.
Subject Matter Expert	An expert within a certain engineering discipline, for example a process or electrical engineer.

System	Combination of interacting elements organized to achieve one or more stated purpose. [3]
System Engineer	An engineer focusing on how to design, integrate and manage complex system over their life cycles.
TAG	A human-readable code for identifying elements of a Facility Asset.
Topology	The topology dimensions stating how elements are *connected to* each other.

Table 2: Abbreviations and their meaning

Abbreviation	**Meaning**
API	Application Programming Interface
CVP	Capital Value Process
DG	Decision Gate
IMF	Information Modelling Framework
OWL	Web Ontology Language
RDL	Reference Data Library
RDS	Reference Designation System
SME	Subject Matter Expert

1 Introduction

The Information Modeling Framework (IMF) is a method, a framework, and a language that allows creating a formal description of a Facility Asset, using common industry reference data libraries containing definitions of elements that are frequently re-used. The resulting Information Model of the Facility Asset contains information in a format which is readable to humans as well as to computers.

1.1 Problem Statement

Facility Assets in the energy industry are characterized by an increasing complexity. There is a need for better communication between organizations, people, and IT systems. Actors in the industry use different methods, tools, and work processes to develop a Facility Asset. This misalignment is seen:

- Along with the Capital Value Process (CVP).
- Along the supply chain.
- Between Disciplines and Systems within the same Facility Asset.

The consequences of this, is loss of information are risk of safety and quality breaches, need for work to be done twice, a lot of manual mapping, reduced possibilities for re-use of concepts and design, lock-in in a portfolio of applications, and a lot of company-specific requirements that are tuned to fulfill needs for a certain portfolio of applications.

Figure 1 summarizes the problem of today's way of working. The figure illustrates the *logical* flow of value creation during an industrial investment and development project, whereas the actual execution schedule will have many overlaps and iterations that are intentionally left out.

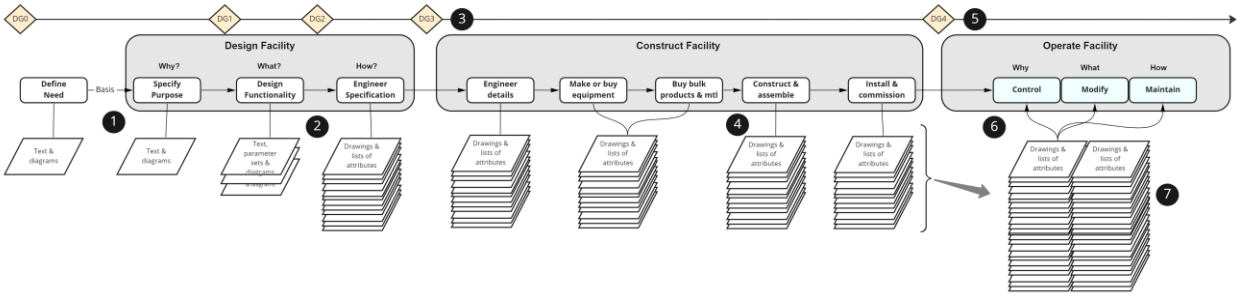


Figure 1: Current documentation practice during an industrial investment and development project.

When a Facility Asset is developed today, the work begins by defining the overall requirements and functionality (DG0). The result is typically contained in a few documents, which means that at this stage a holistic description is feasible ❶. As the work progresses into the design phase of the Facility Asset, more specialization and discipline expertise are necessary to develop the details (DG1 and DG2). Since the way of working is document-based, the result is an increasing number of documents. This leads to a fragmentation of information spread across documents, due to the inherent features of their format. Because of this fragmentation, it becomes gradually impossible to keep up a holistic description of the Facility Asset. The result is extensive interface coordination between the discipline experts ❷.

When the investment decision is made to execute the construction of the Facility Asset (DG3) ❸, the number of documents fabricated grows exponentially as the supply chain involving both contractors, suppliers, and manufacturers ramp up their deliveries for construction, installation, and commissioning

of the Facility Asset ❹. At this stage, and likely to have started earlier in the CVP, a lot of information in documents is duplicated, resulting in several sources of the same information. The consequence of this is labor-intensive work to prevent safety and quality breaches.

When the operation (DG4) and handover to the Operator takes place ❺, the fragmentation has grown to an 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 often results in a need to 're-engineer' the solution to establish the complete picture necessary to operate, control, maintain, and later do modifications on the Facility Asset ❻. Going into the operation phase of the Facility Asset, the Operator is left with an enormous amount of information that is hard to maintain ❼.

It should also be mentioned that a lack of overview at the 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.

1.2 Objectives

The objective of the Information Modelling Framework is to enable a transition from the current documentation practice discussed in Section 1.1, to the use of *Information Models*, as a way of capturing and expressing information about Facility Assets.

To achieve the transition, IMF must:

- Support incremental implementation. Thus, handle both a new Information Model practice and the transition towards it, yet not require existing applications and tools to be replaced.
- Be scalable across disciplines, work processes, and the value chain. Thus, support any level of complexity and level of detail, depending on the need and where in the process the SMEs are.
- Be made such that the Subject Matter Experts (SMEs) themselves are users of the framework. Thus, support and allow existing discipline design workflows.
- Utilize and promote development of shared libraries and build on existing standards.
- Support open protocols for exchange of Facility Asset information and provide a format which enables automated verification and augmented engineering extensions.

1.3 Scope

Specifying IMF involves specifying the Information Modelling Framework itself, as well as specifying how it is intended to work with and interface to external application and resources, as illustrated in Figure 2. The scope of this document therefore includes both the IMF as well as its external interfaces.

These are ❶ Engineering Tools and Engineering Registers that may hold attributes and objects that are part of the Information Model describing the Facility Asset. ❷ Modelling Tools for creating, modifying, and sharing IMF models. ❸ Reference Data Libraries that contain the resources from which IMF models are built. ❹ Semantic verification mechanisms and tools that offer computer-based validation and verification of the IMF model.

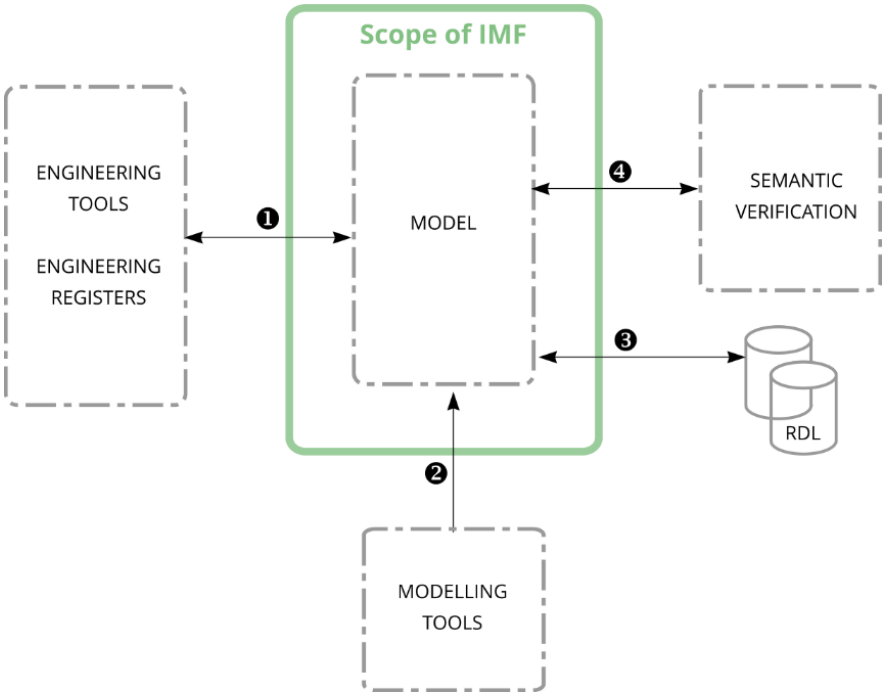


Figure 2: Framing and scoping of the IMF.

Covering both the IMF as well as how it interacts with external systems means that the following is included in the scope:

- A definition of the IMF language, including vocabulary and rules.
- A description of interfaces to external systems being part of the ecosystem.
- A definition of modelling methodology.
- A definition of methodology for creating or modifying modelling building blocks.

1.4 Way Forward

The following topics in the scope of IMF are not fully covered in this version (2.0), but will be covered in later versions of this document or in separate documents:

- Concepts for Management of Change.
- Concepts for ID management.
- Concepts for specifying mappings from an IMF Model to engineering registers.
- Concepts for specifying relationships between descriptors, i.e., between engineering numbering and reference designations.
- Templates mechanism for translating IMF Models to ontology allowing for semantic verification.
- Mechanisms for using RDL resources.
- Specification for implementation of IMF in IT systems.

1.5 Industry Value of the Work

This document and the development of the Information Modelling Framework serves as a reference for:

- Understanding how to implement a new way of working using Information Models.

- Providing the basis for creating a DNV Recommended Practice on the subject of Information Modelling.
- Implementing tools and applications for Information Modelling.
- Alignment of industry Reference Data Libraries.
- Implementation of cross-industry interoperability based on IEC/ISO 81346-1 [2] (structuring principles) and 15926-14 [4] (reference data and verification).
- Computer-based exchange, automatization, and verification of Information Models.
- Contractual requirements for delivery of Facility Asset Information Models. See Appendix B.

1.6 Outline and Readers Guide of this Document

This document will introduce IMF and how IMF can be used to enable new ways of working for engineering of Facility Assets, from the design begins and until the asset facility is decommissioned. In Chapter 2 background knowledge for understanding the IMF concept is given. A brief introduction to the IMF concept and how this solves the problem statement in Section 1.1 is presented in Chapter 3. Chapter 4 formalizes the IMF Language and gives an overview of the different elements, relations, and rules in IMF. Chapter 5 is a guide for how to use IMF to model Facility Assets. Chapter 6 is a guide of how to create the building blocks in the IMF Models. An overview of the foreseen eco-system of applications is given in Chapter 7. At the end of the document a number of appendices covering topics of IMF that might be of interest for certain readers are provided.

For readers that are intended users of IMF (i.e., use IMF for modelling Facility Assets) the entire document excluding appendices should be read in order to get the necessary understanding of IMF and recommendation to how to start modelling Facility Assets.

For readers wanting an overall understanding of the concept of IMF, reading Chapters 1, Background Knowledge2 & 3 should be sufficient. Optionally, Chapter 7 7can be read to get an understanding of how an implementation of an IMF eco-system is intended to work.

2 Background Knowledge

This chapter will present some concepts that are necessary background knowledge for understanding IMF. A brief overview of the system thinking approach where complex systems are broken down to smaller sub-systems is given to provide an intuition of the structures that are used in IMF. Furthermore, a short introduction to Information Model and the benefit of using Information Model is given. The last part introduces Aspects as a way of organizing information into different categories.

2.1 System Thinking

IMF is used to describe Facility Assets, where the Facility Assets typically consist of multiple systems interacting with each other. This section will define *system* and how the structure of a system is reflected in the IMF Models.

Definition: **System –** *A system is a combination of interacting elements organized to achieve one or more stated purpose.* [3]

Definition: **System Element –** *A System Element is a member of a set of elements that constitute a system.* [3]

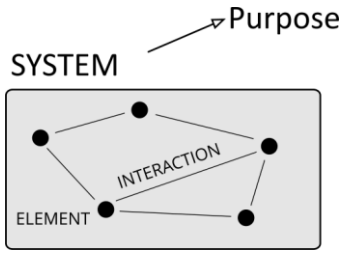


Figure 3: Illustration of the system concept.

The system concept is illustrated in Figure 3**Error! Reference source not found.**. The complexity of the different system elements that constitute a system might vary and the more complex system elements may be considered as a system with its own system elements. This view of a system can be applied recursively, where the system elements are drilled down into systems until a desired level of detail is achieved. The process of drilling down a system element of one system to a system with its own system elements is called a *system breakdown.* The repeated pattern of system breakdowns is illustrated in Figure 4.

Example: **Cooling System**

A cooling system for an oil and gas facility might consist of a circulation system for the cooling medium and a heat exchanger with seawater supply. The circulation system can be drilled down into a set of circulation pumps, distribution headers and cooling medium expansion tank. All these system elements in the circulation system can be further drilled down in new system breakdowns (e.g., cooling medium expansion tank system consists of valves and instruments) until the desired level of details is achieved. The desired level of details might vary between the different parties involved in the engineering of a Facility Asset (e.g., the pump is considered a system element by the engineering contractor while the pump supplier needs to perform a system breakdown on the pump).

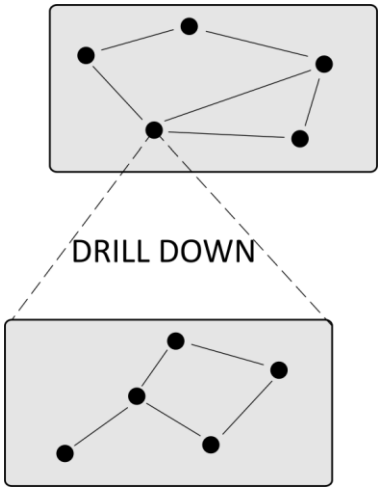


Figure 4: Illustration of a system breakdown

The system elements interact with each other by sending and receiving medium (e.g., material flow, electric energy, digital information, etc.). A system might interact with other systems, where the system elements of one system sends/receives medium to/from a system element in a different system as illustrated in Figure 5**Error! Reference source not found.**. The interface between the systems defines how the systems interact i.e., the medium and medium states that are allowed to be sent/received.

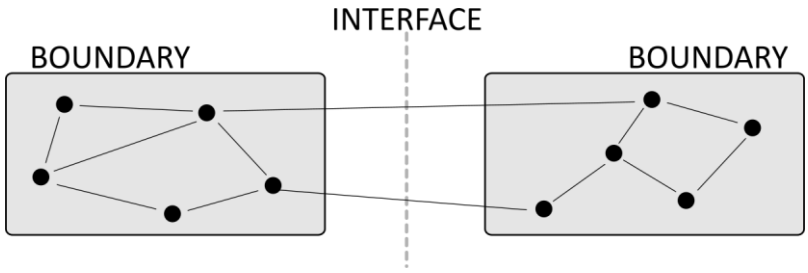


Figure 5: Illustration of the interaction of system elements inside and outside of system boundaries.

Example: **Cooling System cont.**

In the cooling system example, the pump and the heat exchanger interact with each other since the pump sends (pumps) cooling medium to the heat exchanger. The circulation pump is connected to an electric supply; thus, the cooling system interacts with the electrical supply system of the facility.

A repeated use of the principles of system breakdown and interactions gives rise to similar patterns recuring at progressively smaller scales both vertically (system breakdown) and horizontally (system interactions). The pattern is illustrated Figure 6.

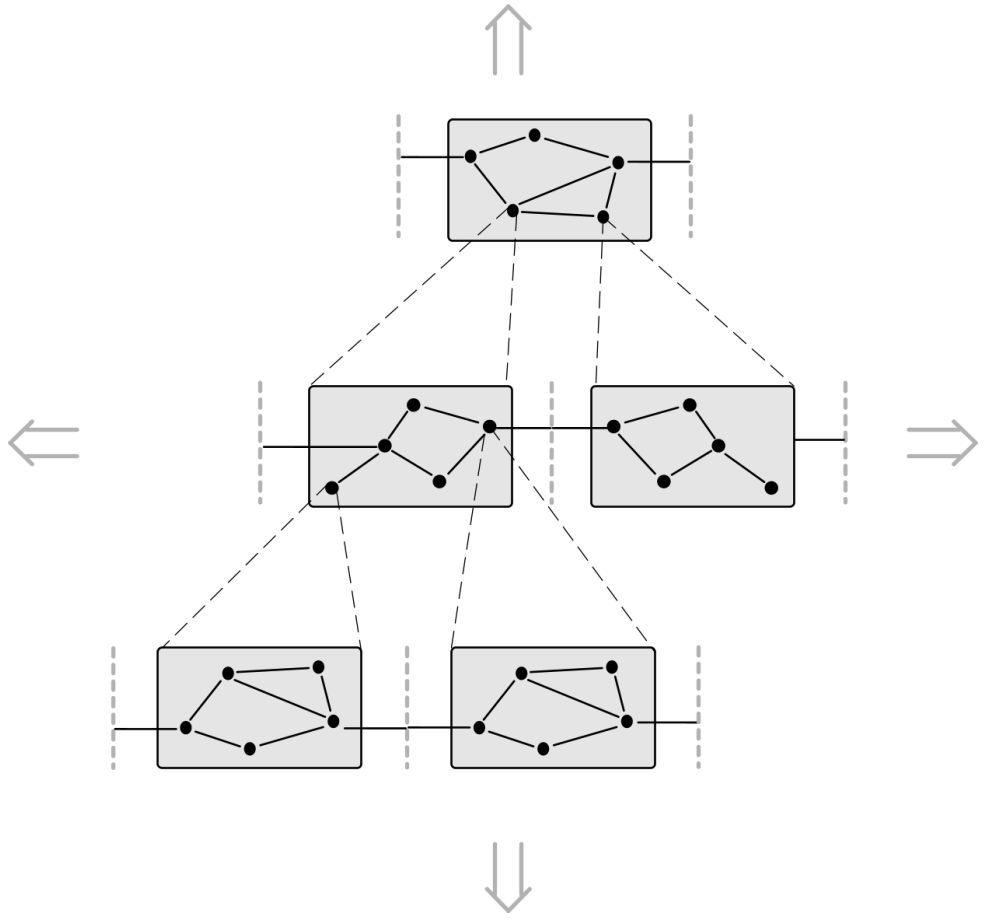


Figure 6: Illustration of system breakdown and topology.

IMF Models are created by performing system breakdowns of the intended Facility Asset and modelling the interactions between the resulting system elements. This is performed in tree structures, where the relations between the system elements are described. The interactions between the system elements are modelled by connecting medium streams between terminals on the system elements.

2.2 Information Model

Definition: **Information Model -** *An Information Model is a structure of information objects and relations between them that is designed to capture knowledge.*

Information Models can be used to model Facility Assets, such as components, systems, and process plants, and can be used for both real-world and abstract assets. In an Information Model properties, relationships, constraints, and rules are used to describe and capture knowledge about an asset. A main objective of an Information Model is to structure and provide the necessary context to interpret the information objects within the scope of the model, e.g., give a context for how a number that quantifies pressure is related to the output pressure from a pump and how this is potentially relevant for a heat exchanger.

An essential characteristic of Information Models is to formalize the description of the model without setting constraints on the implementation of the model. The implementation of an Information Model in a particular software is called a data model. Having an application neutral description of the

information is beneficial when we want to exchange the information between different systems since it limits the amount of mapping required between the systems.

IMF is a framework for creating Information Models, where the allowed elements, relations and rules for the Information Model are described. The formalization of the IMF Language is covered in Chapter 4. By conforming to IMF, the created Information Models can be exchanged with other users of IMF, and the Information Models can be integrated with each other to form larger models.

The intended use for IMF is to make Facility Assets Information Models, where the requirements and specifications for building a Facility Asset are modelled. The effort to model Facility Assets requires input from many different parties across the value chain. IMF is designed to allow all levels of detail. The necessary level of details varies between the different parties involved in the design of the Facility Asset e.g., the details for how a particular pump is made is not needed by an engineering contractor, while for the pump supplier these details are essential. By following IMF, the detailed Information Model from the pump supplier can be exchanged with the engineering contractor and integrated into the engineering contractor's overall Information Model of the Facility Asset.

2.3 Aspects

The concept of aspects is that when viewing something from different perspectives the resulting information will be different. The ISO/IEC 81346-1 [2] 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. For the purpose of modelling a Facility Asset, understood as describing the *intended* one, the aspects *Function*, *Product* and *Location* are needed as a minimum (see Section 4.2.2). To extend the Information Model with information about the *actual* Facility Asset, the *Installed* aspect is also included. Each aspect has a color assigned. The colors are included in the list below and are used in later figures to indicate the belonging aspect. Figure 7 illustrates these aspects as being different perspectives on the information about an intended pumping activity.

- **The Function aspect (Yellow)** ❶ is about the intended activity, in this example: to pump, providing information about required activity, performance, and function.
- **The Product aspect (Cyan)** 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 (Magenta)** ❸ is about the spatial envelope – the size and shape – of the specified pump and the requirements imposed by the location (ambient temperature etc.).
- **The Installed aspect (Dark blue)** ❹ is about information about the actual pump, with information such as serial number, run hours, and status.

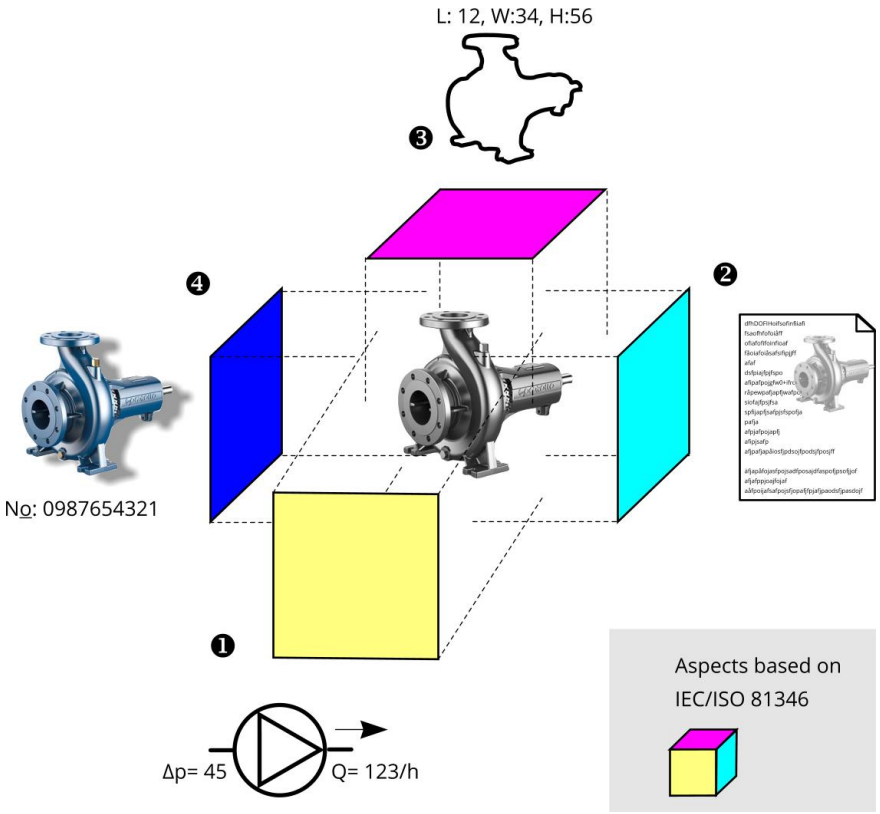


Figure 7: The concept of aspects. Here illustrated using the Function (Yellow), Product (Cyan), Location (Magenta), and Installed (Dark blue) aspects.

It is valuable to structure the information about something into aspects, but more importantly, this enables relating different aspects of something into different Breakdowns (see Figure 8) that each represent different perspectives of the Facility Asset as a whole. Figure 8 illustrates how an object relates to different Aspect Breakdowns here exemplified with a Function, Product and Location Aspect Breakdown.

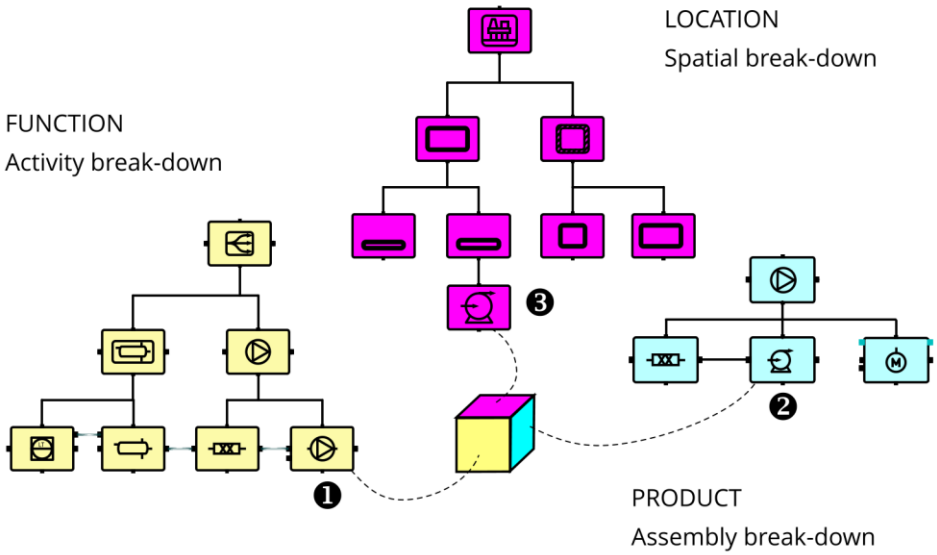


Figure 8: Aspect Breakdowns for the Function, Location and Product aspect.

Relations between aspects may arise when something (the cube in Figure 8) has several aspects that each relate to a different aspect of the Information 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 breakdown 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 product and the breakdown reflects how the product is *part of* a larger assembly of products. The Location aspect ❸ is information about the space occupied by the product. The space occupied by the product is a part of a spatial breakdown, where its relative location and the requirements specific to the space is specified.

3 A New Way of Working using Information Models

Section 3.1 gives a brief introduction to the IMF concept and new ways of working. Section 3.2 discusses new methods, roles, and competencies that IMF introduces.

3.1 Using Information Models

To solve the problems discussed in Section 1.1, fundamental changes in the way we capture and represent information are needed. Used properly, *Information Models* in combination with Reference Data Libraries (RDLs) include the features necessary to handle the problems of the current document practice.

Figure 9 illustrates a new way of working using Information Models through the *logical flow* of value creation during an industrial investment and development project. The actual execution schedule will have many overlaps and iterations that are intentionally left out.

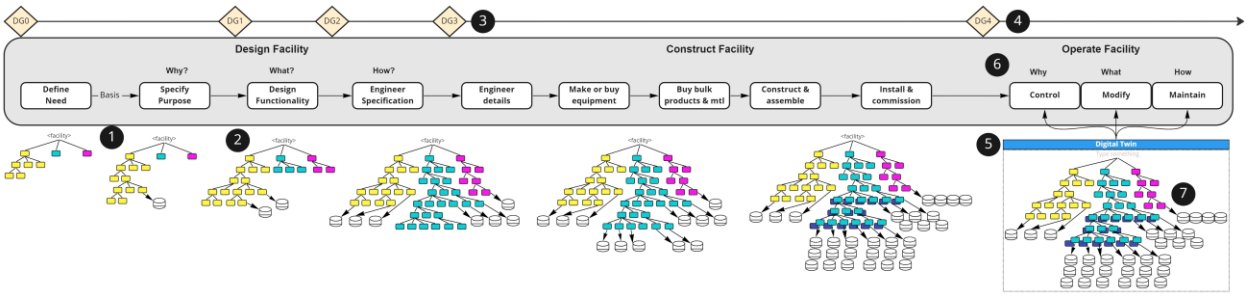


Figure 9: Using Information Models instead of documents during an industrial investment and development project.

The overall requirements to functionality of the Facility Asset (DG0) are captured in an Information Model ❶. As the work progresses into the design phase of the Facility Asset, several more detailed Information Models are created by discipline expertise to mature the design (DG1 and DG2) ❷. Due to the inherent features of the (machine-readable) format, the Information Models can be continuously checked for design flaws by an automated work process. Furthermore, the fragments of information, now represented through several Information Models, can be integrated along the way to ensure a consistent design and a valid description of the Facility Asset on a holistic level.

When the investment decision is made to execute the construction of the Facility Asset (DG3) ❸, Information Models of the parts of the Facility Asset are produced in large quantities by the supply chain.

When approaching operation (DG4) and handover to operator takes place ❹, the resulting Information Model, can be thought of as a *model-of-models*, containing the historical context and design decisions made all the way back to DG0 ❺. The holistic description and the contents of the Facility Asset is available at any level of detail, as required to operate, control, maintain, and later do modifications on the Facility Asset ❻. Furthermore, access to information is not restricted by documents and formatting but can be navigated freely ❼ and maintained efficiently.

3.2 New Methods, Roles, and Competencies

The new way of working implies new roles and requires new competencies. The following roles are relevant to discuss:

- The *Subject Matter Expert* (SME). SMEs are experts on discipline engineering. For example, a process engineer who develops the design of the main processing system, or an electrical engineer who develops the design of the electrical supply and distribution.
- The *System Engineer* is an expert on system integration. For example, a System Engineer is 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 *Data Engineer* is an expert on the flow of data between IT systems and applications. For example, a Data Engineer manages the export of data from a model authoring application into an engineering register system.
- The *Knowledge Engineer* is an expert on semantic technology. For example, a Knowledge Engineer manages the validation and integrity verification of the model of a design.

Engineering and design is a process of making a series of decisions founded on subject matter expertise and governed by requirements and limitations. This is the domain of SMEs and System Engineers, but the new way of working allows a significant shift in focus away from *formatting* of information, and towards *creating* information. It is less constrained by document formats and instead allows a much more flexible and incremental way of creating a design. This will have an impact on how the work is optimally allocated into different roles.

The SME role is by definition holding expertise on a defined subject, usually a single discipline, but the new way of working will reward an approach of also working across disciplines that are interlinked and interdependent, thus reducing time-consuming coordination effort. As systems thinking is a fundamental part, the System Engineer role may have a stronger impact, in particular when integrating segments of models into a whole, and when managing how systems interact.

While conventionally the Data Engineer has had more focus on batch transfer of data between systems, this role will need to strengthen the focus on publishing and sharing of model data, and on leveraging semantic technologies. A significant value of modelling is that it allows use of advanced semantic technologies and mechanisms for machine-based validation and verification. The role of the Knowledge Engineer is instrumental for achieving this.

Figure 10 illustrates how the different roles work together to enable modelling of parts of the Facility Asset that then are integrated into a complete model where semantic technology is utilized to verify and validate the integrity of the model.

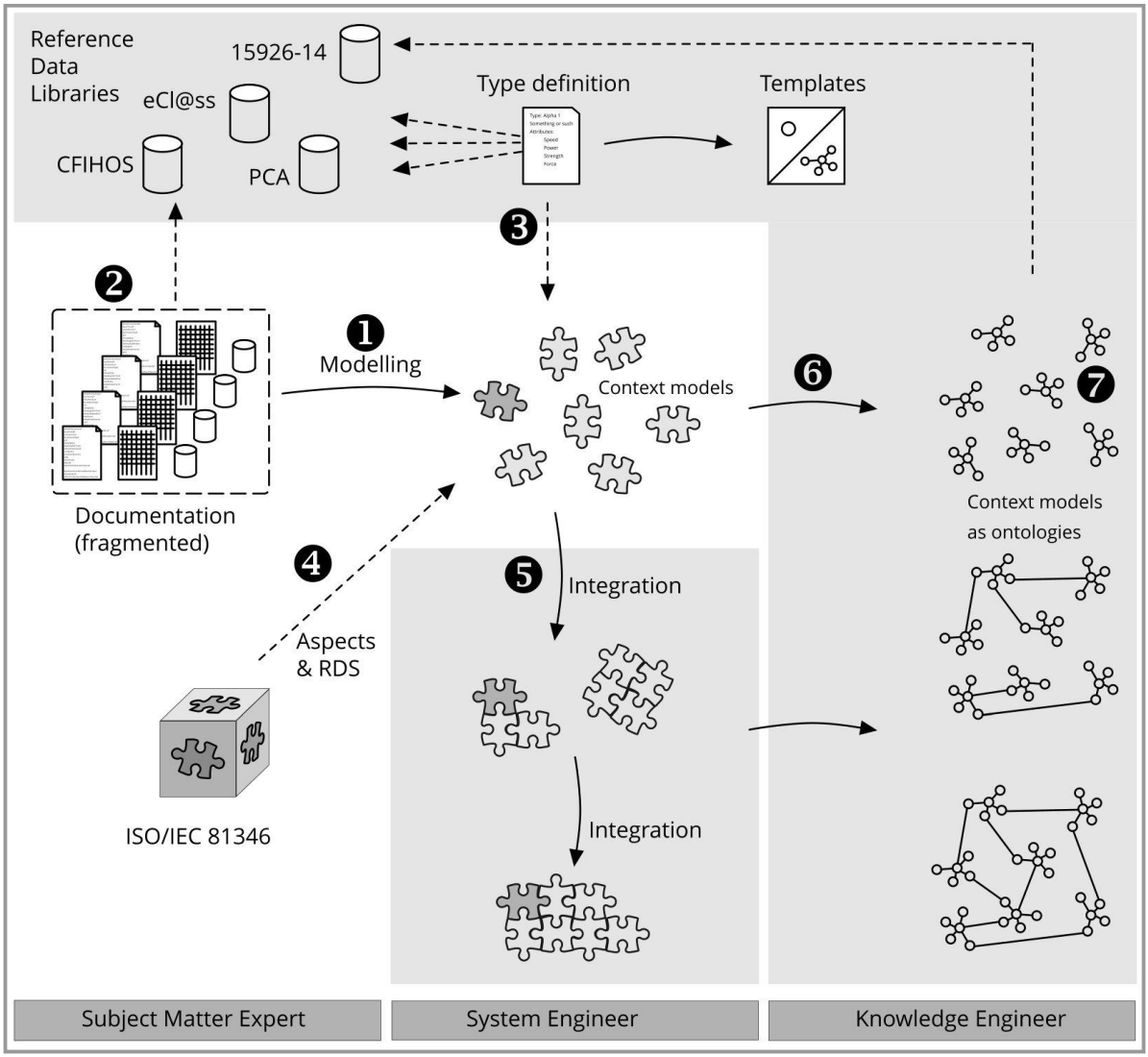


Figure 10: Different expertise and how they enable Information Modelling of Facility Assets.

The SME is modelling information that today is available only in fragments, as documents ❶.
The documents may refer to standards, possibly exploiting reference data, such as names of shared properties and classes, and initiatives to digitally enrich documentation format such as DEXPI[2] ❷.
When modelling, the SME creates the building blocks of the model from definitions held in a common industry library, a Reference Data Library ❸, enabling the re-use of well-proven design patterns (e.g., type of pump configuration). The building blocks are put together into a model in accordance with the IMF rules and structures that build on the IEC/ISO81346 O&G standard [4] ❹.

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 IMF Models, here shown as puzzle pieces, that the System Engineer later can bring together as a

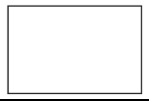
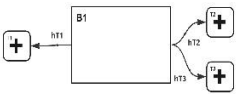
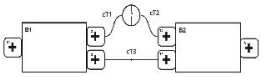
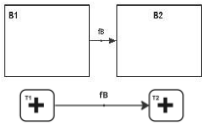
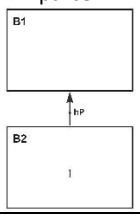
2 https://dexpi.org/specifications/

complete puzzle ❺. IMF Models can be translated ❻ to enable the Knowledge Engineers to exploit verification techniques such as automated reasoning ❼.

4 The Information Modelling Framework Language

This chapter defines the vocabulary and grammar rules of the IMF language in Section 4.2 and 4.3. Further, it introduces the Aspect Object in Section 4.4 that is central in the creation of IMF models, and shows the capabilities of the language through the creation of Breakdowns and Topologies in Section 4.5 and 4.6. Table 3 gives an overview of the different terms that is convenient to have when reading the chapter. Appendix C - Structural Specification gives details for how to implement the IMF language. The vocabulary and grammar of the IMF language is implemented using semantic web technologies, see Appendix D – IMF Semantic Web Resources. A mapping of the IMF vocabulary into first-order logic suitable for developing a translation into ontologies such as ISO 15926-14 is found in Appendix E.

Table 3: An overview of terms and symbols in the IMF language.

Element	Block 	Terminal 	Interface Point 		
Relation	hasTerminal 	connectedTo 	fulfilledBy 	partOf 	
Aspect Elements	**Aspect Block**	Function Block 	Product Block 	Location Block 	Installed Block 
	Aspect Terminal	Function Terminal 	Product Terminal 	Location Terminal Not used	Installed Terminal 
	Aspect Interface Point	Function Interface Point 	Product Interface Point 	Location Interface Point Not used	Installed Interface Point 
Aspect Object		Function Object 	Product Object 	Location Object 	Installed Object 

4.1 Language and Objectives

A language is a structured system of communication. We can say that a language is divided in two parts:

- The **Vocabulary**: The set of free elements in the language.
- The **Grammar rules**: The set of structural constraints of the language.

In general, we can say that natural languages[3] have the purpose of being an effective way of communicating, both written and oral for humans. When we construct the IMF language, we need to define a set of elements that constitutes the vocabulary and grammar rules conforming to the relevant objectives given in Section 1.2:

- *The language shall be made so that the SMEs themselves are users of the framework.* This means that the IMF language must:
    - Contain a limited set of elements (vocabulary) and grammar rules such that it is easy to learn and simple to use.
- *The language shall provide incremental value*. This means that the IMF language must be:
    - Designed such that one can have an incremental approach and model fragment by fragment.
- *The language shall be scalable across disciplines, work processes, and the value chain*. This means that the IMF language must:
    - Enable SMEs from a range of disciplines to fully express their design using the same modelling principles.
    - Enable the SMEs to express different levels of precision in their design.
    - Enable use in different phases of a project (concept, detail engineering, manufacturing, etc.).
- *The language shall have the precision of machine interpretation to allow automated verification*. This means that the IMF language must:
    - Be precise and unambiguous.
    - Allow translation into an ontology language that can use automated reasoning techniques in combination with information in the RDL for verification.

4.2 The IMF Languages Vocabulary

This section formally defines the different elements in the IMF language.

4.2.1 Elements

An SME needs to be able to describe and talk about a Facility Asset, whatever the scope of that is, its boundaries, and conditions for connecting them. To meet these needs, the IMF language defines three *Elements* into the vocabulary: *Block*, *Terminal,* and *Interface Point*.

Definition: **Block (B) -** *A Block **B** is an object with a collection of attributes used to describe an entity*[4]*.*

A Block is visualized with a rectangular box. See Figure 11. A Block is something that is of interest to the SME. We use the Block to set boundaries of something complex which is convenient to treat as one entity. This could be anything from a whole industry plant, a pump system, or a small location that is of interest.

[3] A natural language is languages that have evolved naturally through time by humans.

[4] An entity is a thing with distinct and independent existence.

Figure 11: A Block.

Definition: **Terminal (T) -** *A Terminal **T** is an object with a collection of attributes used to describe the input or output of a Block **B**, and hence its boundaries.*

A Terminal is visualized as a small box with a plus (+) sign, as shown in Figure 12. Input Terminals are placed on the left side of a Block while Output Terminals are on the right, facilitating a left-to-right flow. A Terminal cannot exist without a Block. This makes sense because a Terminal is at the boundaries of a Block, and if there is no Block there is no boundary.

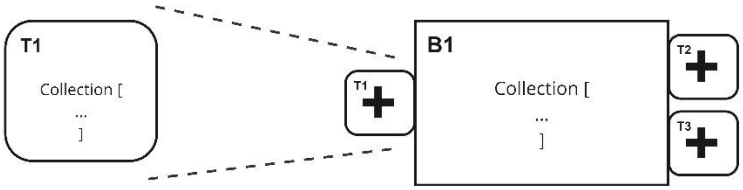


Figure 12: A Terminal.

Definition: **Interface Point (IP) -** *An Interface Point **IP** is an object with a collection of attributes used to specify the condition for connecting one Block **B1** and its Terminal **T1**, to another Block **B2** and its Terminal **T2**.*

An Interface Point is visualized using two half-moons together forming a circle as shown in Figure 13. An Interface Point shall have at least one Block and its Terminal that together constitutes one side of the condition for a connection. To 'complete' an Interface Point, two Blocks and their Terminals are needed. Hence the two half-moons indicating that two conditions (collection of attributes) need to match.

It is useful to think about the Interface Point as a set of equations that must be satisfied.

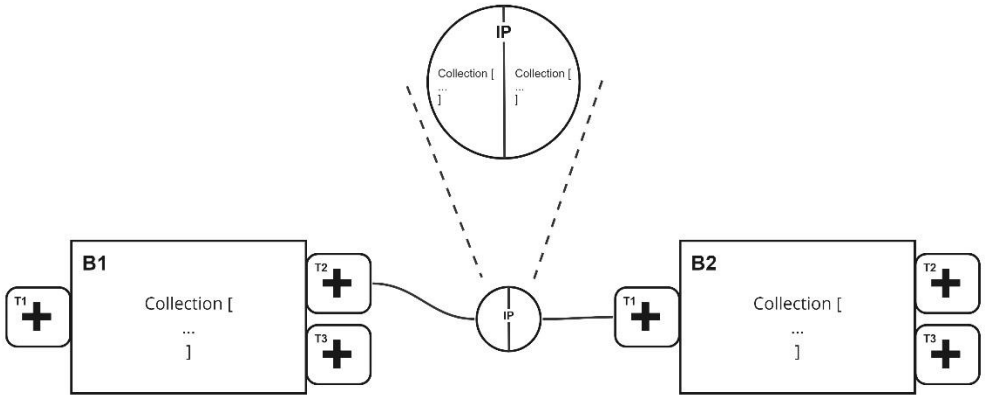


Figure 13: An Interface Point.

4.2.2 Aspects

The concept of aspects is taken from ISO/IEC 81346-1 [2]. Aspects allow us to view a Facility Asset from different perspectives and give a structuring mechanism for separating different types of information about a Facility Asset, and hence allows scalability across disciplines, work processes, and the value chain.

Definition: **Aspect** - *An aspect is a particular way of viewing a Facility Asset. Different aspects have no overlap in information.*

Aspects act like filters on a Facility Asset and highlight the information that is of relevance [2]. The IMF language defines four elementary aspects defined in Table 4. Aspect Elements that represent different views on the same thing can be related across aspects. To ease understanding it is convenient to think about a Facility Asset as a set of *Artifacts*, meaning physical objects or software, with given special extensions, that have the capability to bring about certain *activities*, either individually or in interaction with each other.

Table 4: Definition of different aspects and their associated prefix and colors.

Aspect	Definition	Prefix	Color
The Function Aspect (F)	*The Function Aspect is about the requirements to the intended activity.*	=	Yellow
The Product Aspect (P)	*The Product Aspect is about the specification of the artifact.*	-	Cyan
The Location Aspect (L)	*The Location Aspect is about the specification of the spatial extension of the artifact.*	+	Magenta
The Installed Aspect (I)	*The Installed Aspect is about the documentation of the installed artifact.*	::	Dark Blue

Note: More aspects can be defined to support other ways of viewing a Facility Asset.

4.2.3 Combing Elements and Aspects to Create Aspect Elements

By combing the different Elements and Aspects, from Section 4.2.1 and 4.2.2 respectively, we introduce the *Aspect Elements*:

Definition: **Aspect Element** - *An Aspect Element is an Element (Block, Terminal, or Interface Point) with one, and only one, Aspect associated.*

Thus, from the Definition we have three groups of Aspect Elements:

- Aspect Blocks,
- Aspect Terminals,
- Aspect Interface Points.

An overview of the different Aspect Elements is given in Table 5. Note that Location Terminal and Location Interface Point is not used in this version.

For convenience, when we refer to a Block, Terminal, or Interface Point of a certain aspect we swap the word 'Aspect' for the given aspect the Block belongs to. For example, we say 'Function Block' instead of the general designation 'Aspect Block'.

Table 5: Different Aspect Elements.

Elements / Aspects	**Block (B)**	**Terminal (T)**	**Interface Point (IP)**
Function (F)	Function Block (**FB**)	Function Terminal (**FT**)	Function Interface Point (**FIP**)
Product (P)	Product Block (**PB**)	Product Terminal (**PT**)	Product Interface Point (**PIP**)
Location (L)	Location Block (**LB**)	Not used	Not used
Installed (I)	Installed Block (**IB**)	Installed Terminal (**IT**)	Installed Interface Point (**IIP**)

Every Aspect Element have an own intuition summarized in Table 6, Table 7 and Table 8.

Table 6: Different Aspect Blocks and their intuition.

Name	**Intuition**	**Graphics**
Function Block (**FB**)	A function block holds the requirements to intended activity.	
Product Block (**FB**)	A product block holds the specification of an artifact.	
Location Block (**LB**)	A location block holds the specification to the spatial extension of an artifact and requirements imposed by its spatial position in the Location Breakdown.	
Installed Block (**IB**)	An installed block holds the documentation of an installed artifact.	

Table 7: Different Aspect Terminals and their intuition.

Name	**Intuition**	**Graphics**
Function Terminal (**FT**)	Requirements to one input/output stream state of a purpose activity	

Product Terminal (**PT**)	Specification of one input/output/bidirectional terminal of an artifact	
Location Terminal (**LT**)	Not used	Not used
Installed Terminal (**IT**)	Documentation of an installed artifact terminal.	

Table 8: Different Aspect Interface Points and their intuition.

Name	Intuition	Graphics
Function Interface Point (**FIP**)	A Function Interface Point (**FIP**) holds the requirements to the activity so that two terminals can be connected.	
Product Interface Point (**PIP**)	A Product Interface Point (**PIP**) holds the requirements to the artifact so that two terminals can be connected.	
Location Interface Point (**LIP**)	Not used	Not used
Installed Interface Point (**IIP**)	An Installed Interface Point (**IIP**) documents the Interface between the two terminals of the as-built artifacts.	

4.2.4 Relations

To ensure that SMEs can fully express themselves and their design, they must be able to relate different Blocks and their Terminals in various ways.

In contrast to an Element, a relation does not hold a collection of attributes. A relation is visualized with a drawn line between different Elements. The IMF language defines four different relations: *hasTerminal, connectedTo*, *partOf*, and *fulfilledBy*.

Definition: **hasTerminal relation (hT)–** *A hasTerminal relation **hT** is a specification of a connection between a Block **B** and a Terminal **T**.*

We say that a Block **B** *hasTerminal* **T**, and hence the name of the relation and the direction indicated by the arrow in Figure 14.

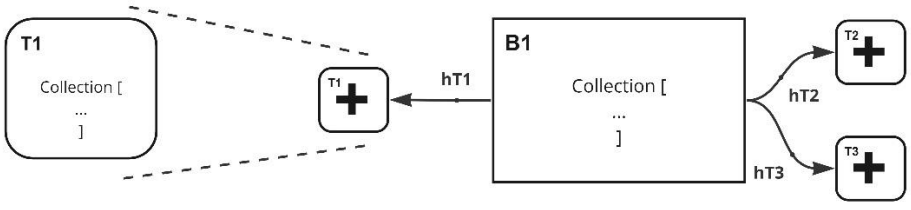


Figure 14: Several hasTerminal relations.

Definition: **The *connectedTo* relation -** *The connectedTo relation is a specification of a connection between two Terminals **T1** and **T2**, and optionally, an Interface Point **IP**.*

We say that an output terminal **T1** is *connectedTo* an input terminal **T2**. If the condition of the output and input Terminals being connected are identical, there is no need for an Interface Point. If not, the conditions for connecting the Terminals must be specified through an Interface Point. Figure 15 illustrates the options.

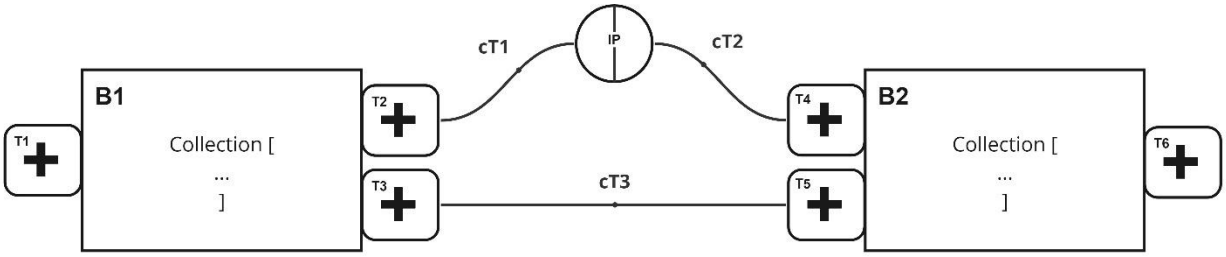


Figure 15: Several connectedTo relations.

Definition: **The *partOf* relation -** *The partOf relation **pO** is a specification of a connection between two Blocks **B1** and **B2** in the sense that **B2** is a part of **B1**.*

We say that a Block **B2** is *partOf* **B1**, and hence the name of the relation and the direction indicated by the arrow in Figure 16. The partOf relation represents what is known as a parent-child relationship. We say that a Block **B2** is child of a parent Block **B1**. The inverse relation of part of is *hasPart*. That means that if **B2** is partOf **B1**, it is also true that **B1** hasPart **B2**

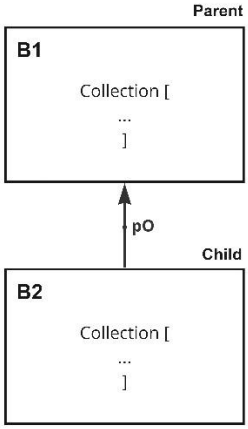


Figure 16: A partOf relation.

Definition: **The *fulfilledBy* relation -** *The fulfilledBy **fB** relation is a specification of a connection between two Blocks **B1** and **B2** in the sense that **B2** fulfills a requirement(s) from **B1**.*

We say that a Block **B1** is *fulfilledBy **B2***, and hence the name of the relation and the direction indicated by the arrow in Figure 17.

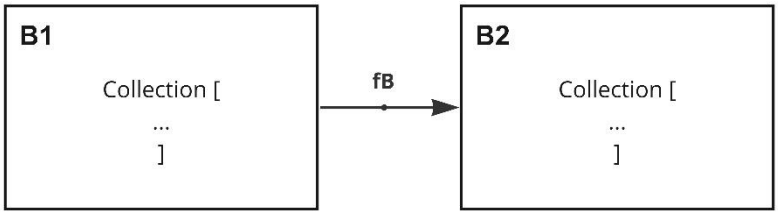


Figure 17: A fulfilledBy relation.

4.3 The IMF Language Grammar Rules

This Section gives an overview of grammar rules in the IMF language.

4.3.1 Use of Relations

The IMF Language defines a set of allowable relations between different Aspect Elements. These are rules that set structural constraints on the IMF language. It is convenient to divide the relations in two categories: *Intra-Aspect, and Inter-Aspect relations*.

Definition: **Intra-Aspect Relations** – *Intra-Aspect relations are relations between two Aspect Elements of the same aspect.*

Definition: **Inter-Aspect Relations** – *Inter-Aspect relations are relations between two Aspect Elements of different aspects.*

Table 9: Different relations allowed.

Elements/ Aspects		Block to Block	Block to Terminal	Terminal to Terminal	Terminal to Interface Point
Intra-Aspect Relations	**F→ F**	`partOf`	`hasTerminal`	`connectedTo`	`connectedTo`
	P→P	`partOf`	`hasTerminal`	`connectedTo`	`connectedTo`
	L→L	`partOf`			
	I→I	`partOf`	`hasTerminal`	`connectedTo`	`connectedTo`
Inter-Aspect Relations	**F→P**	`fulfilledBy`[5]		`fulfilledBy`	
	L→P	`fulfilledBy`[6]			
	P→I	`fulfilledBy`[7]		`fulfilledBy`	

TableTable 9 gives an overview of the different relations allowed. Table 9Table 10 to Table 13 give an overview of the different relations between Aspect Elements and their intuitions.

Table 10: Different partOf relations and their intuition.

partOf relation	Intuition
`partOf(`**`FB1,FB2`**`)`	The purpose activity of Function Block 1 (**FB1**) is a sub-activity of that of Function Block 1 (**FB2**)

[5] This relation is also referred to as `hasFunction` in the sense that the function has a product that performs the function.

[6] This relation is also referred to `hasLocation` in the sense that the product is located in the location.

[7] This relation is also referred to as `realizedBy` in the sense that the installed artifact is a realization of the specified product.

`partOf(`**`PB1,PB2`**`)`	The artifact of Product Block 1 (**PB1)** is a sub-assembly of the artifact of Product Block 2 (**PB2**)
`partOf(`**`LB1,LB2`**`)`	The location of Location Block 1 (**LB1**) is located in the location of Location Block 2 (**LB2**)
`partOf(`**`IB1,IB2`**`)`	The artifact of Installed Block 1 (**IB1**) is sub-assembly of the artifact of Installed Block 2 (**IB2**)

Table 11: Different hasTerminal relations and their intuition.

hasTerminal relation	**Specification of**	**Intuition**
`hasTerminal(`**`FB,FT`**`)`	Activity	The media state of Function Terminal (**FT**) is the input/output of the purpose activity in Function Block (**FB**)
`hasTerminal(`**`PB,PT`**`)`	Artifact	The Product Terminal (**PT**) is a specification of one input/output terminal of the artifact of Product Block (**PB**)
`hasTerminal(`**`LB,LT`**`)`	Location	Not used
`hasTerminal(`**`IB,IT`**`)`	Artifact	The installed terminal (**IT**) holds documentation of an installed artifact terminal (**IB**)

Table 12: Different connectedTo relations and their intuition.

connectedTo Relation	**Specification of**	**Intuition**
`connectedTo(`**`FT1,FT2,FIP`**`)`	Activity	The media state of Function Terminal 1 (**FT1**) is equal to that of Function Terminal 2 (**FT2**) and satisfies the requirement in Function Interface Point (**FIP**).
`connectedTo(`**`PT1,PT2,PIP`**`)`	Artifact	The stream of Product Terminal 1 (**PT1**) is physically connected to the steam of Product Terminal 2 (**PT2**) through Product Interface Point (**PIP**).
`connectedTo(`**`LT1,LT2,LIP`**`)`	Location	Not used
`connectedTo(`**`IT1,IT2,IIP`**`)`	Artifact	The stream of Installed Terminal 1 (**IT1**) is physically connected to the stream of Installed Terminal 2 (**IT2**) documented by Installed Interface Point (**IIP**).

Table 13: Different fulfilledBy relations and their intuition.

fulfilledBy relation	**Intuition**
`fulfilledBy(`**`FB,PB`**`)`	The artifact of Product Block **(PB)** fulfills the purpose of Function Block **(FB)**.
`fulfilledBy(`**`FT,PT`**`)`	Artifact Terminal of Product Terminal (**PT**) provides containment of the media state of Function Terminal **(FT)**.
`fulfilledBy(`**`LB,PB`**`)`	The artifact of Product Block (**PB**) fulfills the location requirements of Location Block (**LB**).
`fulfilledBy(`**`LT,PT`**`)`	The artifact Terminal of Product Terminal (**PT**) fulfills the location requirements of Location Terminal (**LT**).
`fulfilledBy(`**`PB,IB`**`)`	The installed artifact of Installed Block (**IB**) fulfills the artifact specifications of Product Block (**PB**).
`fulfilledBy(`**`PT,IT`**`)`	The installed artifact terminal Installed Terminal (**IT**) fulfills the terminal specifications of Product Terminal (**PT)**.

4.3.2 List of Grammar Rules

The IMF Language has the following grammar rules:

1. An Element is either a Block, a Terminal or an Interface Point.
2. An Aspect Element is an Element with exactly one Aspect.
3. In this version of IMF, there are four Aspects: Function, Product, Location and Installed.
4. partOf, hasTerminal and connectedTo are intra-aspect relations and may only relate Aspect Elements of the same Aspect.
5. fulfilledBy is an inter-aspect relation and may only relate Aspect Elements of different Aspects.
6. partOf is a relation between Blocks. The inverse relation of partOf is hasPart. A Block may have many children and at most one parent.
7. hasTerminal is a relation between Blocks and Terminals. A Block may have many Terminals. A Terminal must belong to exactly one Block.
8. connectedTo is a relation between Output Terminals and Input Terminals and Interface Points. A connectionTo relationship relates one Output Terminal with one Input Terminal and optionally one Interface Point. A Terminal is connected to at most one (other) Terminal. An Interface Point is connected to at most two Terminals.
9. fulfilledBy is a relation between Blocks or between Terminals. The inverse relation of fulfilledBy is fulfills. The following fulfilledBy relations are possible:
    a. A Function element is fulfilledBy at most one Product element. A Product element may fulfill many Function elements.
    b. A Location element is fulfilledBy at most one Product element. A Product element may fulfill many Location elements under the condition that the Location elements reside in different partOf hierarchies.
    c. A Product element can be fulfilledBy by many Installed elements. An Installed element can fulfill at most one Product element.

4.4 The Aspect Object

Until now, we have defined the IMF language vocabulary. By using the Aspect Block, Aspect Terminal, and the hasTerminal relation, we can establish one of the core building-blocks for creating IMF models: *The Aspect Object*.

Definition: **The Aspect Object –** *An Aspect Object is a composite object structure comprised of an Aspect Block **B** and its Aspect Terminals **T** [0..n] that are connected through their hasTerminal relations **hTs**. An Aspect Object has always one, and only one aspect.*

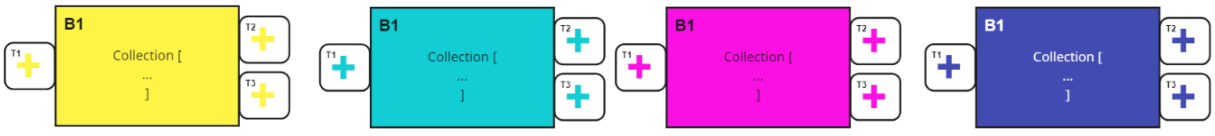


Figure 18: Four different Aspect Objects: Function, Product, Location, and Installed.

The definition entails that only Aspect Blocks and Terminals of the same Aspect can be combined to create an Aspect Object. The Aspect Object makes the SME able to talk about the Facility Asset itself (the Block) and its boundaries (the Terminals) in a precise and effective way. Given the aspects defined in Section 4.2.2, we have four different Aspect Objects, as shown in Figure 18. We use the colors (and

prefixes) given in Table 4 to illustrate which aspect each one resides in. The different Aspect Objects have a well-defined collection of attributes associated, given in Appendix A.

For convenience, when we refer to an Aspect Object of a certain aspect, we swap the word 'Aspect' for the given aspect the Aspect Object belongs to. For example, we say 'Function Object' instead of the general designation 'Aspect Block'.

In the following sections and chapters, we will primarily use the Aspect Object to create IMF models (Chapter 5) and IMF Types (Chapter 6).

4.5 The Rules for Creating Breakdowns using the partOf Relation

It is important for SMEs to express various levels of precision in their design. The partOf relation is a mechanism for this. Figure 19 shows three Aspect Blocks (**AB1**, **AB2**, and **AB3**) and two partOf relations (**pO1** and **pO2**). Together they form a *Breakdown*, also referred to as *Hierarchy*. We use a grey color on the aspect independent Aspect Objects.

The partOf relation promotes a *System-Of-Systems* way of thinking. Using the Breakdown to the left in Figure 19 as an example, we can think of **AB2** and **AB3** as two independent systems in context as part of a larger, more complex system **AB1**. The system-of-system approach facilitates a recursive way to create deeper Breakdowns to connect several levels of the design shown to the right in Figure 19.

Breakdowns can be interpreted in different ways. From a top-down view, it is reasonable to think about the Breakdown formed by the partOf relation as a mechanism for decomposing systems into sub-systems. Thus, in the Breakdown in Figure 19, **AB2** and **AB3** can be viewed as sub-components of **AB1**.

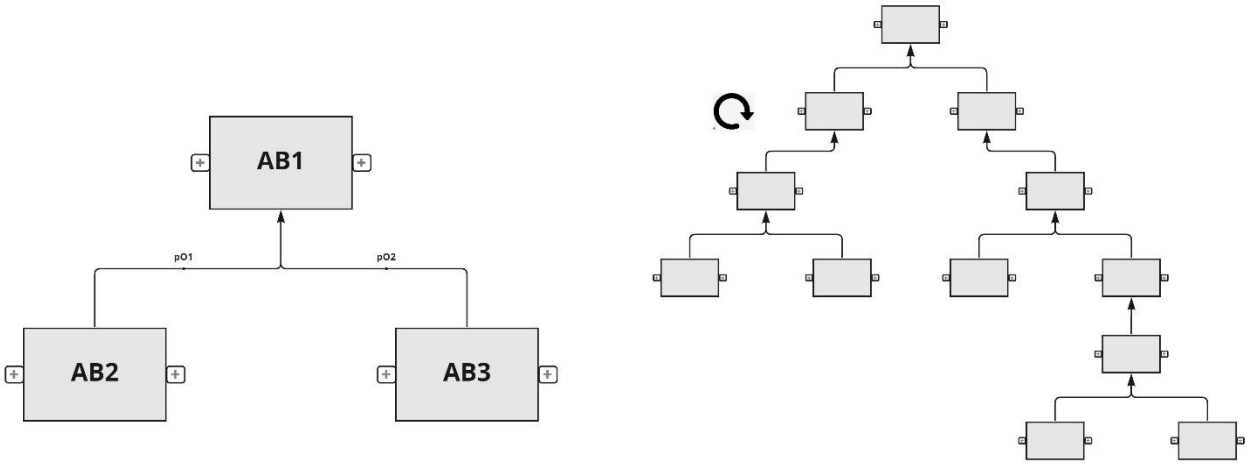


Figure 19: A small Breakdown with a parent AB1 and two children, AB2 anf AB3 to the left. A deeper Breakdown to the right.

From a bottom-up view, it is reasonable to think about the Breakdown as a grouping and abstraction mechanism. Again, using the Breakdown to the left in Figure 19 we can say that **AB2** and **AB3** are part of the grouping or abstraction formed by **AB1**. Furthermore, this is useful when it is convenient to treat a complex system (**AB1**) as a 'black box' by hiding details (**AB2** and **AB3**) and postponing dealing with them to a later stage. Top-down and Bottom-up approaches to modeling is further discussed in Chapter 5.4.1 and 5.4.2.

Breakdowns can be created in all aspects. The intuition about what they represent is given in Table 14.

Table 14: Aspect Breakdowns and their interpretation.

Aspect Breakdown	Intuition	Color
Function Aspect Breakdown	*The Function Aspect Breakdown specifies a breakdown of activities into sub-activities.*	Yellow
Product Aspect Breakdown	*The Product Aspect Breakdown specifies an assembly breakdown of artifacts.*	Cyan
Location Aspect Breakdown	*The Location Aspect Breakdown specifies a spatial breakdown.*	Magenta
Installed Aspect Breakdown	*The Installed Aspect Breakdown documents the as-built assembly breakdown of artifacts.*	Dark Blue

4.6 The Rules for Creating Topologies using the connectedTo Relation

The SMEs must be able to express how Facility Assets are connected to one another. Another way to think about it is the need to express what type of media (e.g., Material, Energy, Force, or Information) is streaming between the Facility Assets. Figure 20 shows three Aspects Blocks (**AB1**, **AB2**, and **AB3**), two Interface Points, and a set of connectedTo relations. Connected together they form a *Topology*. We use a grey color on the aspect independent Aspect Objects.

The connectedTo relation promotes a *System-to-System* way of thinking. Using the *Topology* in Figure 20 as an example we can think that some type of media is streaming from **AB1**, through **AB2**, and further on to **AB3**. The system-to-system approach facilitates a way to create large and complex interactions between Facility Assets through feedforward and feedback loops etc., as shown in Figure 21. Modelling topologies of different Facility Assets are further discussed in Chapter 5.4.3.

*Figure 20: A small topology between three Aspect Objects **AB1**, **AB2**, and **AB3**.*

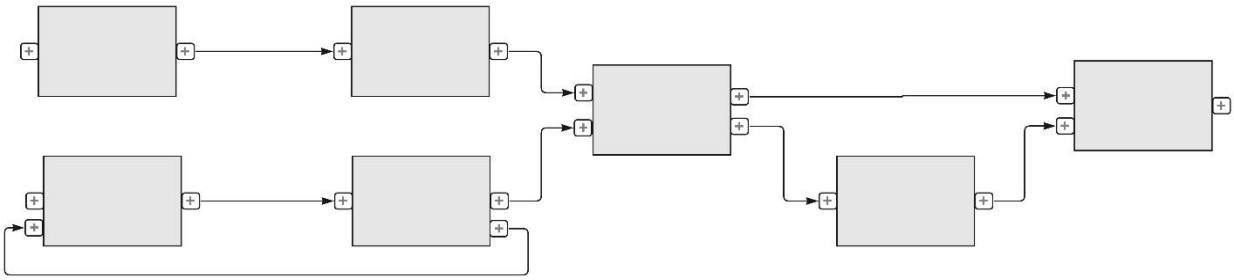


Figure 21: A Topology without Interface Points.

Topologies can be created in all aspects. The intuition about what they represent is given in Table 15. In this version, only Function, Product, and Installed is considered.

Table 15: Aspect Topologies and their interpretation.

Aspect Topology	Intuition	Color
Function Aspect Topology	*The Function Aspect Topology is about how Media is intended to stream between Function terminals.*	Yellow
Product Aspect Topology	*The Product Aspect Topology is specifying how artifacts shall be connected to contain the streams between Product terminals.*	Cyan
Location Aspect Topology	*Not used*	Magenta
Installed Aspect Topology	*The Installed Topology is documenting how the installed artifacts are connected to provide for streams between Installed terminals.*	Dark Blue

4.7 Combining Breakdowns and Topologies

Combining the Breakdowns and Topologies is a powerful modelling feature. A visual of this is given in Figure 22. To the right, the Breakdown is shown. To the left, this is shown in another way. Here, the partOf relation is implicitly given as the dark grey child Aspect Objects is seen as a containment and decomposition, as discussed in Section 4.5, of the light grey parent Aspect Object.

The left-hand side of Figure 22 is often referred to as a 'Block views' and shows the Topology between the child Aspect Objects. Notice how the Topology starts as inputs and ends as outputs from the parent Aspect Objects' Terminals. This is a powerful mechanism that makes us able to connect streams, as discussed in Section 4.6, across different abstraction levels in our Breakdown.

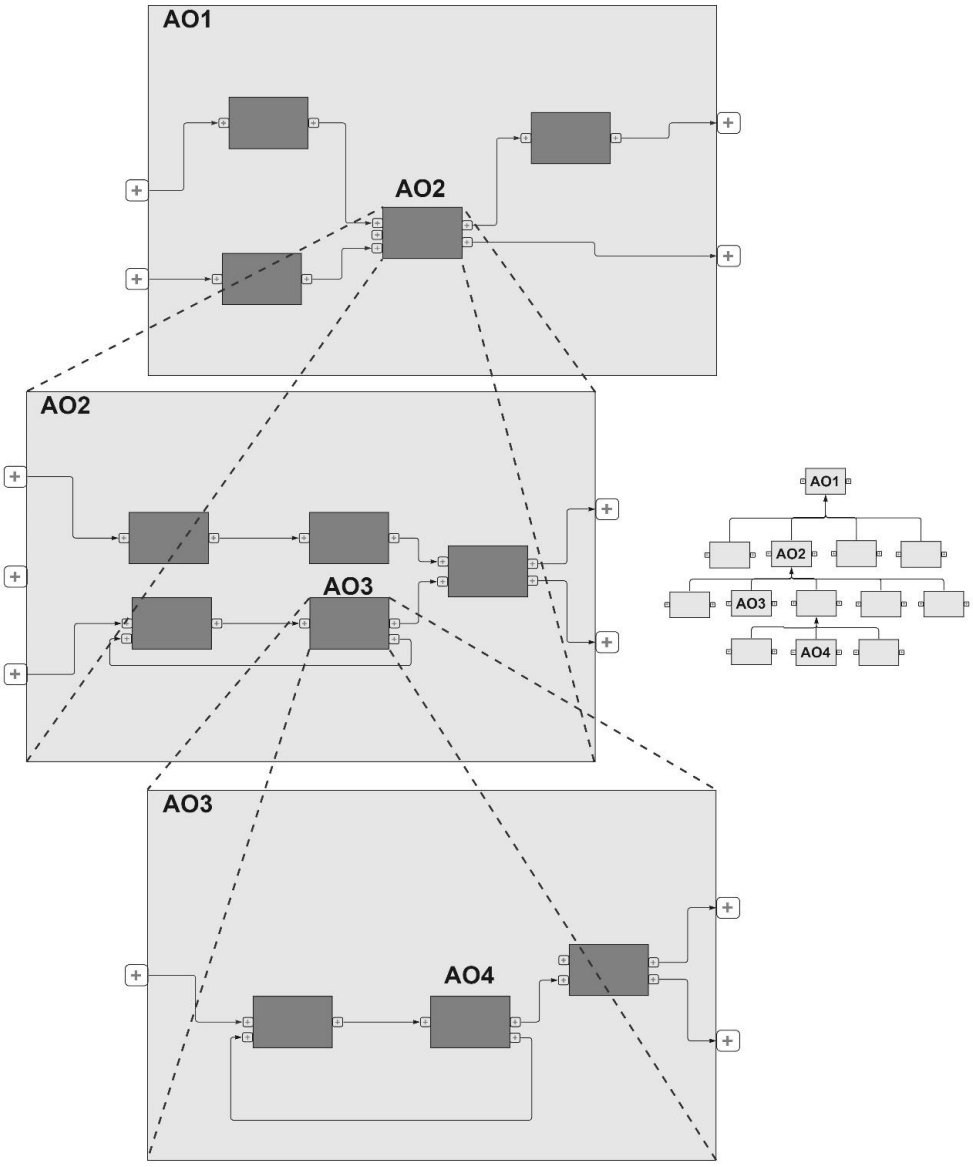


Figure 22: Combination of Breakdown and Topology (Interface Points omitted).

5 How to Create an IMF Model

This chapter gives an introduction and step by step guideline of how to create an IMF model. Prior understanding of the IMF as a framework and language is assumed. Acquaintance with IEC/ISO 81346 is beneficial but not required. When explaining modelling and advising on different approaches, some examples are provided to make the explanations more concrete. These examples cover only a few of the possible use cases, and the application of IMF modelling spans a much larger range.

5.1 What it Means to Create an IMF Model

Taking the example in Figure 23 of a simple fluid separation system, what it means to create an IMF Model of this, is to specify the functional requirements, the spatial arrangement, and the specification of a realization of the fluid separation system – not by graphical diagrams or drawings, but by creating an Information Model. The level of detail that is put into it – the granularity – is dependent on the use case.

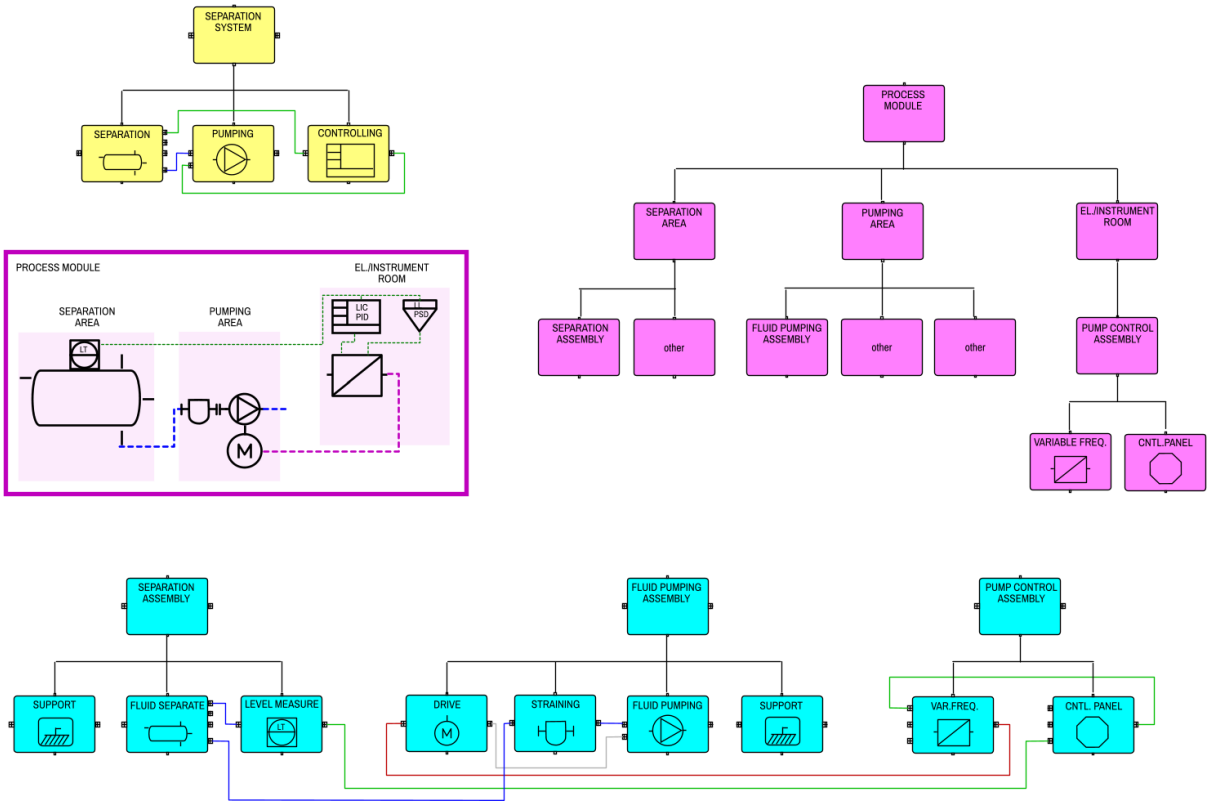


Figure 23: A separation system represented as an IMF model with three aspects: Function, Location, and Product.

When creating an IMF Model as part of establishing an early phase concept, the modelling can be at a very high level, and maybe only include the functional requirements. On the other hand, when creating an IMF model of a specific solution, such as a manufacturer's documentation of a pumping solution, the model can be very detailed and include all aspects of the pumping assembly. It is essential to understand the purpose of the modelling in each individual use case, and how this gives guidance to the best modelling approach.

5.1.1 Incorporating Modelling into an Established Work Process

Established work processes for engineering and design are often directed towards creating diagrams, drawings, and texts contained in documents which have a defined and fixed information scope. Typically, such a document only addresses one discipline, i.e., only Process or only Electro, and often only one sub-section of the Facility Asset. The level of detail is generally dictated by the type of document, i.e., an Overall Single Line Diagram shall only contain information about the main pathways of the electrical distribution system.

The fixed information scope of established document types, contrasts with the flexible information scope of an IMF Model. During the transition of work processes from document-based to model-based way of working there is a need to bridge between the two paradigms by referring to legacy document types when scoping a modelling exercise. For example, the scope for modelling is to define the same level of information content for the Facility Asset as typically represented in a Process Flow Diagram (a document type). As the industry gains experience from the transition from document-based to model-based, new work processes can be developed that are directed towards continuously enriching an Information Model rather than producing fixed format information.

When modelling a Facility Asset there are three different roles or mind-sets to choose from, depending on use case and life cycle phase:

1. Defining needs. That is setting requirements.
2. Specifying solution. That is fulfilling requirements.
3. Documenting the actual installation.

The line between requirements and specification is not always distinct, but in this context, requirements are about what is needed, whereas specification is about how to achieve it. In some use cases, such as when re-documenting an existing Facility Asset, the setting of requirements and specifications may have to be reverse engineered, based on the as-built documentation.

5.1.2 Defining a Need - Setting requirements

When developing a new Facility Asset, the first that must be done is to define the functional requirements, i.e., 'what is needed'. Essentially this is about what activity is needed, broken down into the individual activities it comprises. Taking the above separation plant given in Figure 23 as an example, the main activity needed of the Facility Asset is 'Separation', and the sub-activities are '3-phase separation', 'Pumping', and 'Controlling'. Each of these activity needs are further characterized by means of their attributes, e.g., 'Pumping' attribute 'Volumetric flowrate' = 200 m^3/h.

Activity requirements are captured in the Function aspect whereas requirements that are given by the intended location are captured in the Location aspect.

5.1.3 Specifying a Solution - Fulfilling Requirements

When specifying solutions that fulfil the requirements, the objective is to describe how to achieve the activities required, by means of components, equipment, and assemblies. The specified solutions must not only fulfil the activity requirements, but also the requirements given by the intended location, as well as any overall requirements applicable. Referring to the Separation plant given in Figure 23, the required pumping of 200 m^3/h must be fulfilled, say by a Centrifugal pump with a capacity of 300 m^3/h,

which is suitable for location in the Pumping area by being waterproof, and by meeting the standard API 611 (background) requirements.

Solution specifications are captured in the Product aspect whereas specifications of the space of the solutions and their positions are captured in the Location aspect.

5.1.4 Documenting the Actual Installation

Setting requirements and specifying solutions describes the intended Facility Asset. Once the specified components, equipment, and assemblies have been manufactured, delivered, or installed they can be documented by enriching the IMF Model with this actual information that is about the physical objects themselves. Thus, it can be documented that the specifications are met. The typical example is that the above Centrifugal pump may have a Nameplate capacity of 350 m^3/h and a Serial Nº 5270-2565.

Documentation of actual physical objects and their installation is captured in the Installed aspect (blue).

5.2 Before one Starts to Model

Before starting to model, we need to be clear on the purpose of the modelling. The purpose should be defined based on the end user needs, or the user needs next in the value chain. Figure 24 illustrates a separation system, shown as a conceptual diagram and as an imagined real solution.

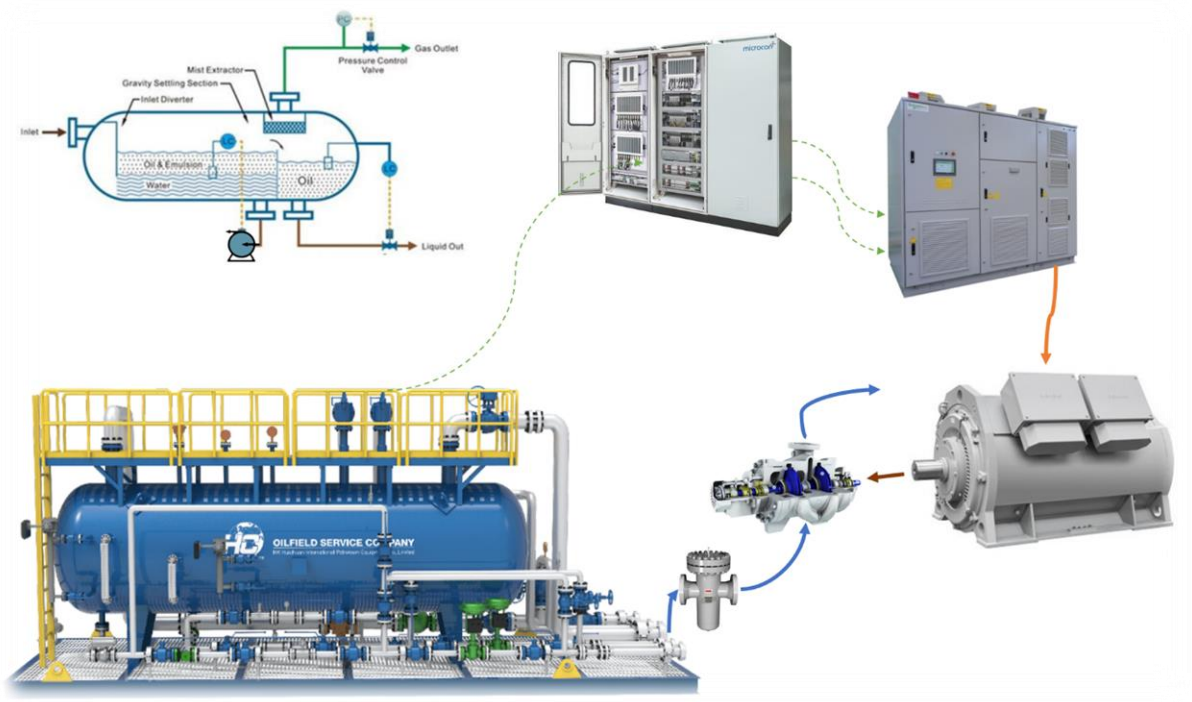


Figure 24: A separation system shown as a conceptual diagram and as an imagined real solution.

5.2.1 Defining the Purpose of the Model

The purpose of the modelling can range from defining requirements at a conceptual level – to documenting the actual Facility Asset as installed. Likewise, the purpose can be to model the automation and control part of the solution, or the fluid processes. In other cases, the purpose is to model a multi-discipline multi-aspect model at a high level of aggregation to provide a skeleton for later more detailed modelling.

Always begin by understanding and stating the purpose of what to model.

5.2.2 Framing the Overall Requirements

As for any engineering and design effort there are overall requirements that must be identified and evaluated to understand how they apply. The volume of such requirements increases with the level of detail of the modelling, but even for high level functional requirements modelling they must be considered. As an example, this could be safety requirements to systems and solutions handling high pressure hydrocarbon fluids.

5.2.3 Framing the Prior Governing Requirements

Usually, at an earlier stage, specific requirements and possibly also solution specifications have been developed, either captured in documents or as a Facility Asset model. This information must be understood as it governs the modelling when it continues.

5.2.4 Outlining the Scope of the Model and its Outside Interfaces

The IMF has a built-in taxonomy for granularity levels which can be employed to define the intended level of detail. This taxonomy is defined by ISO/IEC81346-O&G [5] which specifies the three levels:

1. Field wide (represented by a single letter code).
2. Technical system (represented by a two-letter code).
3. Component (represented by a three-letter code).

With only three granularity levels built in, there may be a need for additional and more fine-grained taxonomies, such as that specified by ISO 14224 [6] or other industry standards or conventions. It is important to remember that granulation detail levels are not the same as system breakdown levels. For example, in an Information Model of a Facility Asset, a Technical System can be part of another Technical System. With IMF being an open format, it is up to the user to employ the most fit-for-purpose taxonomy (that can consist of one or several) to define the intended scope and detail of the modelling.

Likewise defining the scope, it is important to define the outer interfaces. That is stating where does this model end and another model continue? Such interfaces include upstream and downstream interfaces as well as interfaces between different disciplines, between aspects, or between IMF Models.

5.3 How to Choose which Aspect to Begin with

The logical sequence of aspects is:

Function => Product => Location => Installed,

However, is often both necessary and valuable to iterate between aspects, or between setting required activities, specifying fulfilling solutions, and defining and allocating space and locations. However, it may help structure the work process to choose which aspect to begin with and think of as the ‘master’ aspect.

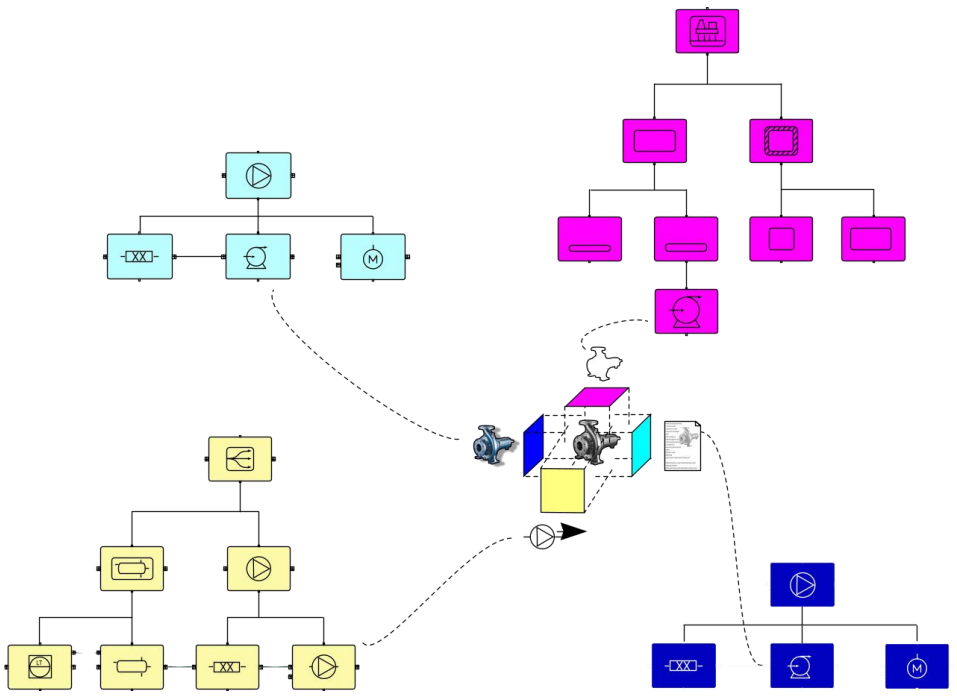


Figure 25: A pump system represented as an IMF model using four aspects: Function, Location, Product, and Installed.

5.3.1 The Function Aspect

Choose the Function Aspect when the objective is to set requirements to activities (what it is going to perform) of the Facility Asset. Often this is logically the first step.

Move from the Function Aspect when required activities have been defined to such a level of detail that solutions can be specified to fulfil them.

5.3.2 The Product Aspect

Choose the Product Aspect when the objective is to specify solutions comprising components, equipment, or assemblies. Often this is the logical second step, assuming a first step has been to set requirements. When creating a model of an already existing design this is the first step. To specify a solution may take place in several steps: first by the engineering contractor as part of a procurement process, secondly by the vendor or manufacturer as part of a bidding process, and sometimes also in further steps involving special expertise on topics such as material technology.

Move from this aspect at the latest when the solution has been sufficiently specified for order placement.

5.3.3 The Location Aspect

When modelling, choose the Location Aspect when the objective is one of two:

1. to specify the spatial and positional properties of a component, equipment, or assembly.
2. to state the requirements that apply within the given location, e.g., noise limitations and ambient conditions.

Modelling in this aspect depends on knowledge about what will eventually be needed to be located, both the size, weight and vulnerability to ambient conditions, as well as safety, access, and maintenance needs. Therefore, the modelling in this aspect depends on the use case and work processes.

Move from the Location Aspect when locations and their requirements have been defined to such a level of detail that solutions can be specified to fulfil them.

5.3.4 The Installed Aspect

When modelling, choose the Installed Aspect when the objective is to document the physical reality, i.e., the actual component, equipment, or assembly. Contrary to what the name of this aspect indicates it does not have to be actually installed (on site) in order to be documented thus. It must be manufactured. Said in other words, it must exist. This means that this aspect can also be used to document components, equipment, or assemblies that are in storage, possibly as spare parts.

This aspect is different from the three aspects above as it does not *model* the Facility Asset, it *documents* it.

5.4 How to Decide which Modelling Approach to use and the Initial Aspect Objects

There are different approaches for how to model, with regards to where to begin and in which direction to progress. In the following sections we will use the IMF Model given in Figure 26 to illustrate the different approaches.

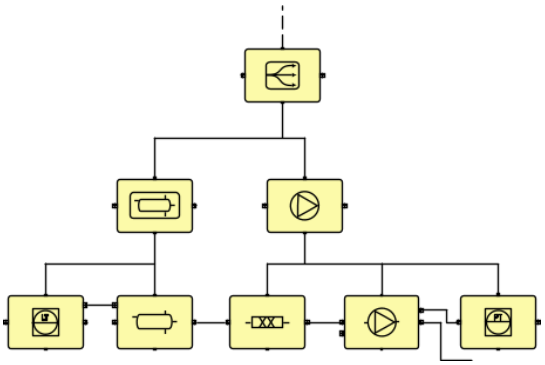


Figure 26: An example of an IMF Model

The approach to select for any given case is to some extent use case driven, but there are multiples of case-to-case conditions that influence the optimal approach. Unless there are clear arguments for doing otherwise, the default choice is the top-down approach.

5.4.1 A Top-down Modelling Approach

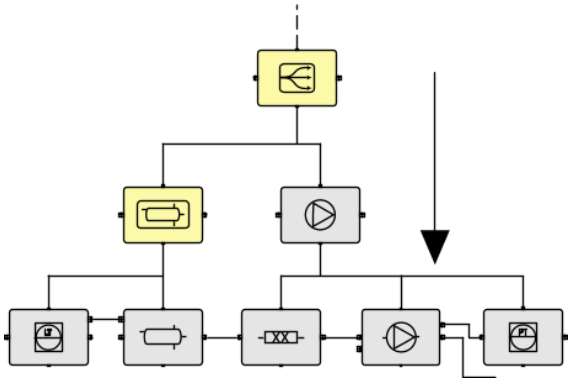


Figure 27: A Top-down modelling approach.

The top-down approach shown in Figure 27, closely mirrors the Systems Engineering methodology, i.e., breaking higher level systems down into their constituent sub-systems, iteratively refining and detailing the model. It allows someone with a high-level understanding of the Facility Asset requirements to establish a design basis which thereafter can be further detailed by someone with more specialized expertise on individual sub-systems. Grey coloring indicates Aspect objects yet to be modelled.

5.4.2 A Bottom-up Modelling Approach

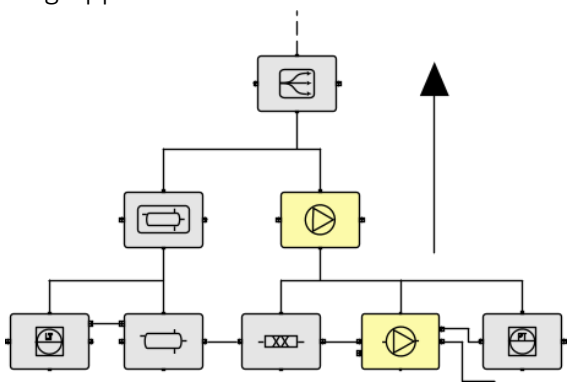


Figure 28: A bottom-up modelling approach.

The bottom-up approach shown in Figure 28, supports the process of integrating sub-systems into higher-level systems in an optimal way. Typically, this is needed if any sub-systems are pre-defined in some way, usually because they represent standardized or off-the-shelf systems. Grey coloring indicates Aspect objects yet to be modelled.

5.4.3 A Follow-Stream Modelling Approach

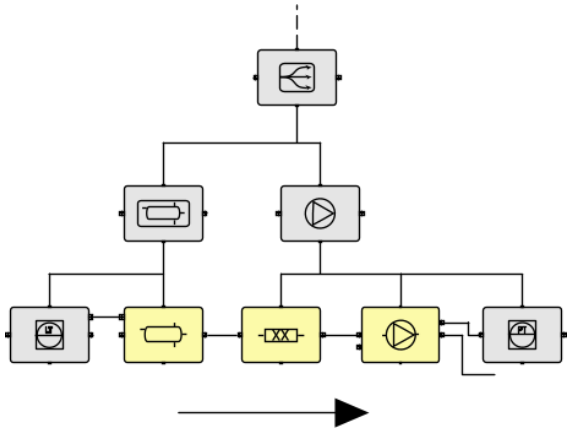


Figure 29: A follow-stream modelling approach.

The follow-stream approach shown in Figure 29, allows full attention on how media shall stream into and through the Facility Asset, ultimately resulting in the required media output. Following this approach, the view taken is that the individual sub-systems are only a means for defining input(s) and output(s) and the required transformation in between. What results from this approach is therefore a topology (horizontal connections), to finish such a model will require a bottom-up integration modelling effort. Grey coloring indicates Aspect objects yet to be modelled.

5.4.4 A Follow-thread Modelling Approach

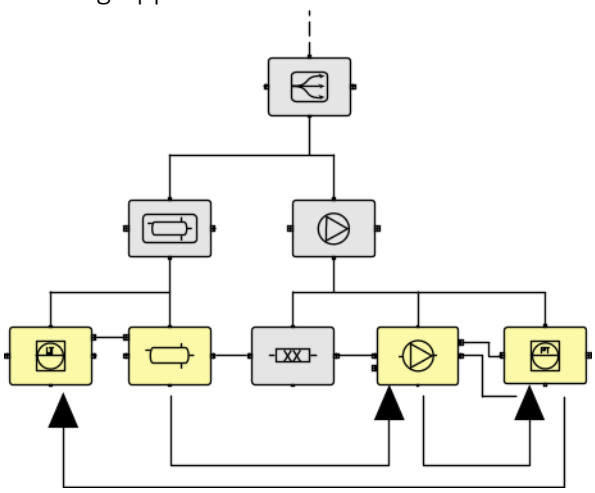


Figure 30: A follow-thread modelling approach.

As illustrated in Figure 30, follow-thread approach is well suited for multidiscipline modelling. The threads can be explained thus:

1. Separation is required, and for it to work optimally a pump is needed.
2. For the pump to do its duty it needs a control system.
3. For the control system to work it needs a level sensor which is connected to the separator.

The disciplines involved in this thread are Process, Process Control, and Instrumentation. What results from this approach is therefore a thread of defined dependencies between systems, and to finish such a

model will require a bottom-up integration modelling effort. Grey coloring indicates Aspect objects yet to be modelled.

5.5 How to Create and Connect Aspect Objects

The building blocks of an IMF model are the various types of Aspect Objects. The Aspect Objects are fetched from a library, or more precisely they are created from definitions called IMF Types that reside in the IMF Type library. When an aspect object is created – or instantiated – it has several attributes defined, reflecting what it represents. For example, if it represents Pumping it is to be expected that it has an attribute named 'Volumetric flowrate', and other attributes that characterizes 'Pumping'.

To model requires to create aspect objects and connecting them together. This is done by placing them into the breakdown hierarchy and connecting them together between outputs and inputs, and by defining relations to aspect objects in other aspects. Depending on the modelling approach chosen, note that not all these different types of connections need to be defined initially.

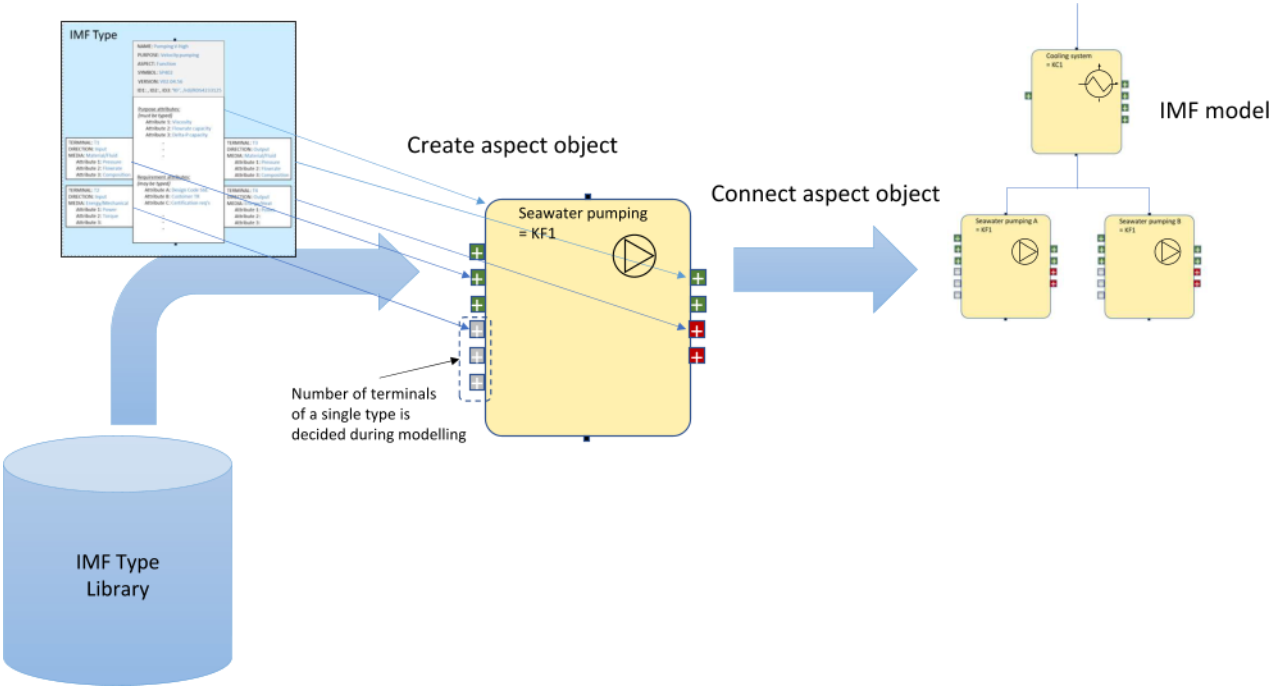


Figure 31: Using an IMF Type from a library.

5.5.1 Selecting an IMF Type

The IMF Type library holds a large volume of IMF Types, and they represent different levels of specialization. For example, an IMF Type 'Liquid Pumping' represents pumping at a generic level, whereas an IMF Type 'Centrifugal high dp liquid pumping' represents a much more specialized pumping. When selecting an appropriate IMF Type, it is therefore not sufficient to only select the desired Purpose (=Pumping), but also the desired level of specialization must be chosen.

Prior to selecting the IMF Type, the working aspect must be selected (Function, Product, Location, or Installed).

5.5.2 What if the Needed IMF Type does not Exist?

If a close enough match cannot be found between what is needed and what is available in the IMF Type Library, then a new IMF Type must be created. The IMF Type library is a common industry resource, which means that creating new IMF Types demands access to applicable tools, as well as having the mandate. See Chapter 6.

5.5.3 How to Place the Aspect Object into the Model

When an Aspect Object has been fetched (created) from the library it can be placed into the model by means of connecting it. There are three kinds of such placements/connections:

- Into the hierarchy of existing aspect blocks - effectively creating partOf relations.
- Into the topology of existing aspect blocks - effectively creating connectedTo relations.
- Into the relation between aspect objects of different aspects - effectively creating fulfilledBy relations.

Depending on the modelling approach, one or more of these connections need to be made early, but ultimately all must be made to finish the modelling.

5.5.4 Set Attribute Values of the Aspect Object

When a new Aspect object is created during modelling it will only have *one* attribute of each Quality. This is provided as part of the IMF Type definition. Qualities are such as 'Pressure', 'Temperature', 'Speed' etc. It is part of the modelling work process to increment as needed the number of attributes of the same Quality, e.g., increment to 3 'Pressure' attributes.

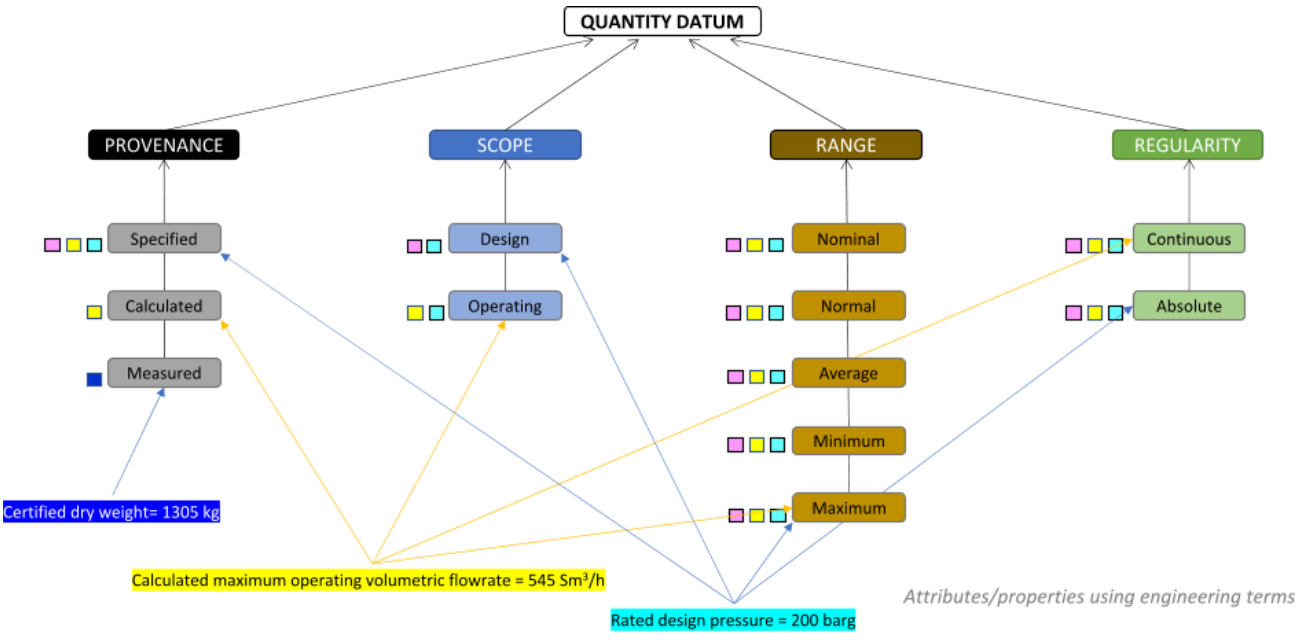


Figure 32: Typing of attributes to provide context.

It is a part of the modelling work process to further define the type of attribute by assigning it to Quantity Datum classes, as shown in Figure 32. Some examples are:

- Pressure: Specified/Design/Maximum/Absolute
- Volumetric flowrate: Calculated/Operating/Maximum/Continuous

Conventional engineering terms may differ from this classification scheme, but the meaning is the same. As an example,
'Rated design pressure' corresponds to 'Pressure: Specified/Design/Maximum/Absolute'.

Once the number of attributes has been entered and their Datum types are selected, they are ready to be given values and units of measure selected. As an example, the Pressure: Specified/Design/Maximum/Absolute may be given a value of 200 barg.

Usually only a few attributes are needed to be given a value to provide a sufficient description. This must be judged in each individual case, and there are no firm rules. As an example, when defining a pumping function, it may be essential to enter a value for the volumetric flowrate and differential pressure, as this characterizes the pumping, but it may be less important to enter a value for rotational speed as this is not about what the pumping produces.

5.5.5 Where Attribute Values Reside

The actual attribute values may reside in the model data itself, or they may reside in other engineering registers or applications and be available as linked data, or they may reside in external systems and not be available as data, but only be pointed to. Where the attribute values reside therefore depends on the actual implementation of the modelling tools and applications. Some examples of how attributes could reside in different places:

- Model data:
    - Activity= Pumping
    - Volumetric flowrate: Calculated/Operating/Maximum/Continuous = 500 m^3/h
- Linked data:
    - Speed= 3000 rpm [/engineering_register/separation_system/pump1/speed]
    - Temperature: Calculated/Operating/Maximum/Continuous= 120 ˚C [/engineering_register/separation_system/separator1/temp]
- Pointed-to:
    - Technical Requirement Specification 0023423-55B

This means that in some implementations *all* attributes reside in the model data (could be a modelling tool for conceptual design), whereas in some other implementations *no* attributes reside in the model data but are in external engineering registers. Most likely the most optimal solutions lie somewhere in between the two extremes.

5.5.6 Allocate Input- and Output Terminals

When a new Aspect Object is created it will only have one Terminal of each type. This is provided as part of the IMF Type definition. A Terminal is defined by being an Input or an Output, and by which type of Media it pertains to, such as 'Energy/Electrical' or 'Material/Fluid'. It is part of the modelling work process to increment as needed the number of Terminals of the same type. For example, increment to 2x Input: Energy/Electric.

5.5.7 Setting Attribute Values on the Terminals

Attributes of a Terminal will only have *one* attribute of each Quality as specified by the IMF Type definition. Qualities are such as 'Force', 'Voltage', 'Power', etc. It is part of the modelling work process to increment as needed the number of attributes of the same Quality for a terminal as needed, e.g., increment to 2 'Voltage' attributes. Furthermore, the attributes need to be typed by assigning them to Quantity Datum classes. Examples are:

- Voltage: Design/Maximum/Absolute
- Voltage: Operating/Normal/Continuous

Once the number of attributes has been entered and their Datum types are selected, they are ready to be given values and units of measure selected. As an example, the Voltage: Design/Maximum/Absolute may be given a value of 6.6 kV.

5.5.8 Connecting Terminal to Terminal between Aspect Objects

Aspect Objects will have Input and Output Terminals (except for Location Aspect objects), and they are the means for modelling how streams flow. An output Terminal of one Aspect Object needs ultimately to be connected to the Input Terminal of another Aspect Object in order to model a stream. Note that the transport and containment itself, of the stream, may also need to be modelled. If so, it must be represented by an Aspect object with one Input Terminal and one Output Terminal. Examples are cable, pipe, drive shaft.

When modelling in the Function aspect the stream connection can simply be across the system border of the two systems, i.e., there is no separate transport.

When modelling in the Product aspect there could be a cable or pipe which itself may need to be modelled. There are also cases in the Product aspect where there is no such transport, such as with a flange-to-flange connection between two equipment.

5.5.9 Iterate on Creating and Editing Aspect objects

As more and more Aspect Objects are created and connected into the IMF Model it becomes richer and richer on Facility Asset information. As the design progresses, it is part of a natural workflow that Aspect Objects need to be edited and possibly also reconnected elsewhere in the IMF Model. The flexibility that this kind of modelling offers allows modelling to begin even when very little information is known, and then repeatedly review and enrich the IMF model until the desired level of accuracy is reached.

5.5.10 Understanding and Managing the Consequences of Changes

The high level of flexibility of modelling also means that changes can be made frequently, and this carries the risk of causing inadvertent changes. The consequences of one small change can reach much longer than when working on conventional documents, where a change often stays local. It is therefore important to understand what the potential far-reaching effects of a change could be. A change to an Aspect Object could mean that other Aspect Objects to which it is connected are also affected by the change, e.g., changing something in a system is likely to affect the system it is partOf. To alleviate this, the tool used for modelling may have mechanisms that warn about such consequences, but how effective these are depends on the particular tool.

5.5.11 Conditions for Shifting the Modelling Aspect

It can be challenging to understand when to model in the Function aspect and when to model in the Product aspect. In many cases there is no sharp line between them. This issue is also recognized from document-based design processes, where the art of writing truly *functional* requirements is difficult. Very often there is a need to include some solution specifications. By principle, solution specifications shall be modelled in the Product aspect, but IMF modelling allows for flexibility in this regard. It does not force all requirements to be in the Function aspect, nor does it force all specifications to be in the Product aspect. A pragmatic approach should be taken, aiming at an IMF model which is fit-for-purpose in each use case.

The Location aspect is more distinct and is entirely different from the other aspects. Because this aspect will hold requirements to component, equipment, or assemblies located there, there may be situations when these are modelled in the Product aspect and the specification reveal that they cannot meet the requirements set by their intended location. In such cases it may be needed to shift to the Location aspect and revise the location requirements, alternatively to change where they shall be located.
The Installed aspect is solely used for documenting real world physical objects.

5.6 Connecting Relations Between Aspects

When working with multiple aspects it may be necessary to specify how something is modelled in the different aspects. This is done by connecting inter-aspect relations, as shown in Figure 33.

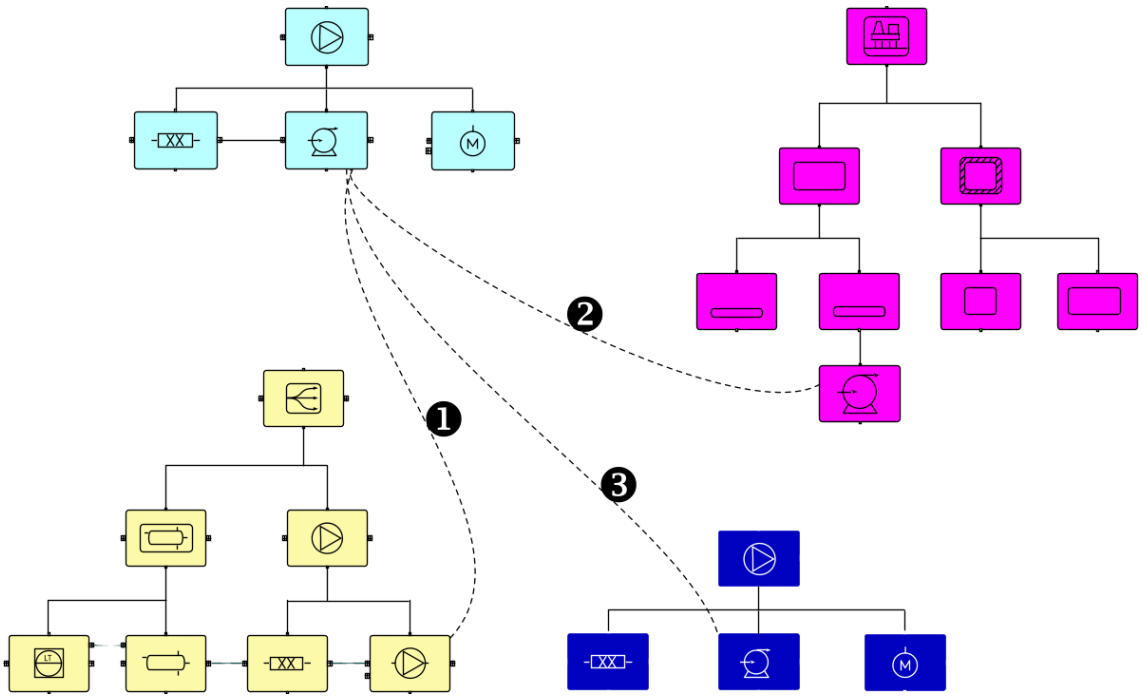


Figure 33: Inter-aspect relations

The relations can be set in any order, but usually the sequence will be as per the numbering of the figure.

1. The logically first is the relation which specifies that activity requirements modelled in the Function aspects are fulfilled by solution specifications modelled in the Product aspect. In this

example the requirements to the pumping activity are fulfilled by the specifications of the pump.

2. The logically second is the relation which specifies two things:
    a. That the solution specification modelled in the Product aspect occupies a specific space and has a particular relative position in a particular location, which is modelled in the Location aspect. In this example the specified pump has a defined size and is relatively positioned in a defined location.
    b. That the solution specification modelled in the Product aspect fulfils the location-based requirements modelled in the Location aspect. In this example the specified area where the pump is located is specified as exposed to weather and is met by that the pump is specified as weatherproof.

Note that for the Location aspect requirements are given by the parent aspect object. E.g., when a pump is located in a process area it is the requirements of that process area that apply to the pump.

3. The relation between the Product aspect and the Installed aspect serves to document that an actual component, equipment, or assembly fulfils the solution specified. Note that there may be several fulfilling the same specification, as is the case for spares in storage.

Not all Aspect objects will have relations to other aspects. An Aspect object in Function which is modelling a large system will almost never have a relation to one solution specified in Product. An area or room or other such location modelled in Location will almost never have a relation to one solution specified in Product.

5.6.1 Connect Function-Product relation

When the requirements to a system of activities have been broken down to such a level of detail through modelling that it is feasible to fulfil the requirements in a solution specification, then a relation between requirement and solution can be established. This is done by connecting the relation between the Function Aspect object representing the requirements, and the Product aspect object representing the solution specification.

5.6.2 Connect Product-Location relation

When the arrangement of the intended locations (rooms, areas, etc.) have been modelled to such a level of detail that a location is available for a specified solution, then a relation between location and solution can be established. This is done by connecting the relation between the Product Aspect object representing the solution, and the Location Aspect object representing the spatial properties of the solution, *and* by connecting the Location Aspect object into the Location hierarchy, i.e., stating which location it is part_of.

5.6.3 Connect Product-Installed relation

When a component, equipment, or assembly is manufactured, delivered, or installed, the documentation of it can be linked to its solution specification, by means of connecting a relation between the Location aspect object representing the actual component, equipment, or assembly, and the Product Aspect object representing the solution specification.

5.6.4 Connect Relations to other Aspects

A Facility Asset model can be further augmented by introducing new aspects, such as an Engineering Numbering aspect (TAG aspect) which can enable relations between requirements, solutions, locations, and their nameplate TAG numbers. Likewise, a Maintenance aspect can enable relations between component, equipment, or assembly and their maintenance task.

Connecting relations to these possible future Aspect objects will follow the same principle, but is yet to be defined.

5.7 IMF Model Integration

IMF will allow modelling smaller IMF models that later are integrated into larger models, as conceptually shown in Figure 34. More advanced mechanisms for supporting this is planned for a later version of IMF. The integration can be between models in different Aspects (1) or between different multi-aspect models (2).

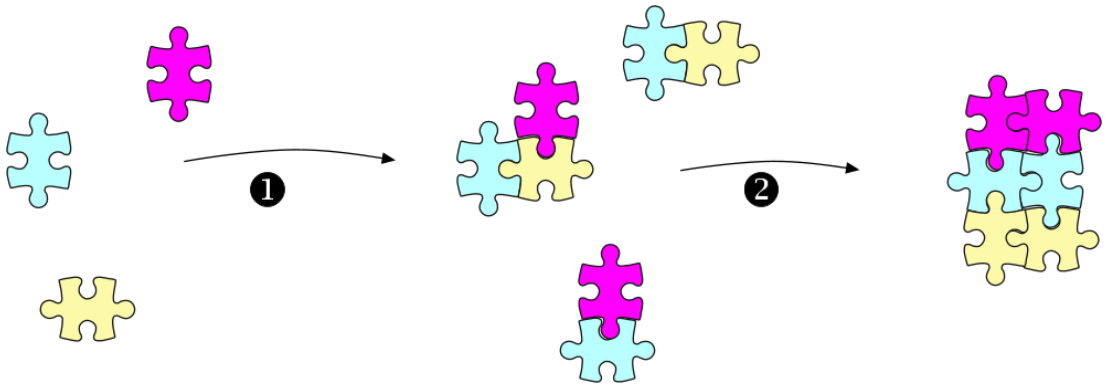


Figure 34: The concept of model integration

This supports a work process where the design work is split onto different resources for later to be integrated into a whole. The integration between aspects is achieved through connecting relations between aspects, as described in previous chapter, whereas the between different multi-aspect models requires managing somewhat complex interfaces between the models. The integration of several models in the same Aspect is simply achieved by placing them into a common hierarchy by means of the partOf relation.

5.7.1 Managing integration of two IMF Models

Integrating one IMF Model with another IMF Model requires managing the interfaces between the two models, as highlighted with focus rings in.

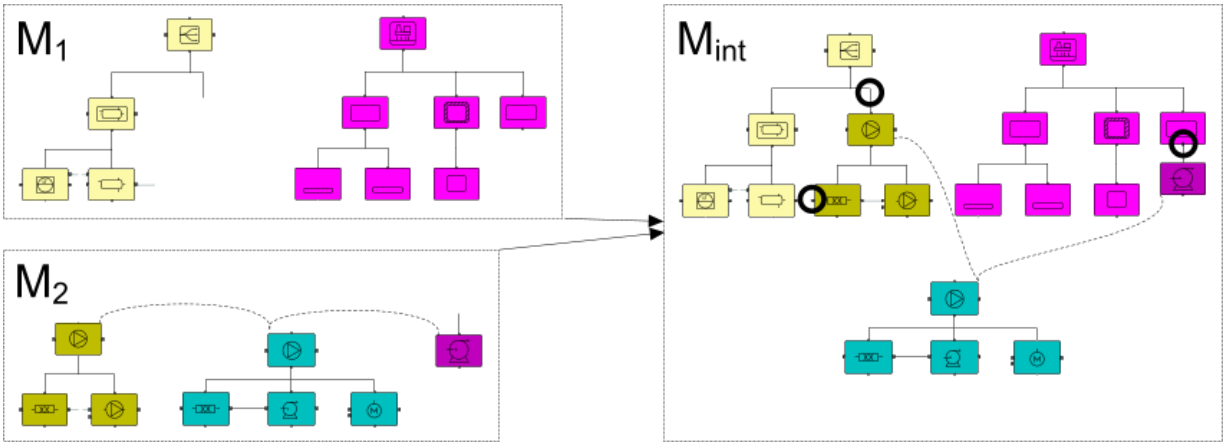


Figure 35: Model integration

When the IMF model M_1 is integrated with the higher-level IMF model M_2 what results is the integrated result of the two, M_{int}. To do this successfully all the connecting points – interfaces – must be carefully managed by performing the appropriate modelling. Integrating same level models will be similar. Splitting out a smaller model from a whole model – which is the reverse operation – similarly requires that the interfaces are managed. To model an integration or split implies making or unmaking any or all of partOf relations, fulfilledBy relations, and Terminal connections.

5.8 IMF Model Examples

5.8.1 A Process Performance Requirement Model

This example is a simple three-phase hydrocarbon separation system where the water level is controlled by pumping. The purpose of this modelling is to describe the system and its performance requirements. To do this, the entire system and the individual sub-systems needed as part of it, are modelled, namely the separation activity, the pumping activity, and the controlling activity. No decisions regarding a solution specification are made, and therefore the entire model is in the Function aspect. The model is illustrated in Figure 36.

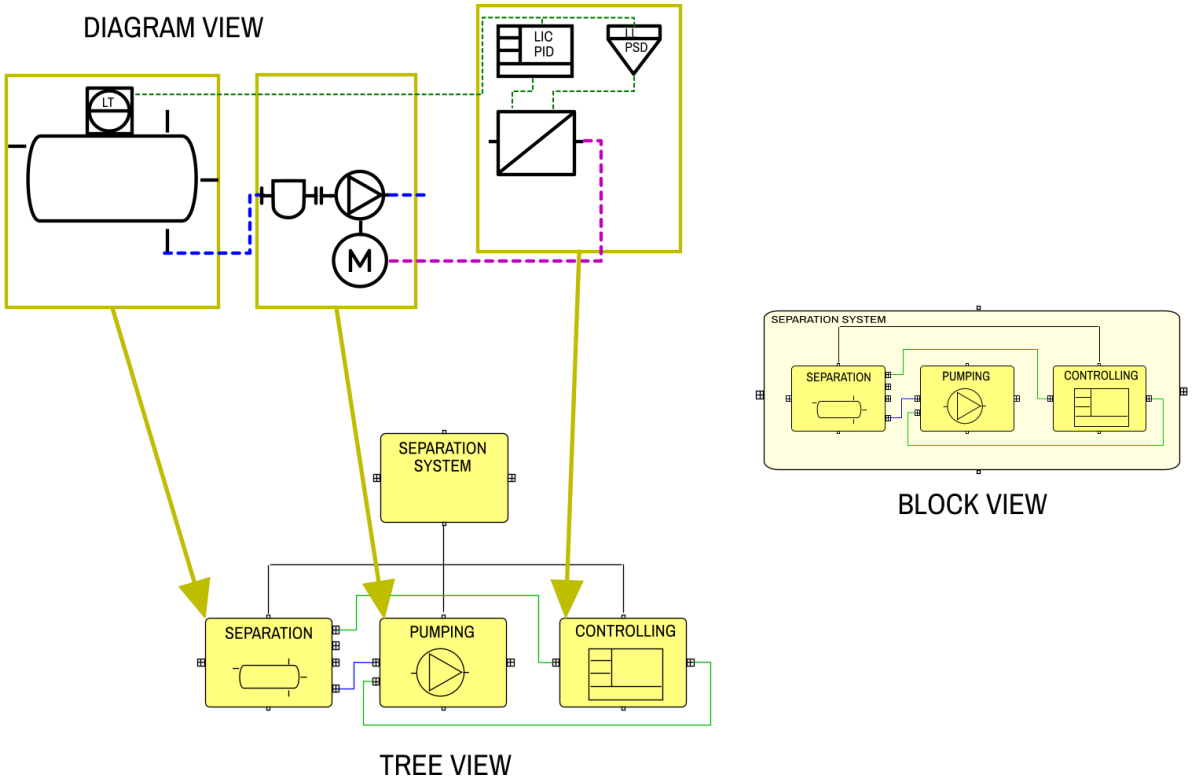


Figure 36: Separation system modelled in Function aspect

The Function aspect hierarchy defines the system break-down structure, and the streams define how the systems are connected together. Each of the Function aspect objects define the requirements to what they represent, e.g., the "Pumping" Function aspect object defines the requirements to the pumping activity, such as required flowrate and differential pressure. To set these requirements by modelling is done by assigning values to the attributes of the Function aspect object, e.g., by setting the differential pressure to 20 bar.

Requirements are defined at different levels of detail, as shown in this example, where the entire separation system is represented at high level by one "Separation system" Function aspect object. At this level the stated requirements could be limited to the required capacity to perform separation of crude oil of a specified quality.

To help interpret the figure, the same processing plant is shown in three different views: Diagram view is for similarity with the formats known from drawings such as Process Flow Diagrams, System Control Diagrams, and One Line Diagrams. Tree view is a view of the model which emphasizes the break-down structure, and Block view is a view of the model which emphasizes the topology (streams).
If there is a need for further detailing the requirement, such as for the pumping performance, further break-down of the pumping system could be modelled, e.g., by breaking the Pumping into Filtering, Pumping, and Driving – each represented by a Function aspect object.

5.8.2 A Multi-Discipline Requirements-Thread Model

This example is a process facility, and the emphasize is on how modelling is directed by the thread of requirements, as illustrated by the red line in Figure 37. No decisions regarding a solution specification are made, and therefore the entire model is in the Function aspect.

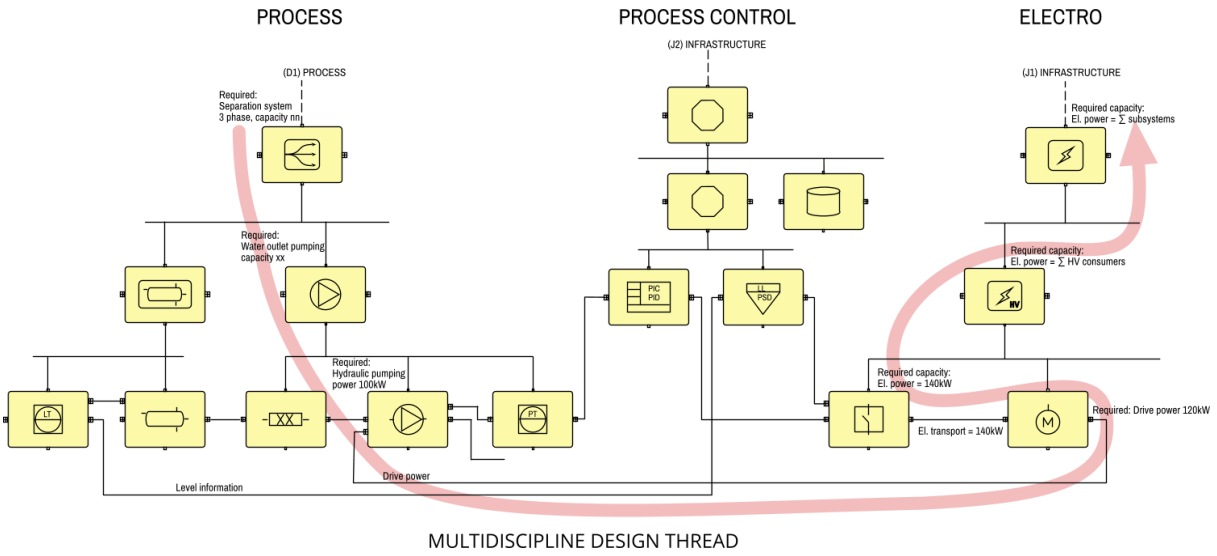


Figure 37: Modelling directed by the thread of requirements.

This particular requirement thread ties high level Process requirements to high level Electro requirements. Other requirement threads (not shown) similarly flow from Process, to Process Control. In the example the Separation requirement is broken down into Separation, and Pumping. Pumping requires mechanical energy, so it must be connected to Driving (motor). Driving requires controlled electrical energy, so it is connected to electrical distribution through a switch. The required electrical energy adds to the required total electrical energy capacity of the facility. As can be seen from this exercise, the conventional division into different disciplines is irrelevant, and instead a multi-discipline approach is enforced.

5.8.3 A Catalogue Equipment Specification

This example is of a small centrifugal pump including an electrical motor. It is a catalogue product, and the purpose of the modelling is to build a specification of this solution, which shall potentially fulfil some customer's requirement to pumping. The example is illustrated in Figure 38.

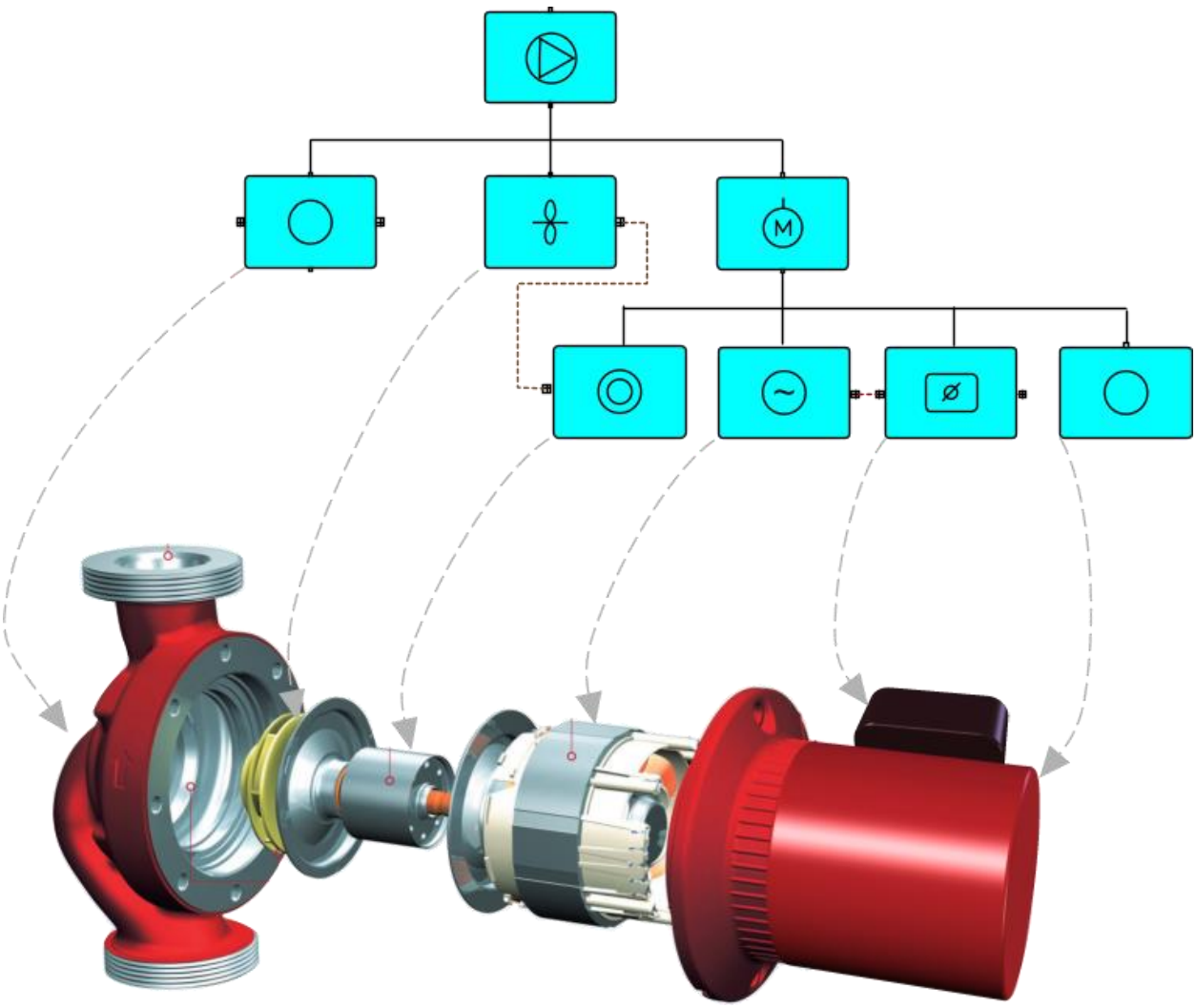


Figure 38: Pump Product model

For illustration the pump is shown in exploded view. It comprises – from left to right – a pump housing, an impeller, a motor rotor, a motor stator, a motor housing, and a motor termination box.

In the model, the breakdown specifies the assembly of these parts, meaning that the partOf relation should be understood as assembled with. Each part shown is represented by a Product aspect object, which provides a specification of that part, including input and outputs specified as Terminals. E.g., the pump housing has an inlet and an outlet, which is represented as Input Terminal and Output Terminal that specify attributes such as connection type and dimension. The motor rotor has an Output of type Energy/Mechanical/Rotating which here is connected to the impeller Input. Not all such connections are shown (for clarity) including the mounting of the motor housing to the pump housing, which would be modelled as Input and Output Terminals of type Force/Mechanical.

Likely one or several of the parts are also used for other catalogue pumps, in which case they need only be modelled once, and be reused across a range of pumps.

The level of detail of specification is dependent on use case, and IMF allows any level of modelling to be done, both as regards break-down into smaller parts, as well as how rich the set of attributes for each part is.

5.8.4 A Facility Asset Architecture Model

This example is of the area architecture of an Oil & Gas Production Platform. The drawings in Figure 39 show a side view and several elevation views of the facility. Information about the location of components, equipment, and assemblies are shown as TAG numbers on the drawing, but without exact positions. Information about the requirements given by a location, such as the open weather conditions on the Weather Deck is not shown at all. The purpose of modelling this architecture in the Location aspect is to specify all such information in the context of the location.

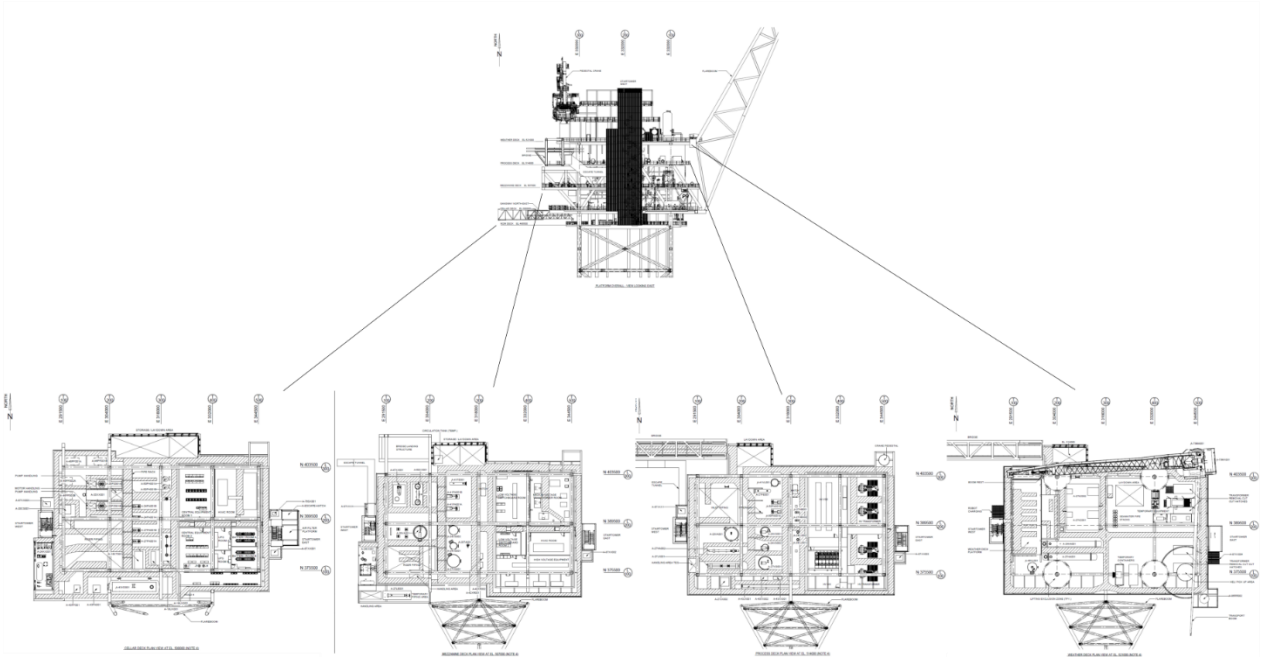


Figure 39: Plot Plans for a Platform

To model a Facility Asset architecture the best approach is to begin at the top, in this case the Platform, and then break down into decks, modules, areas, rooms – until the level of detail is reached when all the components, equipment, and assemblies that is or will be modelled in the Product aspect can be placed in locations, i.e., modelled in the Location aspect.

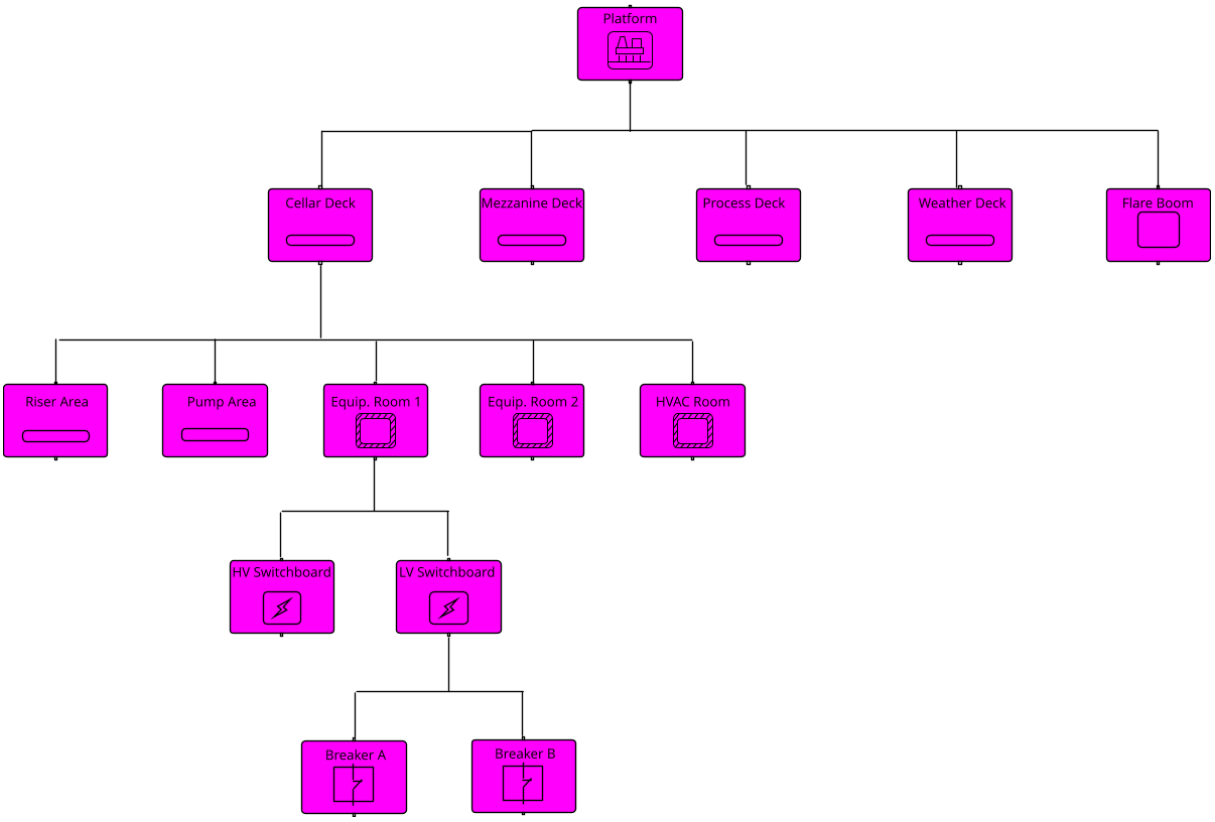


Figure 40: Model in Location aspect

Above shows how the Platform shown on the Plot Plan drawings can be modelled in the Location aspect. For clarity only one location has been broken down to a level of detail ready for locating components, equipment, and assemblies, in this case, electrical equipment. In this example the Equipment Room 1 is likely to be stated as dry, ventilated, and cooled area. This means that there is no requirement to the Switchboards located there to be waterproof. Note that if the LV Switchboard were to be relocated to the Weather Deck it would be subject to the conditions and requirements in that area and would likely be required to be waterproof.

6 How to Specify IMF Types

This chapter gives an introduction and step by step guideline of how to create an IMF Type. Prior understanding of the IMF as a framework and language is assumed. Understanding of Reference Data Libraries is beneficial but not required. When explaining how to create IMF Types, and advising on different approaches to take, some examples are provided to make the explanations more concrete. These examples cover only a few of the possible use cases, and the range of IMF Types is likely to be much wider.

6.1 What is an IMF Type?

IMF Types are building blocks used to model a Facility Asset. An IMF Type is a definition from which Aspect objects are created.

6.1.1 IMF Type and its Role in Information Modelling

The IMF Type provides a pattern with which building blocks are created. The appropriate IMF Type is selected from an IMF Type library. The IMF Type library is an industry shared resource and thus enables standardization as well as minimizing duplication of work. Prior to being used during modelling IMF Types need to be specified and created such that the IMF Type library has sufficient contents.

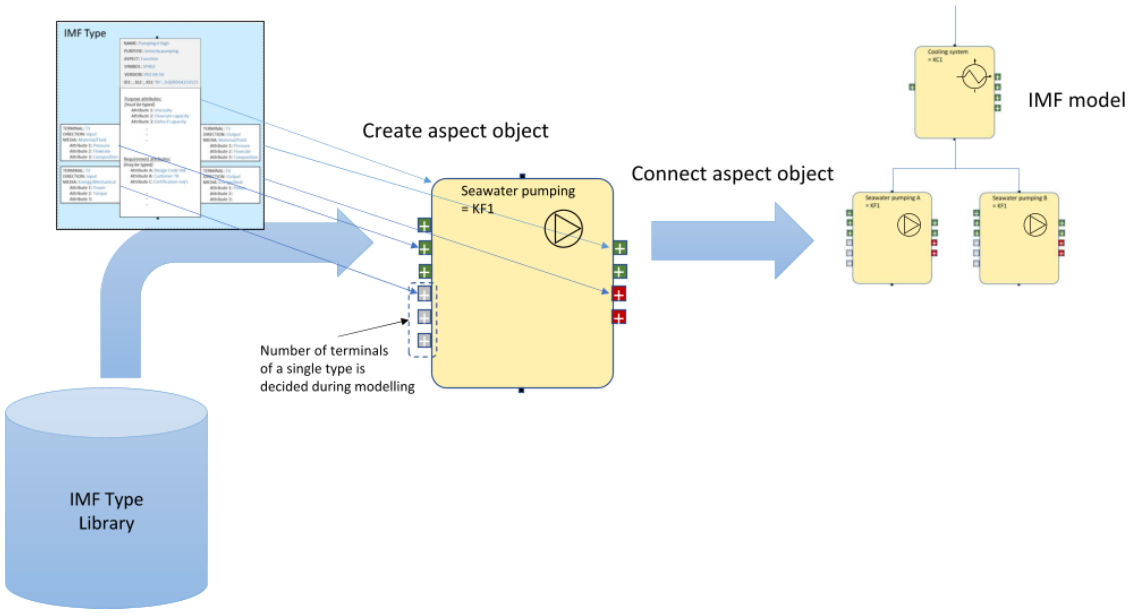


Figure 41: Creating and using an Aspect object from an IMF Type

6.1.2 IMF Type Information Content

The IMF Type contains the information needed to define a building block. There are variants of building blocks, and correspondingly there are variants of IMF Types. The information content differs between the variants but adheres to the same basic structure. The properties of an IMF Type are sourced from Reference Data Libraries, which are industry shared resources and thus enable standardization as well as minimizing the duplication of work. IMF Types are intended to reflect the need of SME for building blocks with which they can describe elements of a Facility Asset. Therefore, the primary input into defining an IMF Type is from the SME. The means for compiling such domain knowledge could e.g. be Digital Datasets or conventional Datasheets.

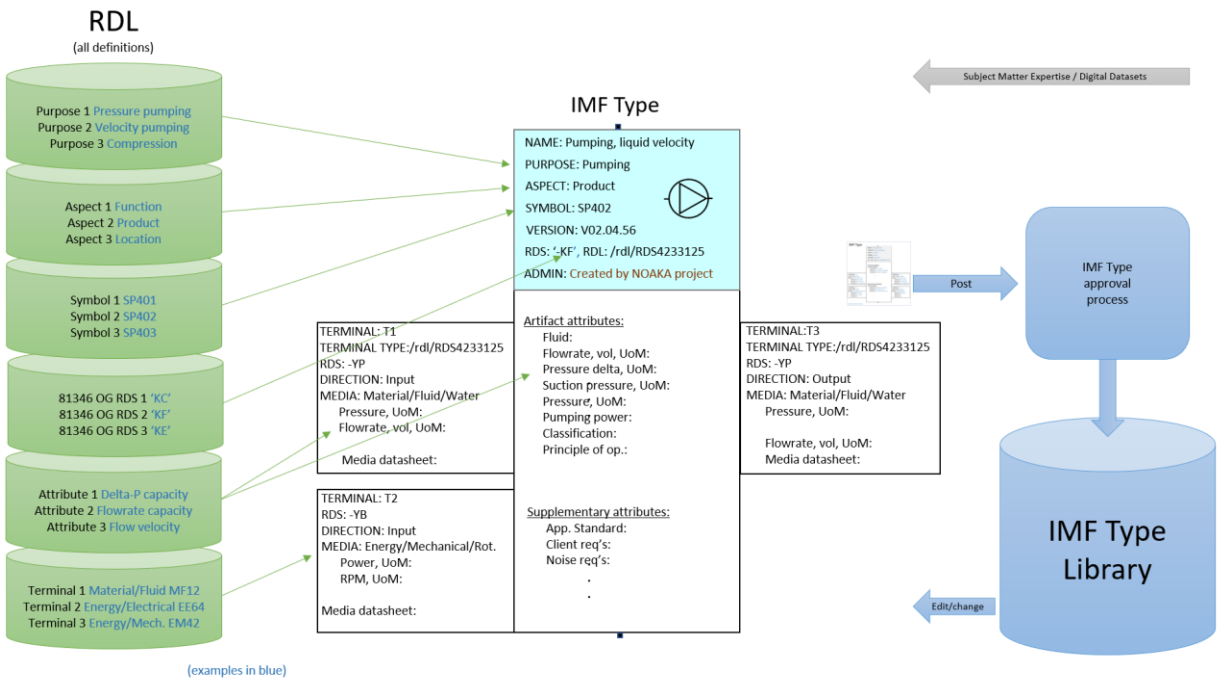


Figure 42: Properties of an IMF Type are sourced from Reference Data Libraries

6.1.3 IMF Type variants: Aspect Object and Aspect Interface Point

Two variants of IMF types have been defined: the Aspect Object variant, and Aspect Interface Point variant. The need for these variants stems from the need for having two kinds of modelling building blocks, as described in Chapter 4. Definition of the Aspect Interface Point IMF Type is planned for IMF Reference Manual version 2.

6.1.4 IMF Type Aspect of Information: Function, Product, Location, Installed

The description of an element of a Facility Asset differs depending on which Aspect of the element has focus. Therefore, Aspect Blocks and Aspect Interface Points representing each Aspect are needed, and consequently, IMF Types representing each Aspect are required. The Aspects: Function, Product, Location, and Installed have been defined in this document. For purposes such as relating to work processes (e.g., Procurement), or external data structures (e.g., Tagging) creating IMF Types in other Aspects may be valuable. The IMF language is open in this respect and allows definitions of new aspects.

6.2 Creating an IMF Type

6.2.1 Target Group

The primary target group for creating IMF Types is the engineering domain SME. To be able to create types, the needed Reference Data must be available. Early on, this data is likely to be incomplete, which means that there is a need for close collaboration between the SMEs and the group responsible for creating and maintaining the Reference Data, such that needs can be identified and incorporated. This should be supported by a tool and process where the SME can propose RLD extensions as part of a defined work process.

6.2.2 Proactive versus Reactive Workflow

To build a comprehensive IMF Type Library requires a significant investment of expert time and resources. Two approaches to this work are valid; proactive, and reactive. *Proactive* implies creating IMF Types up front, with no specific Facility Assets in mind, but aiming at creating the likely most used IMF Types irrespective of particular Facility Assets. To scope out what is 'likely most used' requires understanding of which disciplines comes first, of what industry domains have priority, and at what level of granularity will modelling be done during the first stages of Facility Asset modelling in the industry.

Reactive creation of IMF Types complements the proactive approach in that, as modelling of actual Facility Assets progresses, invariably new needs for IMF Types will be revealed, and have to be created in order to enable the modelling to progress.

6.2.3 Determining the Aspect of the IMF Type

When setting out to define an IMF Type with a defined Purpose, it is a prerequisite to understand what the different Aspects represents. Only by understanding and choosing the appropriate Aspect from which this IMF Type is about to be created, is it possible to choose the applicable Attributes and Terminals.

- **In Function:** In this Aspect the need for an essential activity or function is specified, without knowledge or decision about how this need may be fulfilled.
- **In Product:** In this Aspect it is s specified how to solve and fulfill a need by means of a product or component.
- **In Location:** In this Aspect the spatial properties are defined, including dimensions, relative location, and conditions and requirements within the space.
- **In Installed:** This Aspect is the recorded/documented properties of a manufactured, delivered or installed component.

6.2.4 Identifying the Purpose of the IMF Type

At the core of an IMF Type definition is its Purpose. The Purpose is a verb which describes the essence of the IMF Type. The Purpose is selected from the RDL. One valid Purpose which is widely used in examples in this document is 'Pumping'.

How to select a Purpose:

- **In Function:** consider what is the essential functionality *needed*, which this particular IMF Type shall define the necessary information content for and choose a Purpose which reflects this. Avoid considering a solution to the need, but instead keep the solution space open.
    - **Example:** Choose 'Driving' when some sort of mechanical energy needs to be provided for pumping, compressing, or other functions.
- **In Product:** consider what is the essential solution *provided*, which this particular IMF Type shall define the necessary information content for and choose a Purpose which reflects this.
    - **Example:** Choose 'Driving' for an electrical motor that provides mechanical rotational energy for e.g., a pump.
- **In Location:** the objective is to define spatial properties, including dimensions and relative location, furthermore, to define what conditions are within the space defined, and what

requirements apply. Choose 'Location' as Purpose if it relates to an area, room, or other space that can contain (locate) other entities.

- **Examples:** The spatial properties of an electric motor define its perimeter and relative position. If it is not intended to be a location for (contain) other entities, the Purpose is the same as in the Product aspect (Drive).
  The spatial properties of a cabinet define its perimeter and relative position. It is intended to contain (locate) entities, therefore the Purpose is 'Locate'.
  The spatial properties of a logical space such as 'Wifi coverage zone' are defined by the space where certain conditions are true, e.g., sufficient Wifi signal strength. The Purpose is 'Locate'.

- **In Installed:** consider what is the essential functionality *delivered*, which this particular IMF Type shall define the necessary information content for, and choose a Purpose which reflects this.
  - **Example:** Choose 'Driving' for an electrical motor.

6.2.5 Defining Purpose Attributes

The Purpose is a verb which describes the essence of the type, such as 'Separating', but alone it does not provide a sufficient specification, which is why Attributes are needed. These attributes are meant to further specify the Purpose, e.g., by specifying capacities and other parameters that answers, 'of what', 'in what way', 'how much', relating to the Purpose. Attributes are selected from the Reference Data library. Only one Attribute per 'Quality' is specified - attributes are incremented and datum typed after the IMF type is instantiated. E.g., 'Flowrate, volumetric' is a Quality which may later be Datum typed as 'Flowrate, volumetric'/Calculated/Operating/Maximum/Continuous, and it may be incremented several of same 'Quality' type.

6.2.6 Assigning the IMF Type to a Class

An IMF Type definition is a specialization of a more general IMF Type if it exists - which again is a specialization of a still more general IMF Type, and so on - i.e., there is a hierarchy, even if it is not formally defined in advance: By creating a new IMF Type on the basis of the more general IMF Type, effectively it becomes something similar to a sub-class of the more general IMF Type. A pattern will emerge as more and more specialized IMF Types are created from the more general ones by copying and extending. This pattern could provide a basis for later implementation of a formal class hierarchy. Assigning an IMF Type to a Class is therefore at present limited to stating if it is a specialization (child) of a more general (parent) IMF Type, or a modification (sibling) of some similar IMF Type.

IMF Types are selected from, and stored to the Reference Data library, governed by a 'submit for approval' work process.

6.2.7 Defining Input and Output Terminals

Terminals define what is inputting to- and outputting from-a Aspect Block or Aspect Interface Point. The main properties for a Terminal are direction (Input or Output) and media (e.g., process fluid, electrical current, information). Additionally, other attributes may allow specifying further details relating to the streaming Medium, such as volumetric flow rate, pressure, temperature.

Medium Object: When a higher level of precision is needed, a 'Medium Object' attribute can be specified and be used to refer to a separate information object which supplies details about the Medium composition and state at this Terminal. A Medium Template defines the structure of a Medium object.

A Medium Object specifically for Material is referred to as ‘Material Object’. It may contain a list of chemical components in the material or alternatively a reference to a defined block in a standard physical properties database such as DIPPR.

Similarly, Medium Object specifically for Energy is referred to as ‘Energy Object’. Such data objects may also be implemented for Force and Information Media, if required.

6.2.8 IMF Type in the Function Aspect

In this Aspect the *need* for an essential activity or function is specified, without knowledge or decision about how this need may be fulfilled.

Purpose Attributes are the means for further specifying the Purpose, e.g., by specifying capacities and other parameters that answers, ‘of what’, ‘in what way’, ‘how much’, relating to the Purpose

Example(s) of function purpose:

- The need for an essential activity or function is specified as the Purpose ‘Pumping’. The details of what must be met by a solution to fulfill this need are provided by Purpose Attributes such as ‘Flow, volumetric’, ‘Differential Pressure’, etc.

Supplementary Attributes are for specifying needs that are not directly related to the Purpose, and are the means for stating what must be met by a solution in order to be in compliance with applicable general- or overall requirements.

Example(s) of function attribute:

- Client requirements such as a specific Technical Specification which states that there shall be a 50% margin on all capacity attributes, calculated from the nominal capacity.

Terminals in the Function aspect specify the stream of a specific Medium inputting to- or outputting from the activity or function. Possible media include: Material, Energy, Force, and Information categories, and are selected from the Reference Data library.

Example(s) of terminal:

- An IMF Type for an Aspect Block with Purpose ‘Pumping’ may for instance have a terminal of type Input/Material/Fluid/Water.

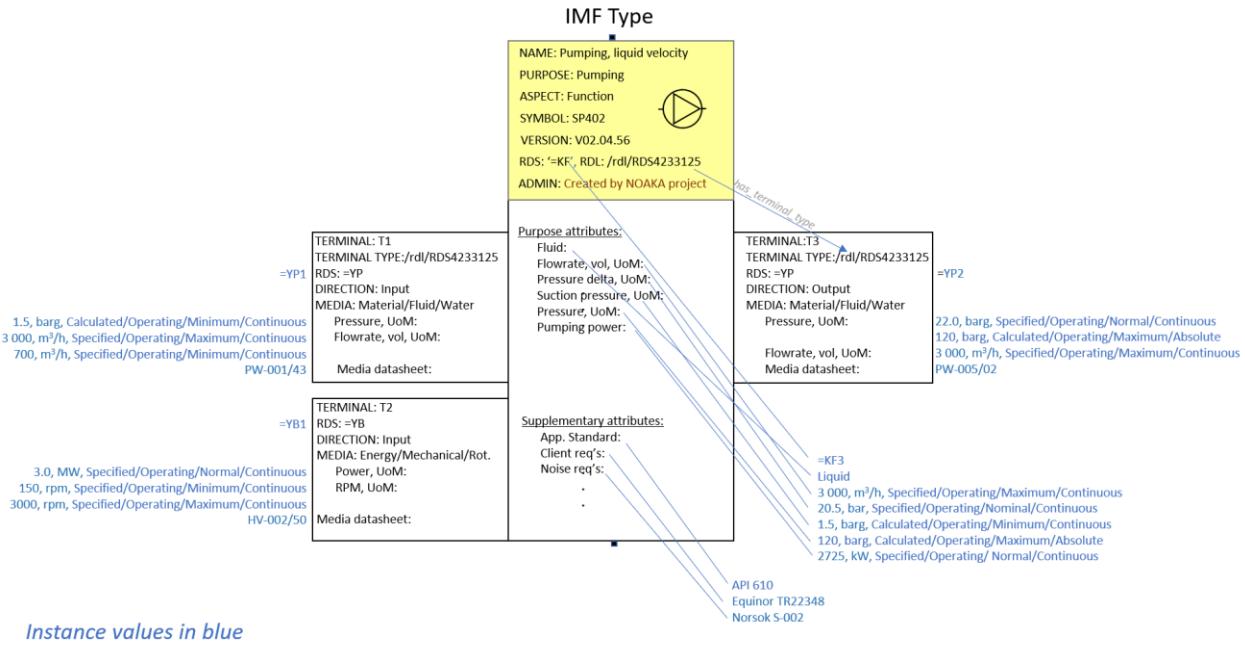


Figure 43: IMF Type example – Function aspect

6.2.9 IMF Type in the Product Aspect

In this Aspect it is s specified how to solve and fulfill a need by means of product or component. Artifact Attributes are the means for providing the details of how a solution is specified.

Example(s) of product purpose:

- The solution is specified as the Purpose 'Pumping'. The details of the solution are provided by Artifact Attributes such as 'Principle of operation', 'Pumping Power', etc..
  Supplementary Attributes are for specifying needs that are not directly related to the Purpose.

Example(s) of product attributes:

- Client requirements such as a specific Technical Specification which refers to a weight certification requirement.

Terminals in the Product aspect specify the *containment* of a specific Medium inputting to- or outputting from the activity or function, such as the pipe flange connections or cable termination. Possible media include: Material, Energy, Force, and Information, and are selected from the Reference Data library.

Example(s) of product terminal:

- An IMF Type for a Aspect Block with Purpose 'Pumping' may for instance have a terminal of type Input/Material/Fluid/Water and a terminal of type Output/Material/Fluid/Water.

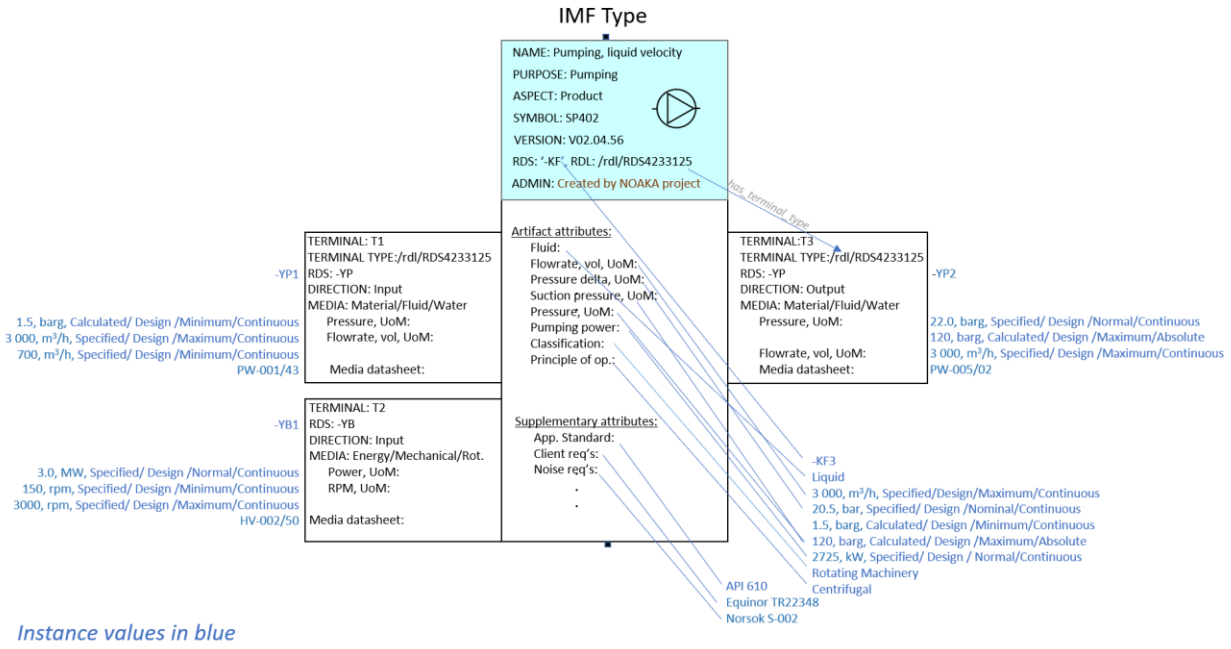


Figure 44: IMF Type example - Product aspect

6.2.10 IMF Type in the Location Aspect

In the Location Aspect the spatial properties are defined, including dimensions, relative location, and conditions and requirements within. Spatial Attributes are the means for holding this information.

Example(s) of spatial attributes:

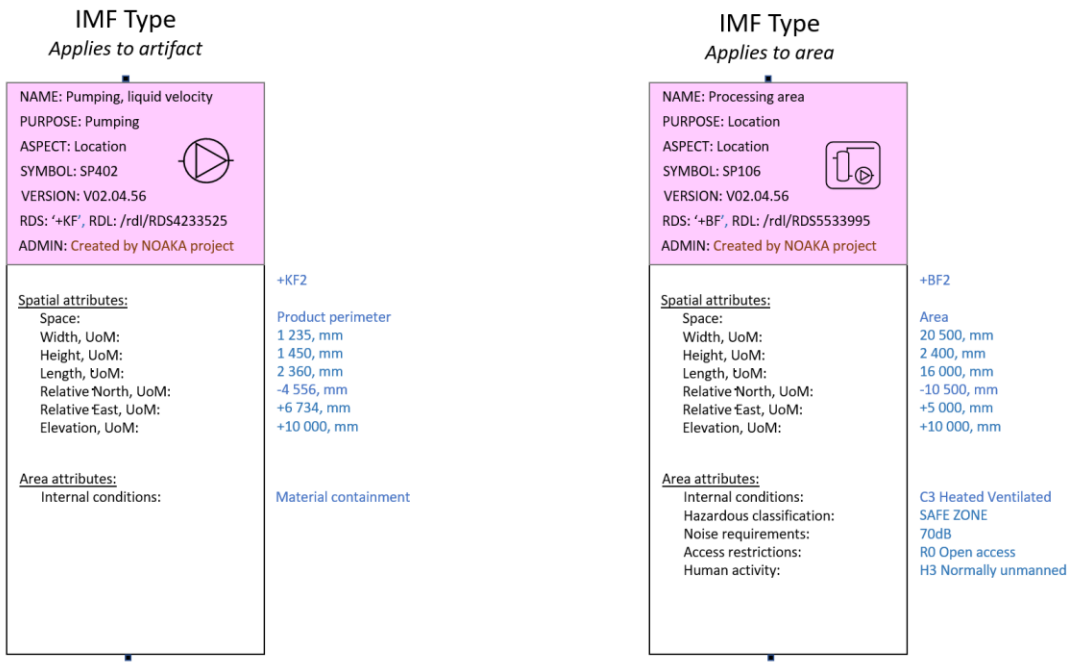
- A pump has a perimeter, given as Height, Width, Length attributes. It is located in some location, with relative position attributes X, Y, Z.
- A Processing area has size, given as Height, Width, Length attributes. As a location it is part of some other location, with relative position attributes X, Y, Z. Within the space of the area there are specific conditions given by Area Attributes, such as hazardous zone classification.

Area attributes are for specifying conditions and requirements that apply to any component located within this location (area).

Example(s) of area attributes:

- Noise requirements that will apply to the emission of noise from any component located in this location.
- Within the space of the pump (it's internals) there are specific conditions that may be specified.

In the Location aspect Terminals are not used.



Instance values in blue

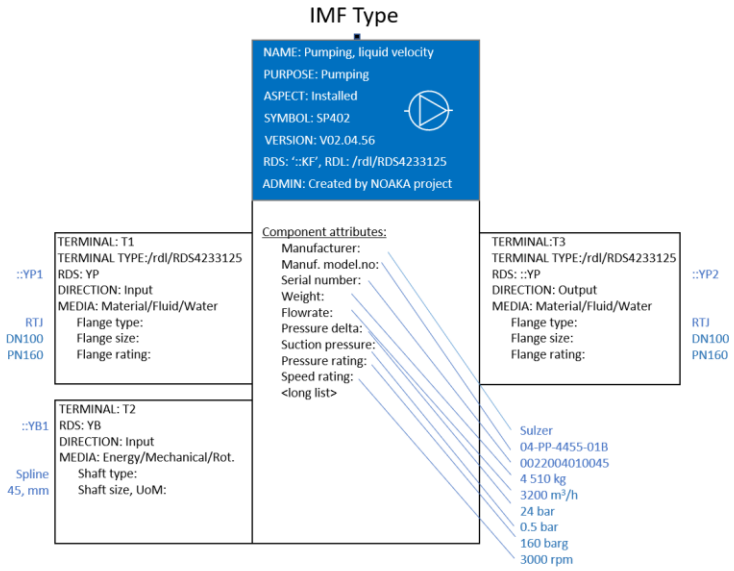
Figure 45: IMF Type example – Location aspect

6.2.11 IMF Type in the Installed Aspect

This Aspect is the manufactured, delivered or installed product or component, with its recorded/documented properties. The Component Attributes are the means for holding this information.

Example(s) of installed attributes:

- Manufacturers model number.
- Reference to a weight certification procedure that has been employed.
  In the Installed aspect the Terminals record/document the containment of inputs to- or outputs from the delivered or installed product.
- Pipe 'Flange size'.



Instance values in blue

Figure 46: IMF Type example – Installed aspect

6.3 Using Reference Data Libraries

IMF Models contain objects populated with attributes and relations according to the IMF types. Developing the IMF types requires a proper source of standardized definitions of the different concepts an IMF Model needs to utilize. Such sources are often referred to as Reference Data Libraries. Developing the IMF Types will mainly be based on the RDL provided by the POSC Caesar Association (PCA). The reference data is provided in the form of ontologies represented in W3C's Web Ontology Language OWL 2 and is compliant with ISO 15926-14 [6] upper level ontology.

The PCA RDL contains a comprehensive definition of types of objects, properties and relations related to industrial automation systems and life-cycle data for process plants including oil and gas production facilities. Some examples of such definitions are taxonomies of equipment, process activities or definitions of physical quantities and unit of measures.

Most of the definitions in the RDL has a reference to the same or similar definition stated in other industrial standards (where it is applicable) to ensure the possibility to provide different representation of an asset model according to definitions in several relevant standards.

IMF Types will utilize the concepts from the RDL and build any specific IMF Type in accordance with the corresponding concept in the RDL. It will ensure an interoperable IMF asset model provided in by the IMF language and can be automatically transformed into an RDF data set according the PCA RDL ontologies without losing any information.

7 The IMF Eco-System

IMF will be implemented in the context of applications that serve different parts of an eco-system. The applications in the IMF Eco-System should enable the engineers to create and share Information Models with minimal training and independent of companies and industries. An overview of the different components in the eco-system are shown in Figure 47. The main components are RDL, IMF Type Library, Authoring Tools, and Data Exchange Formats. In this chapter all references to applications in the IMF eco-system are made to the type of application, not to a specific instance of the applications unless otherwise stated.

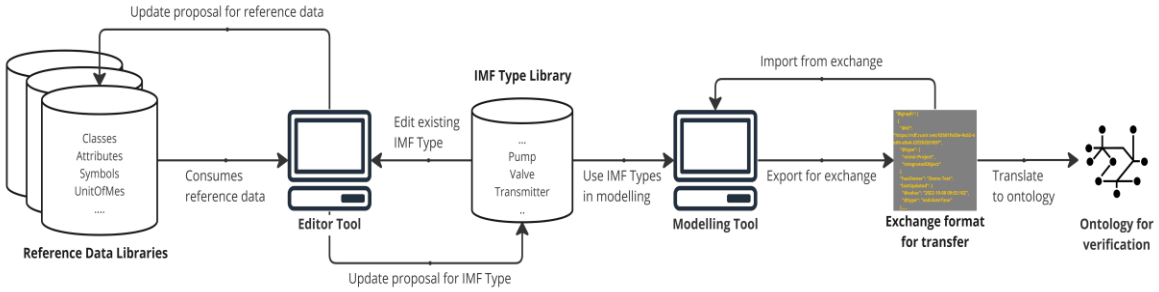


Figure 47: Overview of the IMF Eco-System.

7.1 Reference Data Library and IMF Type Library

7.1.1 Reference Data Library

RDLs are industry shared resources and thus enable standardization as well as minimizing duplication of work. IMF can make use of different RDLs and align them as part of the IMF Model integration process. IMF Models draw on a shared resource of definitions such as IMF Types and classes, property vocabularies, and units of measure. For example, when referring to an IMF Type in the IMF Model, the IMF Type must be from a list of allowed types, and the IMF Type itself must be defined. For this the shared resource is the master, and the reference data is available - for instance in a look-up fashion - when building the IMF Model.

The commercial benefits of deploying the IMF across industry, enabling shared resources and re-use are strong, but in some cases competitive advantage takes priority, with the need to protect Intellectual Property (IP). The IMF fully accommodates this need. Reference data, design codes, and model blocks can be proprietary, in which case they could be hosted by the owner, instead of hosted by an industry service (such as PCA).

7.1.2 IMF Type Library

IMF Type Library is a common industry resource for sharing IMF Types between IMF users. A common industry library of IMF Types will increase the standardization of IMF Types, reduce duplicate work, and make adaptation of IMF easier for new users.

The IMF Type Library shall offer support for:

- Submitting suggestions for new and updating existing IMF Types.

- Quality Assurance process for approval of suggestions for new and updating existing IMF Types. The QA workflow shall be transparent and offer relevant tools for an efficient and easy to use workflow.
- Versioning scheme for the entries in the library.
- API that allows for querying the library.
- Classification schema for the IMF Types.

IMF Type Library shall be community driven by allowing all IMF users to submit new IMF types and updating already existing IMF types in the library. This will allow users across the industry and value chain to collaborate to create high quality content. The approval process can be implemented as a peer-review where groups of SMEs can collaborate to ensure that updates to the library have the necessary quality.

The IMF Type Library is used to store and distribute the IMF Types created using the IMF Type Tool and as a source for IMF Types for the modelling. The API of the IMF Type Library shall offer support for the necessary functionality in the IMF Type Tool and the IMF Modelling Tool.

7.2 Authoring Tools

The Authoring Tools are the front-end tools used by the SMEs to create IMF Types and IMF Models. These tools can be implemented as standalone products or be integrated into already existing engineering tools. To offer seamless integration with the other components in the IMF eco-system the implementation of the authoring tools shall comply with the requirements outlined in the IT-specification that will be issued at a later stage.

7.2.1 IMF Type Editor Tool

The IMF Type Editor Tool is used to create new IMF Types and populate the IMF Type library that is used in the modelling.

The IMF Type Editor Tool shall offer support for:

- Search and browse for existing IMF Types in the IMF Type Library.
- Source attributes, classes, etc. from the Reference Data Library.
- Creation of new IMF Types (aspect blocks, terminals, interface points) using a graphical user interface.
- Updating existing IMF Types in the IMF Type Library.
- Workflow for submitting new or updating existing IMF Types in the IMF Type Library.
- Workflow for submitting suggestions for new reference data entries to the Reference Data Library.

To increase the standardization of the types created using an IMF Type Editor Tool, the Reference Data Library previously described shall be used for sourcing attributes, symbols, and classes for the creation of IMF Types. There should be a workflow that allows the SME to submit suggestions for new reference data entries to the RDL that are identified as missing during the IMF Type creation process.

Open-Source tool *:Tyle*

The IMF Type Editor Tool *:Tyle* has been developed to offer adopters of IMF a web-based open-source IMF Type Editor. *:Tyle* supports creation of aspect blocks, interface points and terminals using the POSC

Caesar Association RDL and IMF Type Library. *:Tyle* is developed as an open source project, where all users are invited to contribute to the development of the tool by submitting code, bug reports or feature requests. Source code, documentation and user tutorials can be found on Github.

7.2.2 IMF Modelling Tool

The IMF Modelling Tool is used by the SME to create the IMF Models and translate the created models into ontologies that can be shared across the value chain.

The IMF Modelling Tools shall offer support for:

- Searching and browsing for types in the IMF Type Library.
- Creation of models using a graphic interface (block diagram) that allows instantiation of aspect blocks, and configuration of all topological and hierarchical relations between aspect blocks, including inter-aspect relations.
- Modelling in all aspects in the core IMF (function, product, location, installed).
- Translate the models into ontologies according to the IMF specification.
- Export/Import IMF Models to/from the standardized IMF Model exchange format.
- Plugins that can perform specific calculations and queries.

IMF Modelling Tools shall source the IMF Types from the IMF Type Library. The sourced IMF Types are then instantiated in the modelling tool. Setting the values or references to values for the different attributes of the IMF Types shall be possible in the modelling tool.

The IMF Modelling Tool should provide a framework for adding plugins that can perform calculations and queries on a model. Use cases include checking that products that are related to functions fulfil the requirements stated by the function.

Open-Source Tool *Mímir*

The open-source IMF Modeling Tool *Mímir* has been developed to offer adopters of IMF a web-based open-source modelling tool. It supports creation of models, sourcing types from an IMF Type Library and compiling the models into ontologies. All users are welcome to contribute to the development of *Mímir* by submitting code, bug reports or feature requests. Source code, documentation and tutorials can be found on Github.

7.3 Serialization and Data Exchange

The IMF Eco-System is intended to be an industry wide collaboration that allows transfer of Information Models between the different parts of the value chain (operators, engineering contractors, suppliers). To enable seamless transfer of Information Models between the different actors, potentially using different applications, a standardized serialization and data exchange format is needed.

The data exchange format will be defined using widely accepted W3C standards. The IMF data exchange format is Resource Description Framework (RDF), using any of its widely supported standardized serialization formats. Several serialization formats for RDF exist, including JSON-LD, which is based on JSON, and RDF/XML, which is based on XML. The data exchange format vocabulary will be specified using the Web Ontology Language (OWL) and its grammar will be defined by the Shapes Constraint

Language (SHACL). The vocabulary and grammar specification of the data exchange format will encompass the semantics and constraints of the vocabulary and grammar specifications of the formal IMF language to the degree that OWL and SHACL languages support this and in a way that semantic and syntactic verification of the exchanged data may be practically applied.

Bibliography

[1] ISO, *15926 -1 - Industrial automation systems and integration - Integration of life-cycle data for process plants including oil and gas production facilities - Part 1: Overview and fundamental principles,* 2004.

[2] ISO/IEC, *81346-1: Industrial systems, installations and equipment and industrial products - Structuring principles and reference designations - Part 1: Basic rules,* 2022.

[3] ISO/IEC/IEEE, *15288: Systems and software engineering - System life cycle processes,* 2015.

[4] ISO, *TR 15926-14 - Industrial automation systems and integration - Integration of life-cycle data for process plants including oil and gas production facilities - Part 14: Data model adapted for OWL2 Direct Semantics.*

[5] IEC/ISO, "81346 - O&G - Reference Designation System for Oil and Gas," [Online]. Available: https://readi-jip.org/reference-designation-system-for-oil-and-gas/ .

[6] ISO, *14224 - Petroleum, petrochemical and natural gas industries - Collection and exchange of reliability and maintenance data for equipment,* 2016.

Appendix A - Attribute Collections for Aspect Objects

In the IMF language, the information is expressed in graphic schemes defined as in Table 16.

Table 16: Attribute collections for Aspect Objects.

Aspect: F	Aspect: P	Aspect: L	Aspect: I
Purpose: Pumping Purpose attributes: Supplementary: attributes Input media: Output media:	Artifact: Pump Artifact attributes: Supplementary: attributes Input media: Output media:	Location: Platform A Spatial attributes: Area attributes: Input media: Output media:	Artifact: Pump Component attributes: Input media: Output media:

Appendix B - Contractual Information Model Requirements for a Facility Asset

The Facility Asset information shall be provided in a digital format, conforming to the following requirements:

1. The Facility Asset information shall be provided in the form of an Information Model, or in the form of several partial Information Models if the partial models include the data required to integrate the models into one Information Model.
2. The Facility Asset Information Model shall be delivered in a digital structured data format.
3. The Facility Asset data shall be structured such that the data is separated into the different aspects of information about the systems of the Facility Asset.
4. For each aspect specific data of a system, the information shall include how the system is *partOf* a parent system to provide context, i.e., the hierarchy dimension of the model.
5. As a minimum, the Function, Product, and Location aspects shall be included.
6. The data about the relation between aspects of a system shall be included. As a minimum, the relations shall include how Product fulfils Function, and how Product has Location.
7. The system information element shall include data about its purpose.
8. The system information element shall include data about its inputs and outputs in the form of terminals, and the medium that they let in or out. As a minimum the media Material, Energy, Force, and Information shall be included.
9. The Facility Asset information shall include information elements with data about how the different outputs and inputs in the form of terminals are connected between different systems, i.e., the topology dimension of the model.
10. Descriptors and identifiers shall be included to: support interoperability with external systems, user interaction, and verification of reference data libraries.
11. The interoperability shall as a minimum include support for loss-less transfer of Facility Asset information between applications and support for interfacing with external registers/archives for exchange of attribute values that apply for each application.
12. The Information Model shall include version control mechanisms to support engineering workflows such as review and revision control to support contractual obligations.
13. Type definitions of systems used to create the Information Model shall be based on international/industrial standards and reference data libraries. Proprietary company and project types shall be avoided unless otherwise agreed.
14. The range of Type definitions needed to create instances in the Information Model is not complete. An outline shall be proposed of how to update and mature the reference data library such that it meets modelling needs.
15. Attributes allocated to any Type definition shall be classified according to the reference data library.

Appendix C - Structural Specification

The structural specification is a format and application independent data model specification of the formal specification of IMF. The structural specification aims to faithfully represent the intentions of the formal specification while also including the specification of how, e.g., objects metadata and attributes and attribute values are represented, which the formal specification does not contain. The purpose of the structural specification is to be a common language and format for developing concrete serialization formats for IMF and serve as documentation for application developers.

The specification consists of a series of partially overlapping diagrams. Figure 61 includes the complete structural specification.

Diagram legend

The diagrams are UML class diagrams are drawn using PlantUML restricted to the following elements:

- UML classes, marked with the icon "C".
- Abstract UML classes, marked with an icon "A", are UML classes that are not intended to be instantiated.
- Enumerations, marked with the icon "E", represent UML classes with a limited list of instantiations and where the instantiations are defined in the IMF language.
- Subclass relationships between classes, directed relations with an open arrow.
- Directed associations (relations) between classes, which are marked with a name and possibly a cardinality. If no cardinality is given, then the cardinality is 0–many.

Figure 48 shows an example of the constructs used.

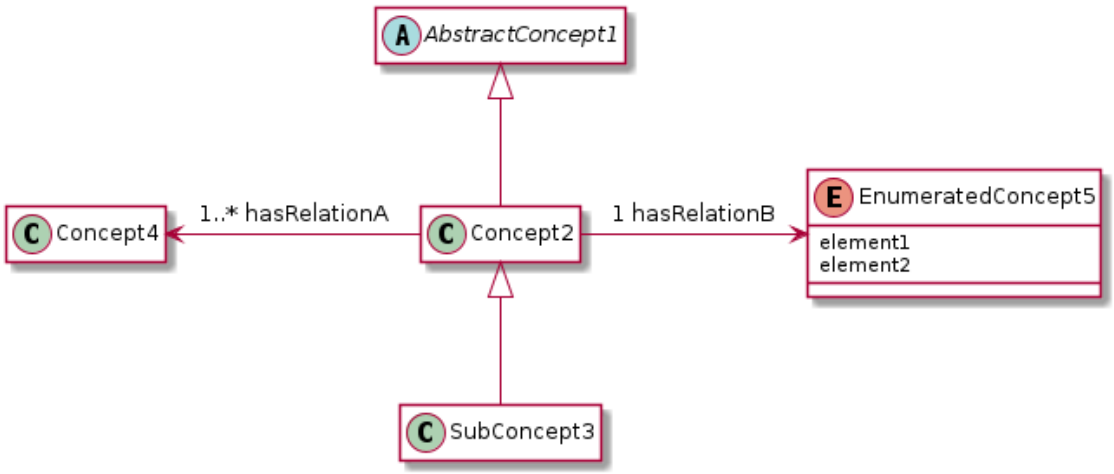


Figure 49: Example of UML class diagram constructs used in this appendix.

Note: The structural specification is described by a series of diagrams, each focusing on a specific part. In the case that a class is used in multiple diagrams, only one diagram contains the full specification of the class, while the other diagrams only contain the class name. The diagram that contains the complete specification combines all diagrams and contains all information.

Model

Figure 50 displays the structural specification of Entity, AnnotatedEntity, ModelElement and Model.

Entity is the most general construct in the structural specification. Everything is an entity. No fields or relationships are required for Entity, e.g., an Entity is not required to have an identifier.

An AnnotatedEntity is an Entity that has an identifier and additional metadata and provenance data as specified in the diagram. An Entity that is not an AnnotatedEntity may only exist through some relation to an AnnotatedEntity.

A Model is a collection of AnnotatedEntities and their dependent entities. These AnnotatedEntities may be called ModelElements. ModelElements are Entities that can occur in Models and are hence those AnnotatedEntities that may be exchanged as part of a Model.

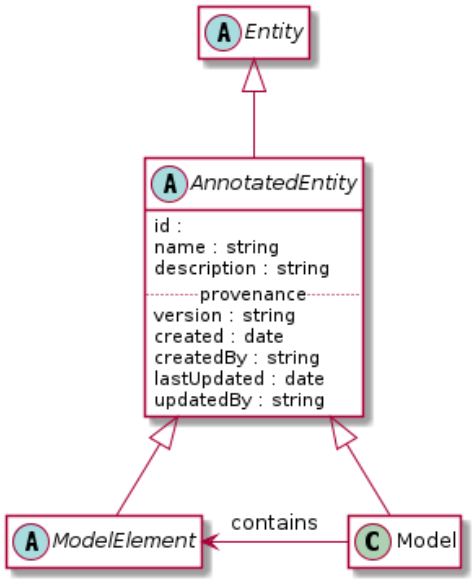


Figure 51: The structural specification of Entity, AnnotatedEntity, ModelElement and Model.

Note: most subclass relationships to Entity are not depicted in diagrams as this make them difficult to read.

System Elements

Figure 52 contains the structural specification of AspectElement and Aspect.

Every AspectElement has exactly one Aspect.

Every SystemElement is an AspectElement. A SystemElement is associated 0–many Datums. The structure of Datums is described below.

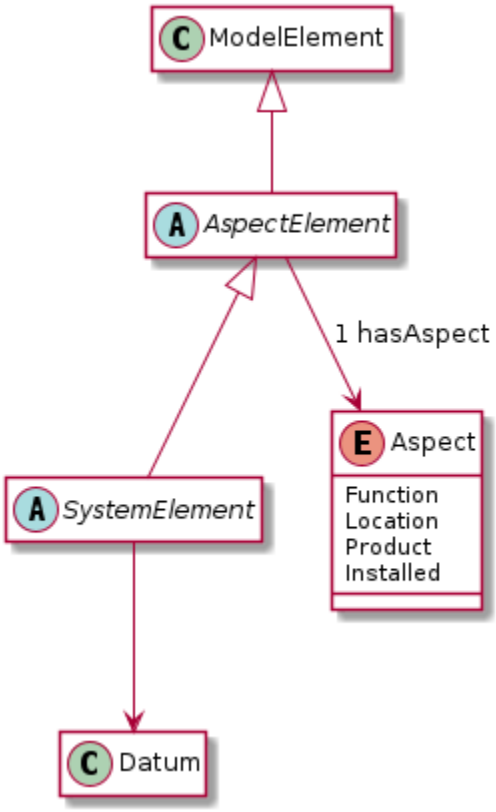


Figure 53: The structural specification of AspectElement and Aspect.

Attribute

Figure 54 contains the structural specification of Attribute.

An Attribute is a specification of Datums. An Attribute must specify a quality, and may specify a value with an associated unit of measure (uom). An Attribute may be classified at most one of each of the Attribute classifiers Provenance, Range, Regularity and Scope.

A Datum is an instantiation of an Attribute and must specify a value and a unit of measure as according to the Attribute specification.

Note: The specification of Attributes is currently basic. A more thorough analysis of requirements for expressing Attribute is planned for future releases.

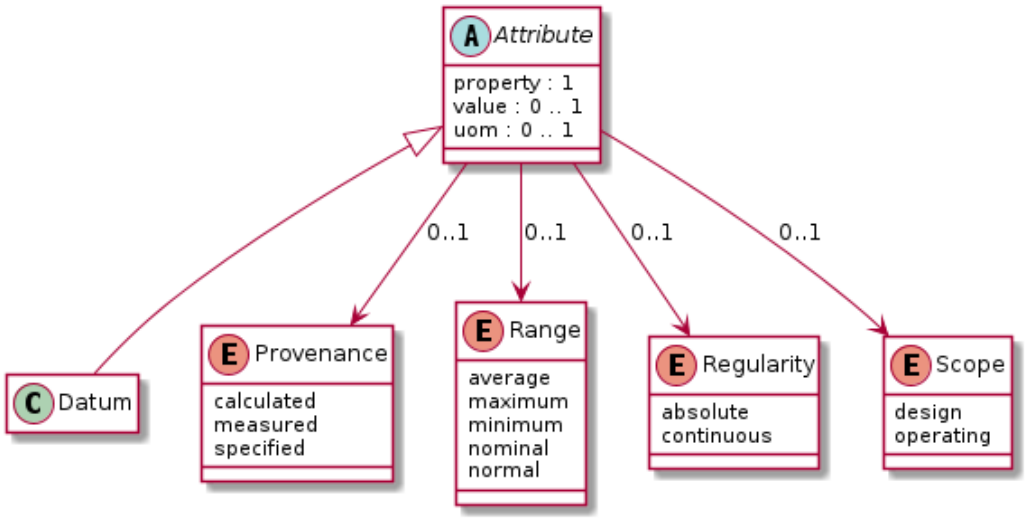


Figure 55: The structural specification of Attribute.

Types

Figure 56 contains the structural specification of Types.

A Type defines a blueprint from which SystemElements are created. There are two kinds of Types: BlockType and TerminalType, each representing blueprints for Blocks and Terminals, respectively. Types specify fields and relationships that must also hold for its instances. A BlockType may specify fields such as RDS string, purpose and symbol, and Terminals by relationships to TerminalTypes. A TerminalType specifies a direction and a Medium.

A Type may include multiple AttributeGroups. An AttributeGroup is a collection of AttributeTypes.

An AttributeType is an Attribute with additional constraints associated. These constraints may specify permissible values for the Attribute in different ways, e.g., by specifying a list of legal values, a range of values (in the case of a numerical values), legal datatype, regular expression, and so on. The expressivity of constraints for AttributeTypes will be defined upon a thorough analysis including both requirements from SMEs, application developers and the expressivity of suitable constraint languages.

An AttributeGroup is an AnnotatedEntity that is used to collect Attributes that naturally belong together to support convenience and ease of reuse. An AttributeGroup also serves the purpose of grouping together Attributes for presentational purposes, similarly as document sections in a data sheet do.

Instances may be created from types in at least two ways:

1. An instance is created from copying a type; hence the instance will explicitly contain the fields and relationships specified by its type.
2. An instance is created by referring to its type with a specified relationship that captures the semantics of the instantiation.

Both cases exhibit different issues with regards to synchronization and versioning. For case 1. an instance may evolve independently of its type, i.e., any updates to the instance will not affect the type

(and vice versa) unless specific measures to avoid this is specified. For case 2, updates to a type will also affect all its instances.

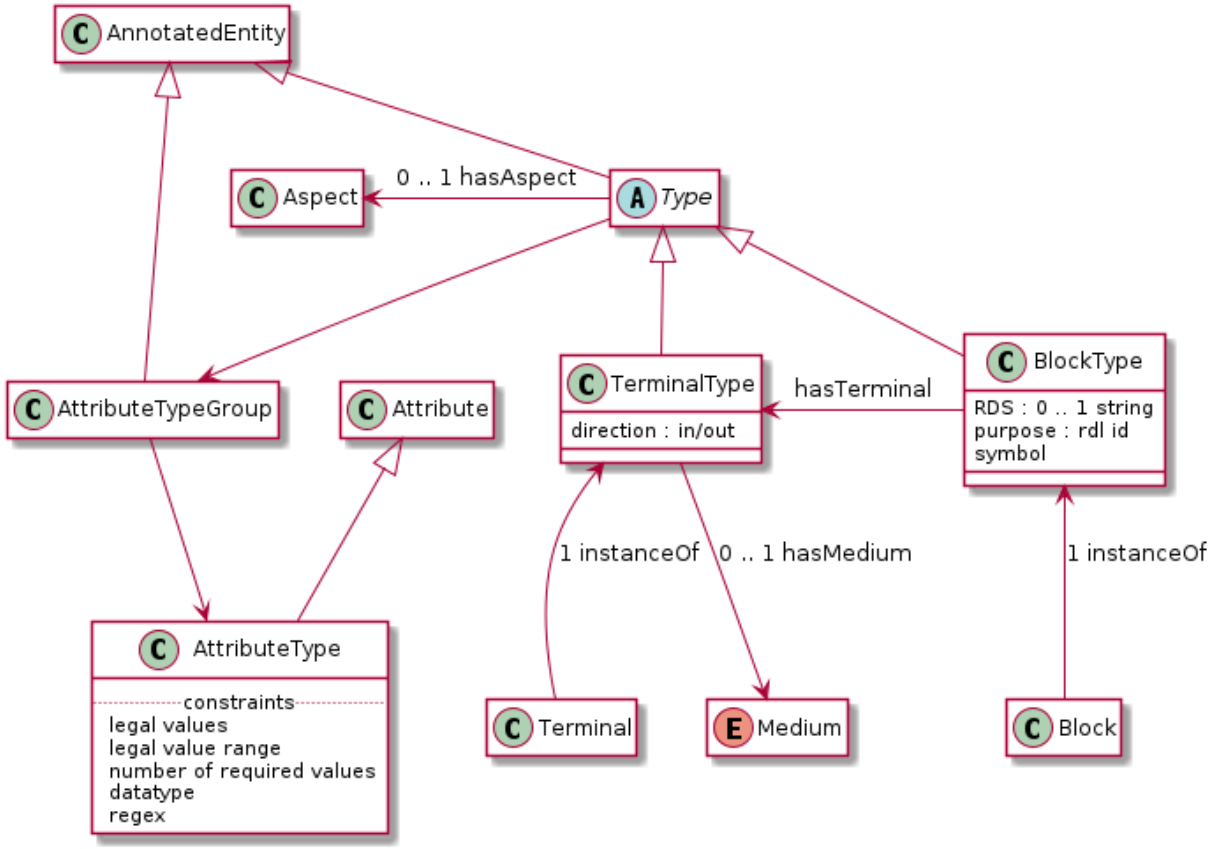


Figure 57: The structural specification of Type.

Instances

Figure 58 contains the structural specification of SystemElements.

The structural specification of the different SystemElements closely follow the formal specification.

The connectedTo relation between InTerminal and OutTerminal may either be a binary relationship between an InTerminal and an OutTerminal, or a ternary relationship between an InTerminal, an OutTerminal and an InterfacePoint. The latter form is represented in the diagram with a diamond.

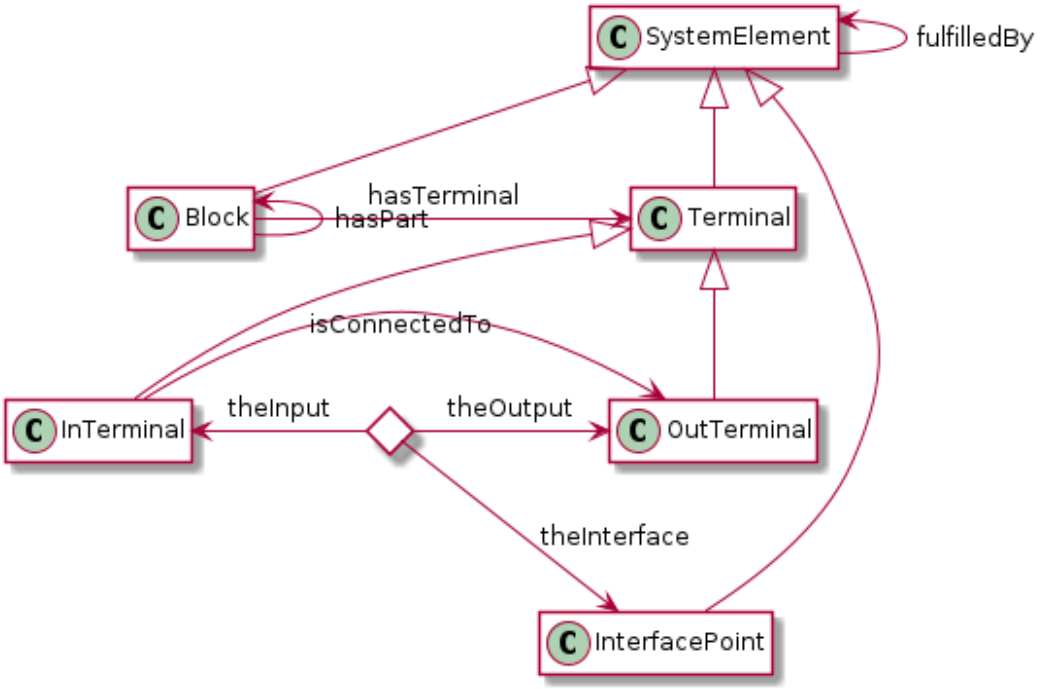


Figure 59: The structural specification of SystemElements.

Complete Specification

Figure 60 includes the complete structural specification.

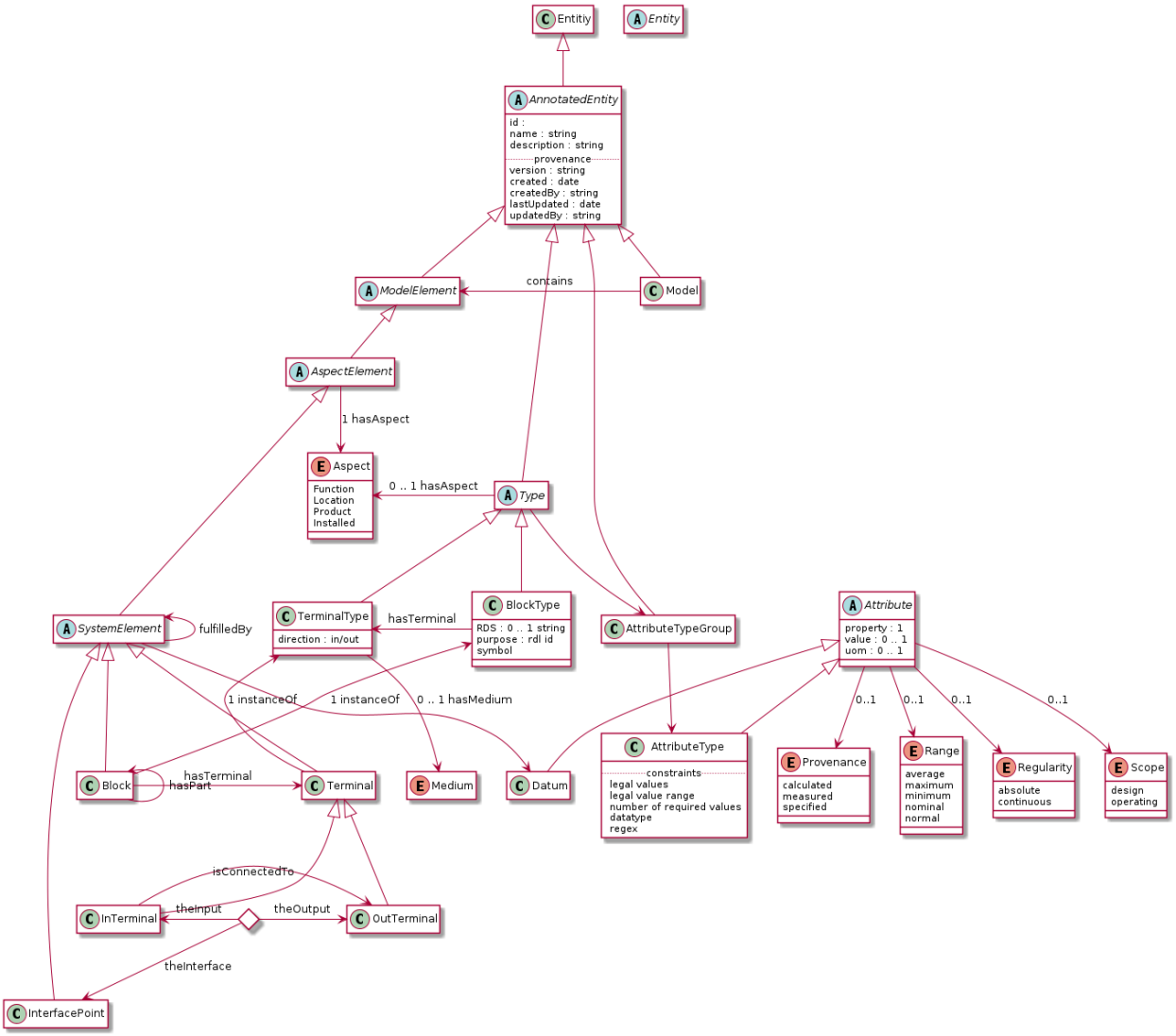


Figure 61: The complete structural specification.

Appendix D – IMF Semantic Web Resources

Semantic web resources, such as OWL ontologies and SHACL shape constraints, are published at http://ns.imfid.org.

Appendix E - Semantic Verification

IMF Types and Models are by design developed by reusing resources in an RDL. In order to enable semantic verification, we assume that the RDL classes are provided in an OWL format that supports efficient use of reasoning. ISO 15926-14 is a upper level ontology with exactly this ambition. We will in this chapter assume that the classes and attributes referred to in IMF Types and IMF Aspect Objects lie in an RDL that implements ISO 15926-14.

The primary aim of semantic verification is to detect inconsistencies in an IMF Model. When this is achieved and demonstrated, it is in the scope of IMF to support complex reasoning patterns based on inheritance of design decisions, as well as encoding of engineering rules. An example of a potential inconsistency in an IMF Model, that could be detected by a reasoning procedure, is conflicts between property restrictions in the definition of RDL classes and the combination of RDL classes and attribute constraints in the definition of an IMF Types. There are many ways in which the use of classes and attributes in IMF may clash with their definitions and position in an RDL class hierarchy, and it is not likely that it is possible to inspect and check all these cases by manual methods when an IMF model grows beyond a small toy example.

A natural step to this end would be to translate an IMF Model into an instance of an ISO 15926-14 ontology. There are two reasons why this is not straightforward. First, IMF types are defined with constraints that cannot be expressed in OWL. However, since validating an IMF Model with respect to such constraints can be achieved independently of semantic verification, this is not an issue. We may just assume that the IMF Model has been validated with respect to its constraints, e.g., by use of SHACL, prior to semantic verification and then we can safely ignore the constraints in the semantic verification. Second, IMF Models are used to capture requirements and specifications, and it is not clear how requirements and specifications can be expressed in ISO 15926-14.

The approach taken in this chapter is to translate an IMF Model into a set of axioms in first-order logic and use quantifiers in the representation of requirements. More precisely, first-order logic is used to describe a scenario, a term which corresponds to what a logician will call a possible world. Intuitively, a scenario is a Facility Asset in operation. Describing a scenario amounts to describing a situation in which the requirements and specifications in an IMF Model are met. Using concepts in ISO 15926-14, a scenario consists of artifacts, understood as physical objects or software, with functions realized in activities, and situated in spatial locations. Media states are understood as activities. Artifacts, functions, and location spaces are related through partOf and connectedTo relations. Note that there are no aspects in a scenario and no requirements, only explicit assertions of what we can view as statements of facts in the scenario.

The encoding in first-order logic of IMF aspect elements and relation connections into a scenario description is defined in two steps. First, a mapping from IMF aspect elements and relation connections is defined such that

- Elements in the Function aspect are mapped into functions,
- Elements in the Product aspect are mapped into artifacts,
- Elements in the Location aspect are mapped into location spaces,
- Relation connections are mapped into instances of relations over functions, artifacts, and location spaces.

Then, the scenario description is completed by adding verification conditions. The verification conditions, with associated axioms, are formal definitions of the meaning of the IMF language elements. The verification conditions are defined using quantification over activities, and it is this detail that is difficult to express in OWL. The task of semantic verification is to check consistency of a scenario description with respect to the axioms in the RDL.

The encoding of an IMF model into a scenario description is accessible to readers with elementary knowledge of first-order logic. The last part of the chapter addresses quantifier elimination and presupposes expert knowledge of first-order logic, the family of description logics that underlie OWL, and ISO 15926-14. This is the conclusion: A scenario description M can be transformed into an instance N of an ISO 15926-14 ontology such that M is consistent in first-order logic if and only if N is consistent in OWL. Hence, we can use OWL to check consistency of an IMF model with respect to the RDL. But M and N are not logically equivalent, hence we cannot replace the first-order logic representation of an IMF model with the OWL representation generated from it, without compromising semantic precision.

Mapping IMF Language Elements into a Scenario

Since the syntax used in this section is that of first-order logic, we will use predicates and not classes. A class subsumption axiom like

$$\text{Pumping} \sqsubseteq \text{Activity}$$

will then be written in the following form:

$$\forall x(\text{Pumping}(x) \rightarrow \text{Activity}(x)).$$

We will use a first-order language with equality, which allows representation of "there is exactly one such that". For instance, we can state that an artifact has a unique function by the formula

$$\forall a(\text{Artifact}(a) \rightarrow \exists! f\ \text{hasFunction}(a, f))$$

as an abbreviation for

$$\forall a\left(\text{Artifact}(a) \rightarrow \exists f\big(\ \text{hasFunction}(a, f) \wedge \forall g(\text{hasFunction}(a, g) \rightarrow f = g)\big)\right).$$

Predicates with associated axioms used to represent a scenario are listed in Table 17.

Table 17: Predicates and axioms for the representation of a scenario.

Class/Predicate	**Axiom**
Function	Pairwise disjoint, e.g., $\forall x(\text{Artifact}(x) \rightarrow \neg\text{Function}(x))$
Artifact	
LocationSpace	
Activity	
FunctionBlock	$\forall x(\text{FunctionBlock}(x) \rightarrow \text{Function}(x))$
FunctionTerminal	$\forall x(\text{FunctionTerminal}(x) \rightarrow \text{Function}(x))$
ArtifactBlock	$\forall x(\text{ArtifactBlock}(x) \rightarrow \text{Artifact}(x))$
ArtifactTerminal	$\forall x(\text{ArifactTerminal}(x) \rightarrow \text{Artifact}(x))$
ActivityBlock	$\forall x(\text{ActivityBlock}(x) \rightarrow \text{Activity}(x))$
MediaState	$\forall x(\text{MediaState}(x) \rightarrow \text{Activity}(x))$
InstalledBlock	$\forall x(\text{InstalledBlock}(x) \rightarrow \text{Artifact}(x))$

InstalledTerminal	$\forall x(\text{InstalledTerminal}(x) \rightarrow \text{Artifact}(x))$

The interpretation of aspect elements is defined via a mapping of an instance of an aspect element to an individual member of the classes in Table 17. An aspect element will also have attribute information. This information can be encoded in a predicate called a restriction. As a case in point, assume that

- Aspect element E is mapped to individual a
- E has purpose attribute set to 'Pumping' and just one purpose attribute: 'Volumetric flowrate' = 200 m^3/h.

The activity restriction defined on a, denoted [a.Activity](x), is given by the formula

$$\text{Pumping(x)} \wedge \text{VolumetricFlowrate(x)} = 200\ \text{m}^3\text{/h}$$

Other restrictions are defined in the same way and are denoted with the same notation Table 18 defines what aspect instances are mapped into, and what kind of restrictions that the attribute set of the aspect instances define.

Table 18: Mapping from aspect elements to a scenario and associated restrictions.

Instance of	Maps to instance of	Defines restriction on
FB	FunctionBlock	ActivityBlock
FT	FunctionTerminal	MediaState
PB	ArtifactBlock	ActivityBlock, ArtifactBlock
PT	ArtifactTerminal	MediaState, ArtifactTerminal
LB	LocationSpace	ArtifactBlock, LocationSpace
IB	InstalledBlock	ArtifactBlock
IT	InstalledTerminal	MediaState, ArtifactTerminal

The mapping of relation connections in an IMF Model is specified in Table 19. As an example of how the table should be read, assume that A and B are two function blocks and that the IMF model states partOf(A,B). Assume that A is mapped to a and B is mapped to b. Then partOf(A,B) is mapped to $\text{partOfFunction}(a, b)$.

Table 19: Mapping of relation connections between aspect elements into instances of scenario relations.

Dom → Range	IMF relation connection	Maps to relation instance of
F→ F	`partOf`	partOfFunction
P→P	`partOf`	partOfAssembly
L→L	`partOf`	partOfLocation
I→I	`partOf`	partOfInstalled
F→ F	`hasTerminal`	hasInFunctionTerminal hasOutFunctionTerminal
P→P	`hasTerminal`	hasInAssemblyTerminal hasOutAssemblyTerminal
I→I	`hasTerminal`	hasInInstalledTerminal hasOutInstalledTerminal

F→ F	connectedTo	connectedToFunction
P→P	connectedTo	connectedToArtifact
I→I	connectedTo	connectedToInstalled
F→P	fulfilledBy	inverse(hasFunction)
L→P	fulfilledBy	inverse(hasLocationSpace)
P→I	fulfilledBy	installedAs

Verification conditions

Verification conditions for scenario predicates are defined Table 20. The conditions make repeated use of the two predicates hasFunction and realizedIn. Domain and range for the predicates are given by:

$$\forall a, f(\text{hasFunction}(a, f) \rightarrow (\text{ArtifactBlock}(a) \wedge \text{FunctionBlock(f)}) \vee (\text{ArtifactTerminal}(a) \wedge \text{FunctionTerminal(f)}))$$

$$\forall f, a(\text{realizedIn}(f, a) \rightarrow (\text{FunctionBlock}(f) \wedge \text{ActivityBlock(a)}) \vee (\text{FunctionTerminal}(f) \wedge \text{MediaState(a)}))$$

Table 20: Verification conditions for scenario predicates.

Individual *b* mapped from aspect block s.t.	**Verification condition**
FunctionBlock(*b*)	$\forall a(\text{realizedIn}(b, \text{a}) \rightarrow [\text{b. ActivityBlock}](\text{a}))$
ArtifactBlock(*b*)	$\forall f, a(\text{hasFunction}(b, f) \wedge \text{realizedIn}(f, \text{a}) \rightarrow [\text{b. ActivityBlock}](\text{a})) \wedge [\text{b. ArtifactBlock}](b) \wedge \exists! f \exists! a (\text{hasFunction}(b, f) \wedge \text{realizedIn}(f, \text{a}))$
LocationSpace(*b*)	$\forall a(\text{hasLocationSpace(a, b)} \rightarrow [\text{b. ArtifactBlock}](a)) \wedge [\text{b. LocationSpace}](b)$
InstalledBlock(*b*)	$[\text{b. ArtifactBlock}](b)$
FunctionTerminal(*b*)	$\forall s(\text{realizedIn}(b, s) \rightarrow [\text{b. MediaState}](s))$
ArtifactTerminal(*b*)	$\forall f, s(\text{hasFunction}(b, f) \wedge \text{realizedIn}(f, s) \rightarrow [\text{b. MediaState}](s)) \wedge [\text{b. ArtifactTerminal}](b) \wedge \exists! f \exists! a (\text{hasFunction}(b, f) \wedge \text{realizedIn}(f, \text{a}))$
InstalledTerminal(*b*)	$[\text{b. ArtifactTerminal}](b)$

Verification conditions for scenario relations are given in Table 21. The partOf relations satisfy the following axioms, where partOf is any of the four partOf relations:

$$\forall x, y, z(\text{ partOf}(z, x) \wedge \text{partOf}(z, y) \rightarrow x = y)$$

$$\neg \exists x \ \text{partOf}^{+}(x, x)$$

Table 21: Verification conditions for scenario relations.

Relation instance	**Verification condition**
partOfFunction(*a,b*)	$(\exists x \ \text{realizedIn}(a, x) \rightarrow \exists! \text{ y realizedIn}(b, y)) \wedge \forall x, y(\text{ realizedIn}(a, x) \wedge \text{realizedIn}(b, y) \rightarrow$ partOfActivity(x,y))
partOfAssembly(*a,b*)	-

partOfLocation(*a,b*)	-
hasInFunctionTerminal(*a,b*)	$\forall x, y(\text{realizedIn}(a, x) \wedge \text{realizedIn}(b, y) \rightarrow \text{isInputTo}(\text{y}, \text{x}))$
hasOutFunctionTerminal(*a,b*)	$\forall x, y(\text{realizedIn}(a, x) \wedge \text{realizedIn}(b, y) \rightarrow \text{isOutputFrom}(\text{y}, \text{x}))$
hasInAssemblyTerminal(*a,b*)	$\forall f, g, x, y(\text{hasFunction}(a, f) \wedge \text{hasFunction}(b, \text{g}) \wedge \text{realizedIn}(f, x) \wedge \text{realizedIn}(g, y) \rightarrow \text{isInputTo}(\text{y}, \text{x}))$
hasOutAssemblyTerminal(*a,b*)	$\forall f, g, x, y(\text{hasFunction}(a, f) \wedge \text{hasFunction}(b, \text{g}) \wedge \text{realizedIn}(f, x) \wedge \text{realizedIn}(g, y) \rightarrow \text{isOutputFrom}(\text{y}, \text{x}))$
hasInInstalledTerminal(*a,b*) hasOutInstalledTerminal(*a,b*)	- -
connectedToFunction(*a,b*)	$\forall x, y(\text{realizedIn}(a, x) \wedge \text{realizedIn}(b, y) \rightarrow x = y)$
connectedToArtifact(*a,b*)	$\forall x, y(\text{hasFunction}(a, x) \wedge \text{hasFunction}(b, y) \rightarrow x = y)$
connectedToInstalled(*a,b*)	-
hasFunction(*a,b*)	-
hasLocation(*a,b*)	-
installedAs(*a,b*)	-

Quantifier elimination

Quantifier elimination is based on repeated application of the following observation. Let Γ be a set of formulas such $\Gamma \nvdash \exists x P(x)$. Then

$$\Gamma, \exists!\, x\, P(x), \forall x(P(x) \rightarrow Q(x))$$

is consistent if and only if

$$\Gamma, P(a), Q(a)$$

is consistent, provided a is an individual that does not occur in $\Gamma, P(x), Q(x)$.

Proof: Assume $\Gamma, P(a), Q(a)$ is satisfiable. Since $\Gamma \nvdash \exists x P(x)$ there is a model $\mathcal{M}$ of Γ where $\exists x P(x)$ is false. Since a does not occur in $\Gamma, P(x), Q(x)$, pick a value for a that makes $P(a), Q(a)$ both true. Then $\Gamma, \exists!\, x\, P(x), \forall x(P(x) \rightarrow Q(x))$ is true in $\mathcal{M}$. Conclude by completeness. The other direction is trivial.

Verification conditions of the form $\forall x\big(P(x) \rightarrow Q(x)\big)$ without a condition of the form $\exists!\, x\, P(x)$ are generated in mappings from aspect elements of the function aspect to a function individual. When a function aspect element is related to a product aspect element via fulfilledBy, a formula of the form $\exists!\, x\, P(x)$ is added to the verification condition of the function individual, and the observation above applies. We can then select a fresh individual (this will be an activity individual) and substitute this for the variables, as stated in the observation. This will reduce the quantifiers in the verification condition. The situation will progress upwards in the partOfFunction hierarchy and enable us to eliminate quantifiers until we reach the leaf node in the partOfFunction hierarchy. If there are still universally quantified variables left, there must be function individuals that are in the bottom of the partOfFunction hierarchy and that are not the function of any artifact individual. Then add a new fresh activity individual and repeat the procedure sketched above. Finally, there may be existentially quantifiers left of the form

$\exists!\, f\ \mathrm{hasFunction}(b, f)$. Remove the quantifier and substitute a fresh individual for the free variable that results (in this case f).

Given a set Γ of formulas generated from a mapping of an IMF model into a scenario, this results in a quantifier free set of formulas Δ such that Γ is consistent in first-order logic if and only if Δ is consistent.

The set Δ contains a number of closed equality formulas of the form $a = b$. This is inefficient for reasoning purposes. It is simple to substitute one for the other and remove identity statements, e.g., substitute a for b and remove resulting occurrences of $a = a$. This results in a quantifier free and equality free set of formulas Λ such that Γ is consistent if and only if Λ is consistent.

The set of formulas Λ can be represented in DL in the language of ISO 15926-14. Its consistency with respect to a reference data library compliant with ISO 15926-14 can hence be checked automatically with the help of OWL reasoning services.

Example

Assume that an IMF model contains

- A function block E mapped to f
- A product block F mapped to b

The verification conditions give

- $\forall x\big(\mathrm{realizedIn}(f, x) \rightarrow [\mathrm{f.ActivityBlock}](\mathrm{x})\big)$
- $\forall g, a\big(\mathrm{hasFunction}(b, g) \wedge\ \mathrm{realizedIn}(g, \mathrm{a}) \rightarrow [\mathrm{b.ActivityBlock}](\mathrm{a})\big)$
- $[\mathrm{b.ArtifactBlock}](b)$
- $\exists!\, g \exists!\, a\ \big(\mathrm{hasFunction}(b, g) \wedge\ \mathrm{realizedIn}(g, \mathrm{a})\big)$

If we add fulfilledBy(E,F), we will add hasFunction(b,f) to the scenario model as illustrated in the figure below to the left (x is a universal variable and g, a are existentially quantified variables). In this case we can eliminate the quantifiers, substitute individuals f and a' for the variables, and get the situation in the figure below to the right.