Information Modelling Framework Manual

Version 2.1

2023-07-05

Document Description

Title: The Information Modelling Framework Manual

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

Classification: Open

Owner: Magnus Knædal

Status: Revision 2.1.1. Ready for beginning implementation and development of DNV RP.

Authors: 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 version 2.1.1 of The Information Modelling Framework Manual. The development of the manual will continue after this release.

The document was updated 2023-10-27 with the isolated change of not calling the document a "draft version", but rather just "version". The reason for using the word "draft" was to indicate that IMF is in development and changes must be expected. The core concepts of IMF in this version can be considered stable, experimental concepts are clearly marked.

Valid from: 2022-11-21

Updated: 2023-07-05 (Minor update 2023-10-27, see Remarks above)

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 projects, and then was extended to include the partners of the NOAKA Digital Cooperation, Equinor and Aker BP. The work has from the beginning been conducted in collaboration with the SIRIUS Centre at the University of Oslo.

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. The learnings from this piloting work 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 results shall form the necessary basis to create a DNV Recommended Practice on how to implement IMF in the industry. Responding to an increasing interest and involvement from industry, a long term objective is make the results a basis for an international standard.

[1] https://readi-jip.org/

Introduction

The Information Modeling Framework (IMF) establishes a new way of working using Information Models through the capture and logical flow of information during an industrial design and construct project through to operations, maintenance and disposal.

The IMF is a method, a framework, and a language that allows creating an engineering friendly formal description of a Facility Asset, using graphical figures and 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.

The overall requirements to functionality of the Facility Asset are captured in an initial Information Model. 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. Due to the inherent features of the (machine-readable) format proposed in IMF, the Information Models can be continuously checked for design flaws by an automated work process. Furthermore, the fragments of information, now represented through several IMF aligned 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 handover to operator takes place the Information Model, a model-of-models, contains records of the historical context and design decisions. This 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. Furthermore, access to information is not restricted by documents and formatting, but can be navigated freely and maintained efficiently.

The goal of IMF is to enable more reliable and efficient 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 in the Capital Value Process (CVP), along the supply chain, and 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.

Outline and Readers Guide of this Document

This document introduces 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.

Chapter 1 states the scope of this document.

Chapter 2 contains a brief description of the problem that IMF is developed to solve and a vision for how IMF can be implemented, including an overview of the foreseen eco-system of applications. This chapter situates IMF in the context of its intended use.

Chapter 3 introduces fundamental concepts underlying IMF. This chapter is necessary for understanding IMF in depth, but it is not required for a first reading.

Chapter 4 is an easy to read overview of the IMF language that can serve as reference when reading the example material.

Chapter 5 is a guide for how to use IMF to model Facility Assets.

Chapter 6 is a guide for how to create the building blocks for IMF Models.

Chapter 7 formalizes the IMF Language and gives an overview of the different elements, relations, and rules in IMF. This chapter is written for readers with prior knowledge of semantic technology.

For intended users of IMF the entire document excluding Chapter 7 and 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 Chapter 1, Chapter 2 and Chapter 3 should be sufficient.

Contents

List of Figures

List of Tables

List of Terms

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 [2], 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 [3].

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 [5].

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.

List of Acronyms

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.

Chapter 1

Scope

This document contains a specification of the Information Modelling Framework, including fundamental concepts and guidelines for use, as well as its intended working relations to external resources and applications. Referring to Figure 1.1 the external resources and applications are:

❶ Engineering Tools and Registers that 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, integrity checking, and verification of the IMF model.

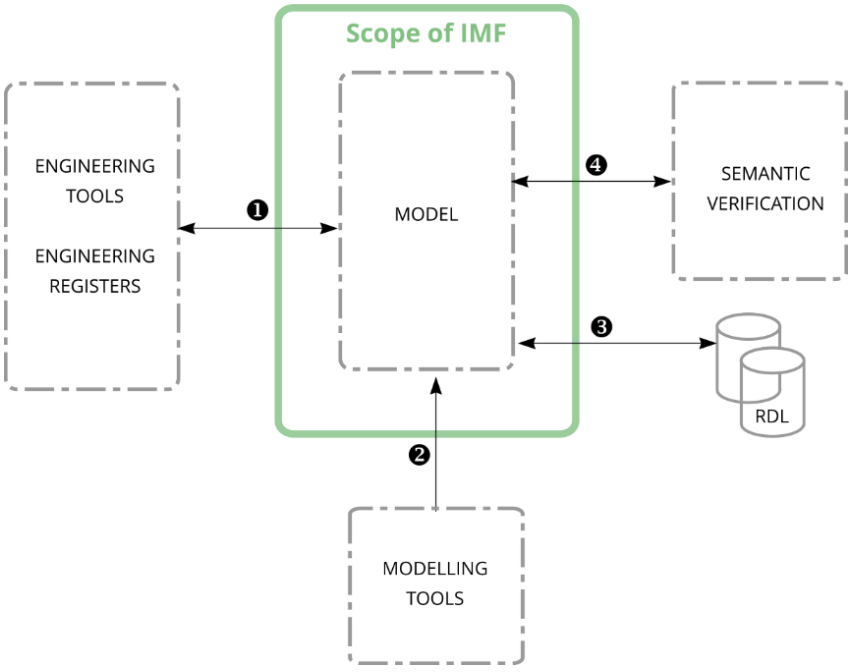


Figure 1.1: Framing and scoping of the IMF.

The following are within the scope of this Manual:

- 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
- Examples of engineering Tools and Engineering Registers that may hold attributes and objects and describe the Facility Asset.
- Identification of Modelling Tools for creating, modifying, and sharing IMF models.
- Explanation of use of Reference Data Libraries that contain the resources from which IMF models are built.
- Semantic verification mechanisms and tools that offer computer-based validation and verification of the IMF model.
- Guidelines that enable Subject Matter Experts (SMEs) to be users of the framework, supporting and improving existing discipline design workflows.
- Processes and artefacts that are scalable across disciplines, work processes, and the value chain, supporting any level of complexity and level of detail, and depending on the need and where in the process the SMEs are.
- Examples of shared libraries and use of existing standards.
- Open protocols for exchange of Facility Asset information and provide a format which enables automated verification and augmented engineering extensions.

This document and the development of the Information Modelling Framework is intended to serve as a reference for:

- Understanding how to implement a new way of working using Information Models.
- Providing the basis for 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 [3] (structuring principles) and 15926-14 [6] (reference data and verification).
- Computer-based exchange, automatization, and verification of Information Models.
- Contractual requirements for delivery of Facility Asset Information Models. See Appendix A.

Chapter 2

A New Way of Working using Information Models

2.1 Problem Statement

Facility Assets in the energy industry are increasingly complex. 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 are loss of information, risk of safety and quality breaches, duplication of work, 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 2.1 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.

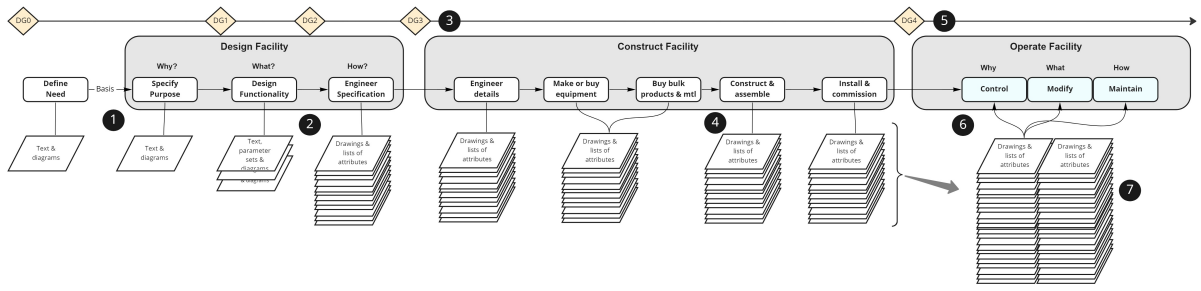


Figure 2.1: Current documentation practice during an industrial investment and development project.

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 specialized discipline expertise is necessary (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 increasingly difficult to maintain a holistic description of the Facility Asset. The result is extensive interface coordination between discipline experts ❷.

When the investment decision is made to execute the construction of the Facility Asset (DG3) ❸, the number of documents produced 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 quality deviations and HSE incidents.

When the operation (DG4) and handover to the Operator takes place ❺, information is fragmented and lack relational information. This often results in a need to 're-engineer' the solution to establish the holistic view necessary to maintain, control and evolve the facility asset ❻.

It is worth noting that reduced information quality can reduce the decision quality and is often costly and inefficient to manage.

2.2 Requirements to Solution

The objective of the Information Modelling Framework (IMF) is to enable a transition from the current documentation practice discussed in Section 2.1, to the use of *Information Models*, as a way of capturing and expressing information about Facility Assets.

To achieve this transition, the solution must facilitate for:

- 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.
- Scalability across disciplines, work processes, and the value chain. The method needs to be granular enough to support any level of complexity and detail, depending on the need and where in the process the SMEs are.
- Ease of use. Be made intuitive to the extent that the SMEs themselves are users of the framework. Thus, the method must support and allow existing discipline design workflows.
- Utilization and promotion of shared libraries and existing standards.
- Open protocols for exchange of Facility Asset information and provide a format which enables automated verification and augmented engineering extensions.

2.3 Using Information Models

To solve the problems discussed in Section 2.1, 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 2.2 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.

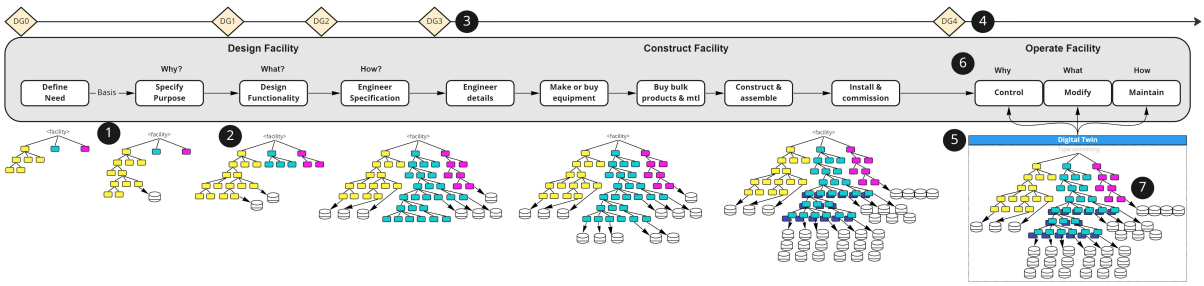


Figure 2.2: Using Information Models instead of documents during an industrial investment and development project.

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 describing 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-model*, 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 granular level 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.

2.4 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 to hold 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 2.3 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.

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[1] 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 [6] 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 complete puzzle ❺. IMF Models can be translated ❻ to enable the Knowledge Engineers to exploit verification techniques such as automated reasoning ❼.

[1] https://dexpi.org/specifications/

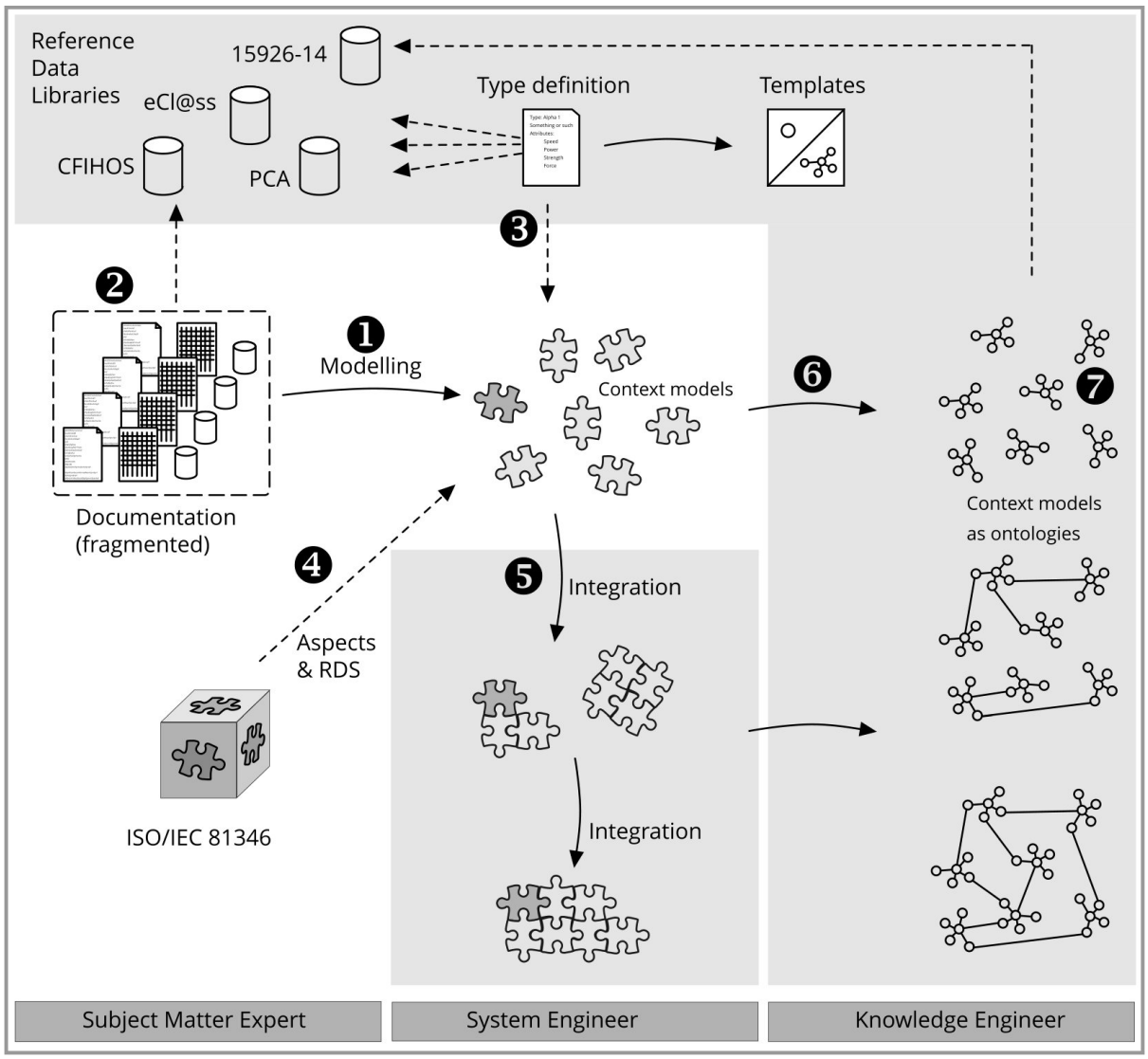


Figure 2.3: Different expertise and how they enable Information Modelling of Facility Assets.

2.5 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 2.4. The main components are RDL, IMF Type Library, Authoring Tools, and Data Exchange Formats. In this section 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.

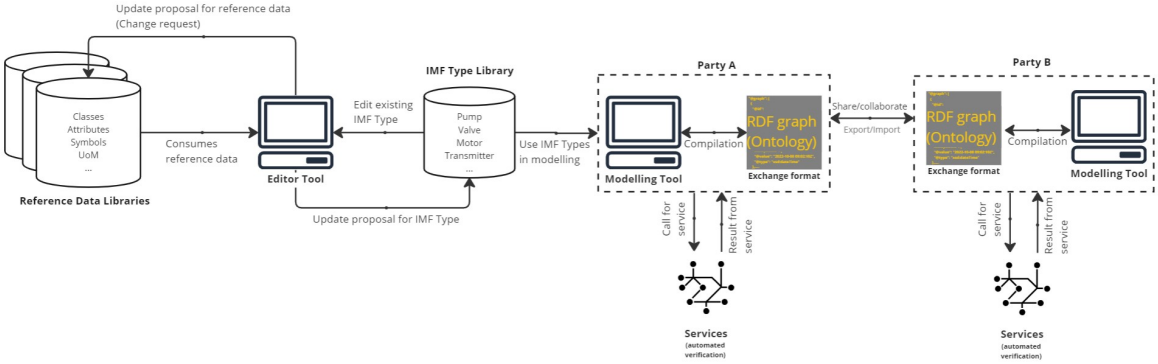


Figure 2.4: Overview of the IMF Eco-System.

2.5.1 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).

2.5.2 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.

2.5.3 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.

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.

2.5.4 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.

2.6 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.

Chapter 3

Fundamental Concepts for IMF

This chapter introduces fundamental concepts from systems engineering as a foundation for the IMF language. The aim of Section Section 3.1 is to provide a shared understanding of the system concept. Section 3.1 to Section 3.3 are intended for readers with engineering background. Section 3.4 is intended for readers who also want to obtain a basic understanding of the formal background of IMF.

3.1 System

3.1.1 Definition

The concept of system lies at the core of IMF. We will use the following definition. *A system is a group of connected elements that interact through media to provide one or several functions, where:*

- *A function is understood as a capability to realise an activity;*
- *Like the system itself, also its elements, called system elements, provide functions; the system elements functions are called sub-functions of the system;*
- *A medium is understood to be either material, energy, force, or signal carrying information;*
- *Both the system and its system elements have terminals;*
- *Two system elements are connected via connecting a terminal of one system to a terminal of the other; the point at which the terminals connect is called a connection point;*
- *The set of connection points connecting terminals of one system to terminals of another is called the interface between the two systems;*
- *System elements of a system connect to elements outside the system boundary in the same way as they connect to system elements within the system boundary*.

3.1.2 Engineered System

An engineered system is designed with a purpose; the purpose is served by one of the system functions called the main function. Engineered systems whose main function is to transform a state of media at input terminals to a state of media at output terminals are of particular interest to IMF.

The definition in Section 3.1 is closely reflecting the definition in the Systems Engineering standard ISO/IEC/IEEE 15288, where a system is defined as a combination of interacting elements organized to achieve one or more stated purpose [5]. Figure 3.1 graphically illustrates the definition in ISO/IEC/IEEE 15288.

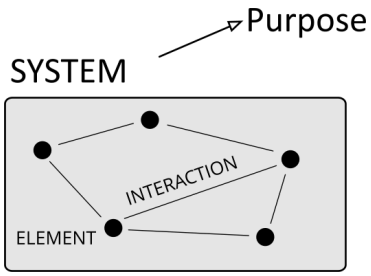


Figure 3.1: The system concept from ISO/IEC/IEEE 15288, illustrated as a group of elements interacting within a boundary to achieve a purpose.

3.1.3 Recursive Pattern

The concept of systems comprises a recursive pattern:

- The system elements of a system can themselves be systems; this results in a system breakdown
- System elements can connect to elements outside the system boundary, and these elements can themselves be systems and be elements of systems; this results in a topology
- These system breakdown and topology principles can be combined in a recursive matter

The points are substantiated in the following using a cooling system as an example. The elements of a cooling system may be artefacts such as pumps, pipes, chillers, etc. The purpose of the cooling system is to deliver heat exchanging. The main function of the cooling system is thus to provide a heat exchange function.

3.1.4 Breakdown

The system *breakdown* describes how a system is drilled down to system elements recursively. A complex system element of a system may be considered as a system with its own system elements (illustrated in Figure 3.2). This view of a system can be applied recursively, with system elements being drilled down into sub-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*.

For example, 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).

The desired granularity of breakdown might vary between the different parties involved in the engineering of a facility asset. For instance, the engineering contractor may take the pump as an artefact that serves as a system element, while the pump supplier needs to view the pump as a system and perform a further system breakdown. Note that since the system elements of a system comprise a group, a breakdown is also a way of grouping. The notion of granularity will be extended in future versions of the IMF manual.

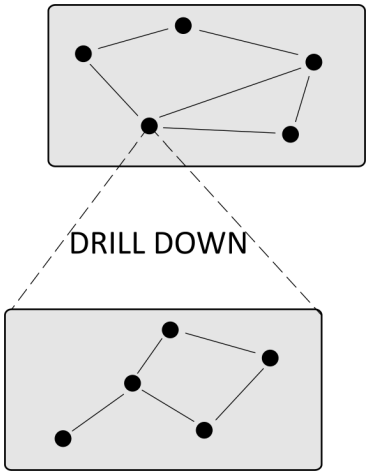


Figure 3.2: Illustration of a system breakdown

3.1.5 Topology

A systems *topology* describes how system elements are interconnected and hence how they interact. In IMF it is assumed that system elements interact through media, i.e., material flow, energy, force, or signal.

Different disciplines will typically address different media. Process engineers may, for example, be interested in how the liquid and gas flow between system elements. The topology within a process system may hence be focused on these types of media. An electrical engineer will be interested in how the electricity flows between the system elements and design the topology of an electrical system accordingly.

A complex system typically involves interaction from different disciplines. This system can be broken down into sub-systems where in each system elements interact through only one single media type. This situation is illustrated in Figure 3.3. The interface between the systems specifies how the systems interact.

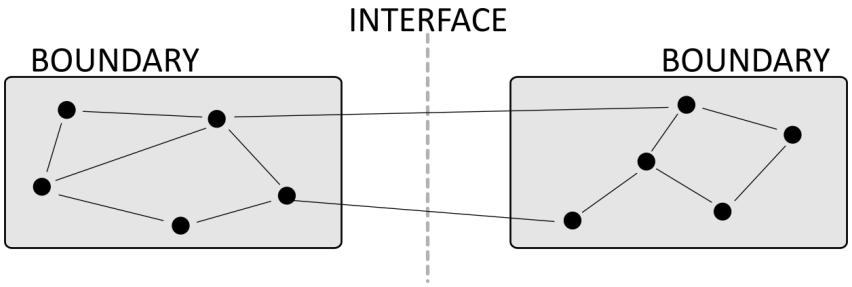


Figure 3.3: Illustration of the interaction of system elements inside and outside of system boundaries.

In the cooling system example, the pump and the heat exchanger interact by means of the cooling medium that the pump pumps 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.

3.1.6 Combining Breakdown and Topology

A repeated use of the principles of system breakdown and interaction gives rise to similar patterns recurring at progressively smaller scales both vertically (system breakdown) and horizontally (system interactions). The pattern is illustrated in Figure 3.4.

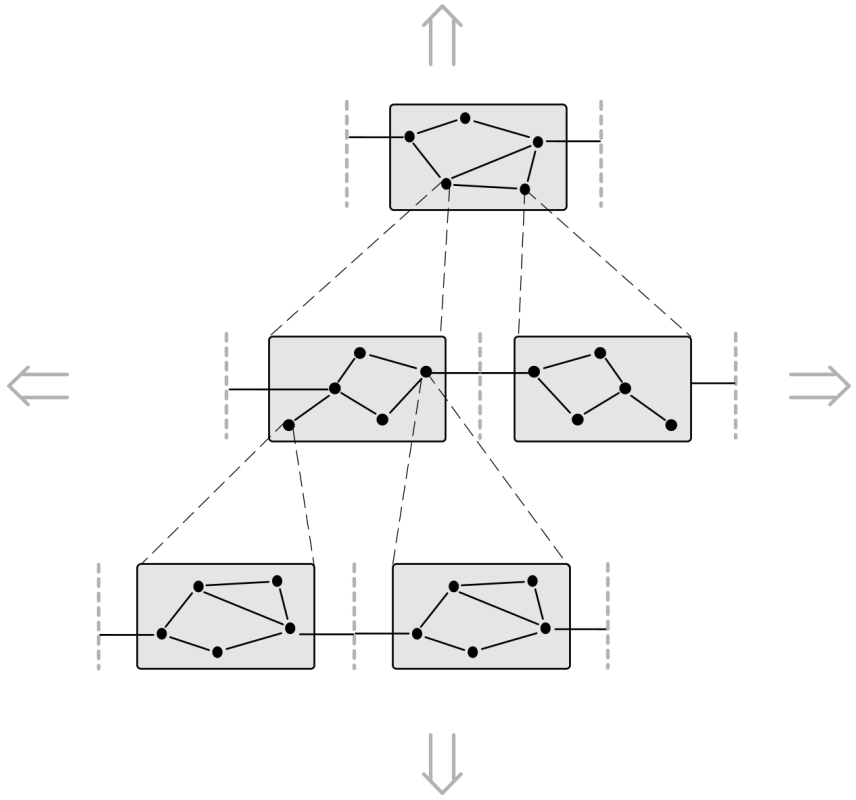


Figure 3.4: Illustration of system breakdown and topology.

3.2 Describing Systems through Aspects

In engineering practice, it is natural to describe systems from different points of view, typically from the views of physical artefacts, activity, and location. IMF is a language designed for describing systems from such different points of view, which we refer to as "*aspects*". The notion of *aspect*s is essentially the context of modelling that needs to be clarified before modelling, e.g., whether it is about the physical artefact or about the activity, what the purpose of modelling is, and who the users are. Note that the aim of IMF language is to describe systems, not to define systems.

3.2.1 Aspect Definition

In IMF an *Aspect* is defined by specifying the following triple: <*Perspective*, *Interest*, *Modality* >.

3.2.2 Perspective

A *perspective* refers to a particular view of a system or system element. The IMF manual mainly addresses the following perspectives (Figure 3.5) and allows extension by users:

- *Activity* is to view the system from the perspective of the activity that a system performs or is designed to bring about;
- *Artefact* is to view the system from the perspective of physical objects or software;
- *Location* is to view the system from the perspective of the geometry, or position (geographical position).

Closely related to the *activity* perspective is the perspective of system *Function*, referring to the capability of a system to bring about an activity. This concept is fundamental in systems design. The concept of *function* as *activity* are a dualism of *intended* versus *actual*, and frequently discussed together. The names for perspective reflect class naming in the Industrial Data Ontology (IDO).

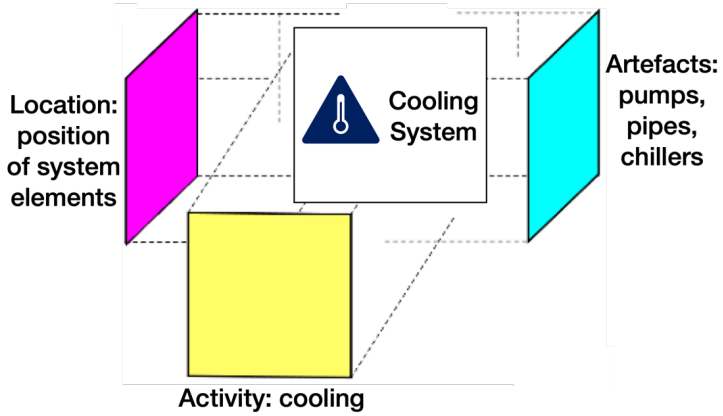


Figure 3.5: The same system is viewed from different perspectives.

For example, from the perspective of artefact, a cooling system is a set of physical objects, such as pumps, pipes, and chillers. From the perspective of activity (function), the function of the cooling system is to realise a cooling activity. From the perspective of location, each artefact system element has a position in and a geometric extension.

3.2.3 Interest

The *interest* is the modelling purpose; it points to the intended usage of the information, the users of the information and their preferences and working habits. The categorisation of interests points to the different stages of the lifecycle of engineering systems, such as system design, detailed engineering, requirement engineering, manufacturing, procurement, commissioning, built, operation, maintenance, disposing, etc. Since the interest contains the information of purpose, usage, and users, it also provides principles of system modelling with breakdown and interaction, and hence how the system elements will be hierarchically grouped and broken down, and how the interactions shall be understood.

Each interest will be further elaborated in future versions of the IMF manual, and aligned to a system lifecycle ontology introduced in the future.

3.2.4 Modality

The *modality* tells if a set of information about engineering systems is a *requirement* or an *intended solution* of the requirement, or it is an *actual solution*. The modality deals with two dualism, the *intended* versus *actual*, and *requirement* versus *solution*, where a requirement is always intended, thus resulting in three modalities:

- *Requirement* is to specify the intended system requirements, e.g., the functional requirements from the clients;
- *Intended Solution* is to describe the intended solution for a set of requirements, e.g., the product plan from the supplier;
- *Actual Solution* is the actual solution for the requirements, e.g., the actual installed artefacts in a factory.

Important to note is that the same set of information can be requirement for some group of information user, and at the same time intended solution for another group of information users. For example, if the interest is procurement, the client A provides the datasheet about some product to specify the required product (requirement). Then the supplier B provides the datasheet about some product based on some product catalogue that the supplier B can deliver (intended solution). The datasheet with the modality requirement will be verified against the datasheet provided with the modality intended solution. The later one then can serve as the requirement for the internal units of the supplier B, or the supplier C of the supplier B.

The evolving of information of the three modalities typically follows that in Figure 3.6: for the initial requirement an intended solution is proposed; then both of them are evolved to the revised requirement in the next stage, for which again an intended solution is proposed; this revision repeats until at the end an actual solution is installed.

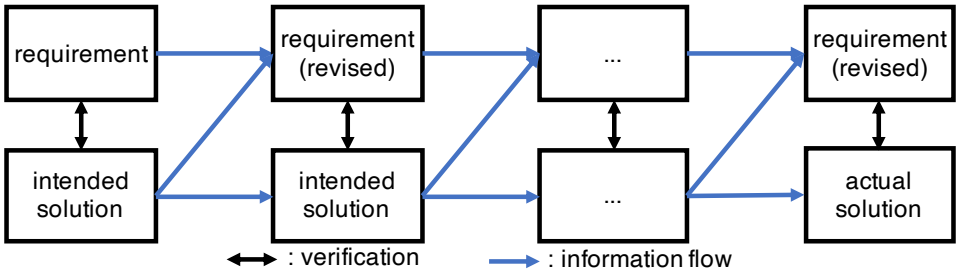


Figure 3.6: The three modalities and a typical evolving flow

3.2.5 Reserved Aspects

The IMF manual uses reserved names for the following four aspect definitions (Table 3.1):

- Function Aspect = <Activity, System Design, Requirement>
- Product Aspect = <Artefact, Built, Intended Solution>
- Location Aspect = <Location, Geometry and Position, Intended Solution>
- Installed Aspect = <Artefact, Built, Actual Solution>

Each aspect has a color assigned. The colors are included in Table 3.1 and are used in later figures to indicate the belonging aspect. Figure 3.7 illustrates these aspects as being different perspectives on the information about an intended pumping activity.

- The *Function Aspect* (yellow) ❶ is to specify the intended activity, e.g.,: the information about activity, performance, and function about a pump.
- The *Product Aspect* (cyan) ❷ is to specify the intended solution of a set of physical artefacts to perform the activity, e.g., the information about the physical artefacts of the pump.
- The *Location Aspect* (magenta) ❸ is to specify the geometrical and positional information of the intended solution, e.g., the size and shape of the specified pump.
- The *Installed Aspect* (dark blue) ❹ is to describe information about the actual solution, e.g., the serial number, run hours, and status of an actual pump installed in a plant.

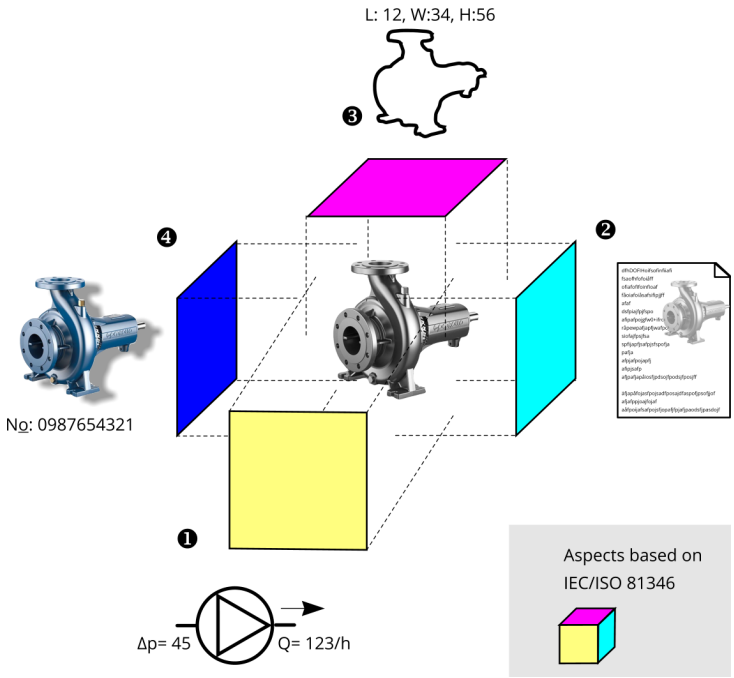


Figure 3.7: The four reserved aspects are illustrated: the Function (yellow), Product (cyan), Location (magenta), and Installed (dark blue) aspects.

Table 3.1: Definition of the four aspects and their prefixes and colors.

Aspect	Definition	Prefix	Color
Function Aspect (F)	The functional requirements to the intended activity	=	yellow
Product Aspect (P)	The intended solution of physical artefacts	-	cyan
Location Aspect (L)	The geometrical and positional information of the intended solution	+	magenta
Installed Aspect (I)	The description of the actual artefacts installed in industrial facilities	::	dark blue

A core concept of ISO/IEC 81346 is the notion of an aspect. IMF borrows this concept, but both widens it and makes it more specific. The IMF concept is wider than the aspect concept in ISO/IEC 81346 because it also addresses the individuals of a facility. ISO/IEC 81346 explicitly states that the standard only addresses so-called occurrences, which can be viewed as tag objects that are distinct from individuals. Also, IMF makes the topology more explicit than it is in ISO/IEC 81346. The IMF concept is narrower in the sense that an aspect is a specific information structure. Also, in contrast to ISO/IEC 81346, IMF does not treat a type as a distinct aspect.

3.2.6 Breakdown and Topology in Different Aspects

The different aspects of same system have different breakdown and topology.

For the reserved aspects, the breakdown of a system indicates the relations between the system elements and the system it comprises. Between each two layers of the breakdown, we refer to the system elements in the lower layer as sub-systems of the system in the higher layer. In particular, the breakdowns in four aspects indicate follows:

- For the Function Aspect, the breakdown means the intended activity of a sub-system is a sub-activity of the system of the higher layer.
- For the Product Aspect, the breakdown means the intended product artefact of a sub-system is a sub-assembly of the product artefact of the higher layer.
- For the Location Aspect, the breakdown means the intended location of a sub-system is in a sub-location of the system in the higher layer.
- For the Installed Aspect, the breakdown means the actual installed artefact of a sub-system is a sub-assembly of the installed artefact.

For the reserved aspects, the topology (connections) of a system indicates different interactions that connect the terminals of system elements.

- For the Function Aspect, the connection between two system elements A and B means the media state of a system element A is equal to that of the system element B.
- For the Product Aspect, the connection between two intended system elements A and B means the system element A is physically connected to system element B via some media, e.g., material flow, force.
- For the Location Aspect, there exists currently no definition in the IMF manual.
- For the Installed Aspect, the connection between two actually installed system elements A and B means the system element A is physically connected to system element B via some media.

3.3 Reusable Patterns and Types

In the context of this section a *pattern* is an *information structure* that partially details an engineering solution. A pattern will typically contain attribute information, with or without values, potentially with a range of admissible values that an attribute may take, information about interfaces, and, in complex cases, also with some detail about a breakdown of the pattern into elements and a topology over those elements.

In all lifecycle phases engineers reuse patterns. In standardised design such patterns are called, e.g., design codes, typicals, or best practice. In commercial practice they may be called, e.g., modules, packages, or product types.

In IMF, we refer to a reusable pattern as a *Type*, which is a predefined *template* for a information structure. Types in IMF are meant to reside in a library from which it can be accessed. When a type is selected, two operations on types are needed:

- Configuration. In the simplest case attribute values are inserted. In addition new attributed may be added. In complex cases more structure is added.
- Instantiation: The configured type, with associated identifiers, is inserted into an IMF model.

See Figure 5.9 for an illustration of how these ideas can be implemented and applied.

In the current version of the manual, only elementary types are addressed, with the exception of the experimental text in Section 5.9 and in Section 6.6.

3.4 Syntax and Interpretation

IMF is a framework that consists of a family of languages. In IMF a language specification consists of:

- A *syntax* that prescribes what counts as admissible expressions in the language. The syntax can be specified formally, so that it can be parsed by computer programs, or informally so that humans can understand how to write expressions in the language, possibly allowing some degree of freedom in exact ways of how to write them; such syntactical freedom is unproblematic as long as it does not compromise clarity;
- An *intuitive interpretation* (typically informal) aimed at users of the language. This interpretation should contain sufficient explanation so that the users understand how to express their use cases in the language and how to understand use cases that others have expressed in the language;
- A *formal interpretation* that captures the meaning of the language in terms of mathematically precise structures. Formal interpretations of IMF will be specified by translation into logical languages that have a precise formal (i.e., set theoretic) semantics, such as OWL.
- In order to be useful in practice, the IMF languages will need translations to other languages (typically formal languages) that preserve syntactical structures. We refer such translations as *structure preserving translations*. If the source language and the target language have different structures, such translations are difficult to specify.

All languages in the IMF family are object languages. Hence Section 3.4.1 briefly reviews the terminologies of object languages.

3.4.1 Object Language and Object Model

An object language consists of

- Mechanisms for creating *objects*, including mechanisms for giving the objects a unique identifier;
- Mechanisms for attaching *attributes* to objects;
- *Relations* that connect objects. Typically we discuss *binary relations* where a relation exactly connects two objects.
- Mechanisms for creating *relationships*, which are instances of relations.

An object model is an instance of an object language, consisting of a set of expressions that specify objects, attributes or relations. For example, the following is an object model:

- A set of objects attached with tag numbers (i.e. identifiers) created by a tag master process that ensures unique tag numbers;
- Instances of the relation `mountedOn`. If `t1` and `t2` are tag numbers, `mountedOn(t1,t2)` is a relationship in the model.

A model with only binary relations can be naturally transformed to a directed graph, where objects are nodes, and relationships are edges between the objects.

3.4.2 IMF as Object Languages

In IMF, objects are called elements. Elements are created from types in analogy with the way that objects in object-oriented programming language are created for execution of a computer program.

It is useful to enrich an object language with classes ordered in a classification hierarchy. IMF also has hierarchies of types similar to classification hierarchy, but a type in IMF is not the same as a class as typically used in formal languages such as OWL: IMF types are constructive, and used to reuse patterns and create elements. Classes are essentially constraints used to classify elements.

3.4.3 Visual Syntax

IMF models are meant to be developed by engineers by manipulation graphical forms using applications. For this reason IMF has a visual syntax. While the precise syntax supported by an application is left for the specification of that application, the visual syntax used in this manual exemplifies how this can be done.

In this version of the manual displayed elements of an IMF model are blocks and terminals, while relationships between them are displayed using lines just as one would display binary relationships as edges in a graph.

In later version of this manual this syntax will be extended so that the lines can hide an object, called an association point. In this way we can associate attributes to a relationship, and also relations to other elements.

3.4.4 Informal Interpretation

The informal interpretation of IMF is the given by the intuitions, examples, explanations, and discussions of this manual, and especially chapters 5 and 6.

3.4.5 Formal Syntax

The formal syntax of IMF is expressed in the ontology languages OWL and SHACL, both standardised as W3C recommendations. It is specified in Chapter 7 of this manual.

3.4.6 Formal Interpretation

The formal interpretation of IMF is defined by a translation that maps IMF models into models in the language of the Industrial Data Ontology, IDO. In this translation an element (i.e., a block or terminal) in an IMF model is mapped to

- An OWL class restriction if the modality of the aspect of the element is Requirement or Intended solution;
- An OWL individual if the modality is Actual solution.

The interpretation will map information about the same system that is spread over several elements into one system object and adjust domain and range of relations. Note that the notion of a system is a concept in IDO, but not in IMF.

The formal interpretation is essential for integrity checking of an IMF model. This can in many cases be efficiently implemented using reasoning over the IDO interpretation of the IMF model at hand.

3.4.7 Structure Preserving Translations

If a model in a different language than IMF, e.g., AML, is designed according to the IMF principles, it is in theory straightforward to translate between such model and a corresponding IMF model.

This point will be addressed in forthcoming versions of this manual.

Chapter 4

IMF Language: Quick Reference

4.1 Language Elements Overview

The IMF language consists of

- Aspect elements
- Binary relations between aspect elements

Aspect elements in an IMF model have been created as instantiations of types and relationships between aspect elements inserted as instances of relations.

Figure 4.1 on page 23 gives an overview of terms and symbols in the IMF language.

Note that Figure 4.1 lists Association Point as an element. This element is not used in the current version of this manual.

4.1.1 Aspect Elements

Aspect Elements are blocks and terminals with exactly one aspect, as listed in Table 4.1.

Table 4.1: The different Aspect Elements.

	Element	**Block (B)**	**Terminal (T)**
Function (F)	Function Element	Function Block (FB)	Function Terminal (FT)
Product (P)	Product Element	Product Block (PB)	Product Terminal (PT)
Location (L)	Location Element	Location Block (LB)	Location Terminal (LT)
Installed (I)	Installed Element	Installed Block (IB)	Installed Terminal (IT)

4.1.2 Relations

Relations in IMF are binary: they connect two aspect elements. Table 4.2 and Table 4.3 list all relations with their domain, range and cardinality.

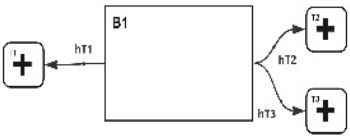
Elements

(a) Block

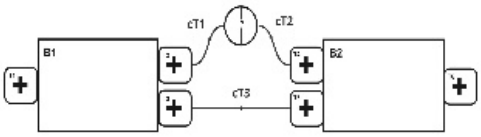
(b) Terminal

(c) Association Point

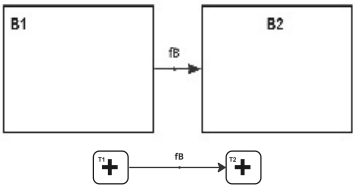
Relations



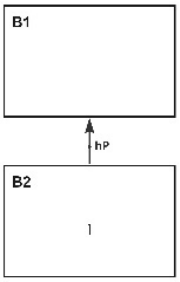
(d) hasTerminal



(e) connectedTo

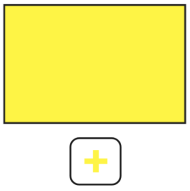


(f) interAspect

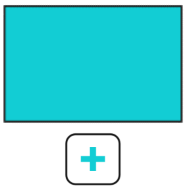


(g) partOf

Aspects



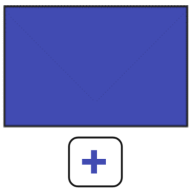
(h) Function



(i) Product

Terminal is not used

(j) Location



(k) Installed

Figure 4.1: An overview of terms and symbols in the IMF language.

Table 4.2: Summary of partOf and connectedTo relations.

Relation	Domain	Range	Cardinality
partOf	Block/Terminal	Block/Terminal	Many – 1
connectedTo	Terminal	Terminal	1 – 1

Table 4.3: Inter-aspect relations and their domain and range and cardinality

Relation	Domain	Range	Cardinality
asProduct	FunctionElement	ProductElement	1 – many
asProduct	InstalledElement	ProductElement	1 – 1
asFunction	ProductElement	FunctionElement	1 – many
asLocation	ProductElement	LocationElement	1 – 1
asInstalled	ProductElement	InstalledElement	1 – many

4.1.3 Visualisation

A Block is visualised as a box, possibly with a collection of attributes written inside. A Terminal is visualised as a small box with a plus (+) sign, as shown in Figure 4.2.

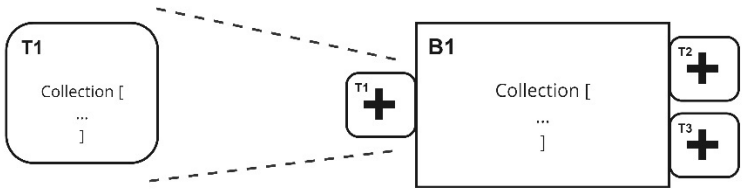


Figure 4.2: A Terminal.

A partOf relationship partOf(B2, B1) is illustrated with an arrow pointing from the top of B2 to the bottom of B1, see Figure 4.3.

In this manual a connectedTo relationship is visualised as line between the Terminals it connects, as illustrated in Figure 4.4. Figure 4.4 also illustrates a connectedTo relationship with an associated Association Point as a split circle placed in the middle of the line between the connected Terminals. This is not further used in the current version of this manual.

In an IMF model the connectedTo and partOf relation are constrained so that they only relate blocks associated with the same aspect.

The details of the blocks and their interrelations is described in Chapter 7.

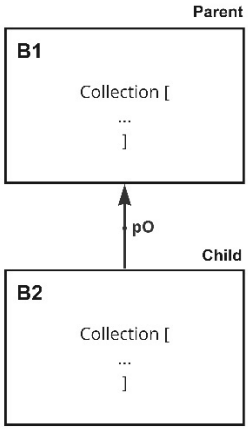


Figure 4.3: A partOf relationship between Blocks

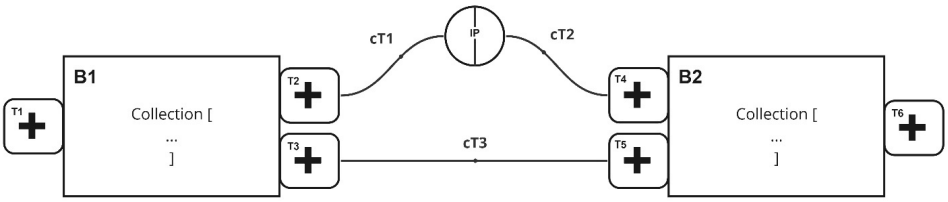


Figure 4.4: Two connectedTo relationships

4.2 Block

Block is the basic building block of the IMF language. A Block represents something of interest to the SME by setting the boundaries of anything which is convenient to treat as an entity. This could be a whole industry plant, a pump system, or a small location of interest.

Table 4.4 summarises the informal interpretation of blocks in each of the reserved aspects.

Table 4.4: Aspect Blocks and their intuition.

Name	Intuition
Function Block (**FB**)	A Function Block holds the requirements to an intended activity.
Product Block (**PB**)	A Product Block holds the specification of an intended artefact.
Location Block (**LB**)	A Location Block holds the specification of the spatial extension of an intended artefact.
Installed Block (**IB**)	An Installed Block holds the documentation of an actually installed artefact.

4.3 Terminal

A Terminal represents a channel of communication for a Block; hence a Terminal cannot exist without a Block. A Block may have any number of Terminals that each represents a different communication

channel or port with which the Block may receive input and/or give output. A Terminal that is specified to only receive input is called an *input* Terminal. A Terminal that is specified to only give output is called an *output* Terminal. A Terminal that may both receive input and give output is called a *bidirectional* Terminal, or, simply, a Terminal.

Table 4.5 summarises the informal interpretation of terminals in each of the reserved aspects.

Table 4.5: Aspect Terminals and their intuition.

Name	**Intuition**
Function Terminal (**FT**)	Requirements to one input/output stream state of a intended activity
Product Terminal (**PT**)	Specification of one input/output/bidirectional terminal of an intended artefact
Location Terminal (**LT**)	Not used
Installed Terminal (**IT**)	Documentation of an actually installed artefact terminal.

4.4 hasTerminal

Table 4.6 summarises the informal interpretation of hasTerminal relationships in each of the reserved aspects.

Table 4.6: hasTerminal relations and their intuition.

Relationship	**Intuition**
hasTerminal(**FB**,**FT**)	The media state of Function Terminal **FT** is the input/output of the intended activity in Function Block **FB**
hasTerminal(**PB**,**PT**)	The Product Terminal **PT** is a specification of one input/output terminal of the intended artefact of Product Block **PB**
hasTerminal(**LB**,**LT**)	Not used
hasTerminal(**IB**,**IT**)	The actually installed terminal **IT** holds documentation of an actually installed artefact terminal **IB**

4.5 partOf

The partOf relation promotes a System-Of-Systems way of thinking, cf. Section 3.1.4.

Table 4.7 summarises the informal interpretation of partOf relationships in each of the reserved aspects.

4.6 connectedTo

Table 4.8 summarises the informal interpretation of connectedTo relationships in each of the reserved aspects.

Table 4.7: partOf relations and their intuition.

Relationship	Intuition
partOf(**FB1**,**FB2**)	The intended activity of Function Block **FB1** is a sub-activity of that of Function Block **FB2**
partOf(**PB1**,**PB2**)	The intended artefact of Product Block **PB1** is a sub-assembly of the intended artefact of Product Block **PB2**
partOf(**LB1**,**LB2**)	The intended location of Location Block **LB1** is located in the intended location of Location Block **LB2**
partOf(**IB1**,**IB2**)	The actually installed artefact of Installed Block **IB1** is sub-assembly of the actually installed artefact of Installed Block **IB2**

Table 4.8: connectedTo relations and their intuition.

Relationship	Intuition
connectedTo(**FT1**,**FT2**)	The media state of Function Terminal **FT1** is equal to that of Function Terminal **FT2**.
connectedTo(**PT1**,**PT2**)	The Product Terminal **PT1** is physically connected to the Product Terminal **PT2** via some media.
connectedTo(**LT1**,**LT2**)	Not used
connectedTo(**IT1**,**IT2**)	The Installed Terminal **IT1** is physically connected to the Installed Terminal **IT2** via some media.

4.7 Inter-Aspect Relations

Table 4.9 summarises the informal interpretation of the inter-aspect relationships asFunction, asProduct, asLocation, asInstalled, relationships in each of the reserved aspects.

These relationships are called *proxies* because, intuitively, two elements related in this way express different pieces of information about the same system element.

Table 4.9: inter-aspect relations and their intuition.

Relationship	Intuition
asFunction(**PB**,**FB**)	The intended artefact of Product Block **PB** realises by the entire or parts of the intended activity of Function Block **FB**
asFunction(**PT**,**FT**)	The intended artefact terminal of Product Terminal **PT** provides physical connection of the media state of Function Terminal **FT**
asProduct(**FB**,**PB**)	The intended artefact of Product Block **PB** realises the entire or parts of the intended activity of Function Block **FB**
asProduct(**FT**,**PT**)	The intended artefact terminal of Product Terminal **PT** provides physical connection of the media state of Function Terminal **FT**
asLocation(**PB**,**LB**)	The Location Block **LB** describes the spatial extension and position of the intended artefact of Product Block **PB**
asInstalled(**PB**,**IB**)	The actually installed artefact of Installed Block **IB** realises the intended artefact specifications of Product Block **PB**
asInstalled(**PT**,**IT**)	The actually installed artefact terminal of Installed Terminal **IT** realises the intended terminal specifications of Product Terminal **PT**

4.8 IMF Models: Constraints and Understandings

An IMF Model is a collection of instantiated Aspect Elements and relationships between them.

4.8.1 Constraints of IMF Language on Modelling Systems

IMF has the following constraints on describing systems:

- An IMF model can contain system descriptions of multiple aspects, e.g., the system description of Function Aspect, Product Aspect, Location Aspect together represents different perspectives and interests of the Facility Asset as a whole.
- A partOf relation can have at most one "parent" in the breakdown.
- Aspect elements of one aspect are connected via hasTerminal, partOf, or connectedTo only to aspect elements of the same aspect;
- Aspect elements of different aspects are connected by inter-aspect relations.
- IMF does not allow cross-aspects systems breakdown and systems topology, i.e., the description of any aspect can only contain system elements of the same aspect.
- All blocks in a model that can be reached by following a chain of connectedTo and hasTerminal relationships should be at the same scale.

4.8.2 Understanding Breakdowns

Intuitively blocks that capture information at different scale are inserted in a breakdown structure. We can think about a breakdown as a way of zooming to inspect the elements more closely. In IMF, a feature of a breakdown relation is that it can have at most one "parent" in the breakdown. As a consequence, a

breakdown can be viewed as the result of applying a breakdown principle in either direction (i.e., both for zooming in and zooming out).

Different usage of the information modelling may need different principles for breakdown, even for elements viewed from the same perspective, but for different interests. System elements of artefacts, for example, may be grouped according to their geometry and weights for logistics, while for procurement the system elements may need to be grouped according to their suppliers, and a built breakdown for operation and maintenance will typically group elements according to how they are assembled and mounted. Different breakdowns result from applying different breakdown principles. These different breakdowns have to be modelled through breakdowns in different aspects.

4.8.3 Understanding Topology

Blocks capturing information at the same scale are inserted in a topology. Hence all blocks in a model that can be reached by following a chain of connectedTo and hasTerminal relationships will be at the same scale. A block structure spanned by a breakdown and a topology has a shape reminiscent of a fractal. It is a pattern that is repeated in a recursive way and that can be navigated by zooming in and out reflecting views of different scale. In combination the relations facilitate the zooming in and out between abstraction levels and topology structures that is illustrated Figure 3.4.

Chapter 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 5.1 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 diagrams or drawing documents, but by creating an Information Model, by composing a structure of building blocks with relations, both in hierarchy (vertical) and in topology (horizontal). The level of detail that is put into it – the granularity – is dependent on the use case. This applies both to the detail of structure that is composed, as well as the extent to which attributes are assigned.

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 features 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

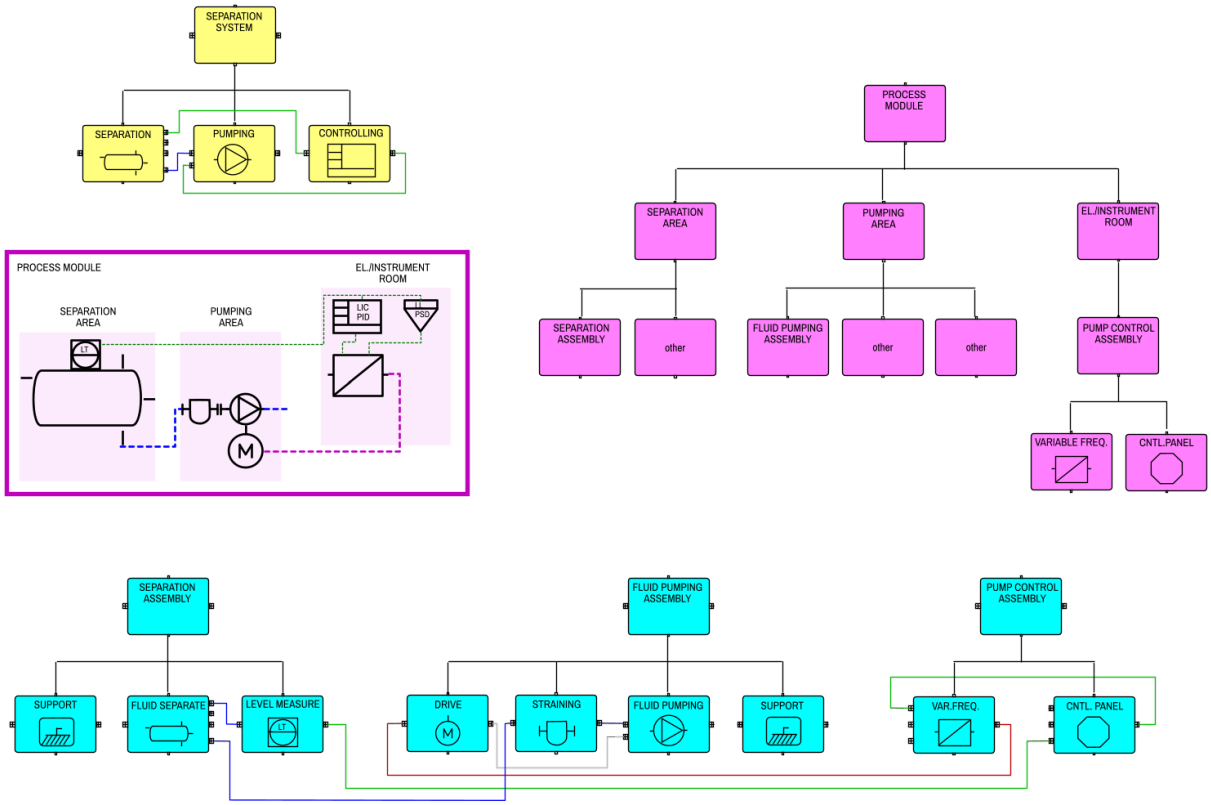


Figure 5.1: A separation system represented as an IMF model with three aspects: Function, Location, and Product.

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 (or assembly, equipment, device, component).

The dividing 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 thing 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 5.1 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 that apply. Referring to the Separation plant given in Figure 5.1, 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 the actual information 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 N^o 5270-2565.

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

5.2 Before one Starts to Model

Before starting to model, we need to be clear on the purpose or *interest* of the modelling. The interest should be defined based on the end user needs, or the user needs next in the value chain. Often this is given by the work process the user is executing at this point in the Life Cycle. Figure 5.2 illustrates a separation system, shown as a conceptual diagram and as an imagined real solution.

5.2.1 Defining the Interest of the Model

The interest of the modelling can range from defining requirements at a conceptual level – to documenting the actual Facility Asset as installed. Likewise, the interest can be to model the automation and control part of the solution, or the fluid processes. In other cases, the interest 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 interest of why and 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.

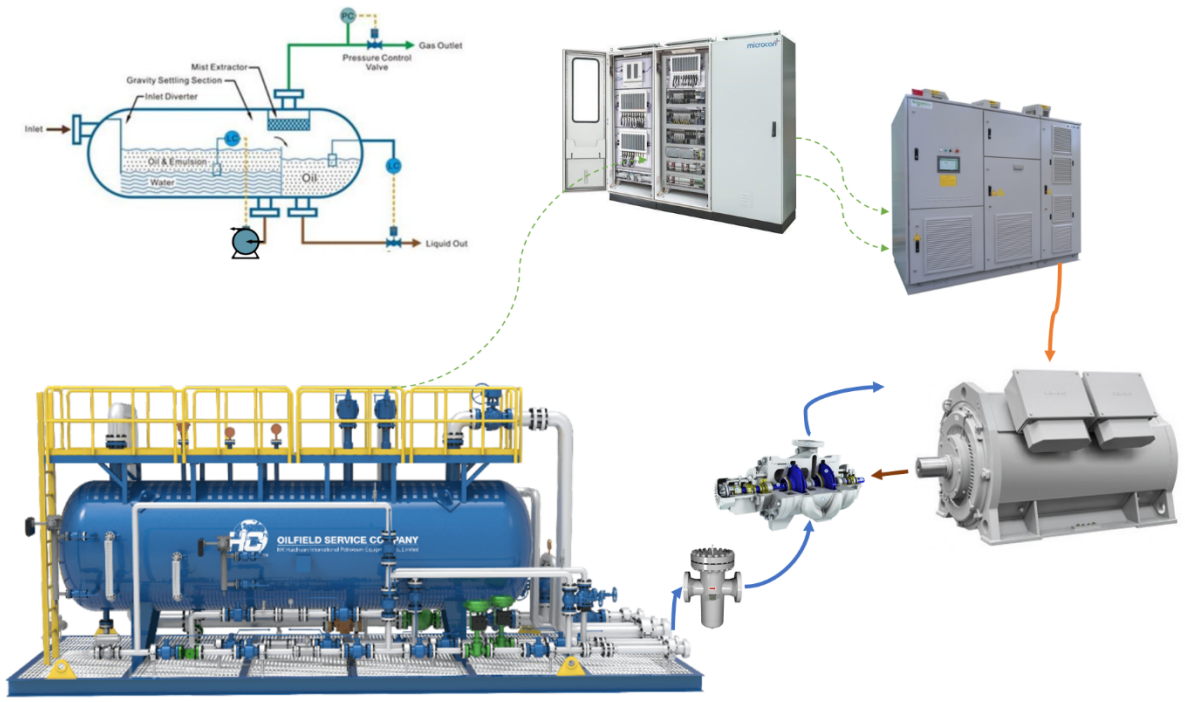


Figure 5.2: A separation system shown as a conceptual diagram and as an imagined real solution.

As an example, this could be safety requirements for systems and solutions that handle high pressure hydrocarbon fluids. 'Overall Requirements' means any information that has this purpose, in general they are found outside of the model, in documents, authority requirements, etc. To include these into the IMF model may only be feasible by using , say an IMF Type "Requirements Link" which contains a link to the external documentation of the requirements, but in the future such requirements could be implemented as a model.

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 default taxonomy for granularity levels which can be employed to define the intended level of detail. This taxonomy is defined by ISO/IEC 81346-O&G [4] 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).

There may be a need for additional and more fine-grained taxonomies, such as that specified by ISO 14224 [1] or other industry standards or conventions. 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. It is possible to begin modelling without having decided how deep in granularity to go, or how wide in scope to span, but this would not be best practice (indeed it is not best practice for any kind of effort). Good planning of the work would include to target a specific scope and a specific level of detail.

5.3 How to Choose which Aspect to Begin with

A proposed sequence of aspects is:

$$\text{Function} \Rightarrow \text{Product} \Rightarrow \text{Location} \Rightarrow \text{Installed}$$

However, it 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. It may help structure the work process to choose which aspect to begin with and think of as the 'master' aspect.

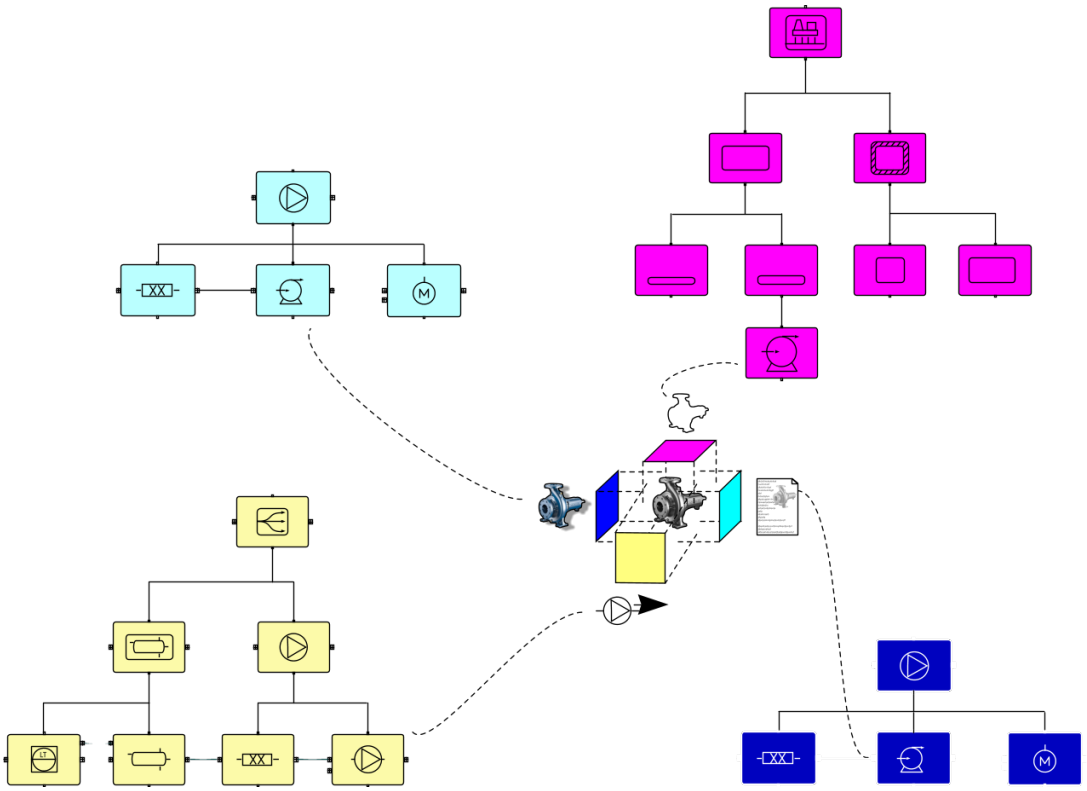


Figure 5.3: A pump system represented as an IMF model using four aspects: Function, Location, Product, and Installed.

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 the 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 materials technology.

Move from this aspect at the latest when the solution has been sufficiently specified for procuring/acquiring.

5.3.3 The Location Aspect

The location aspect can represent the physical - architectural/construction - perspective, meaning modules, rooms, decks, areas, etc., with their X, Y, Z coordinates and characteristics. When modelling, choose the physical 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 – area restrictions, e.g., noise limitations, ambient conditions, etc.

Modelling in this aspect depends on knowledge about what will eventually be needed to be located, both the size, weight and suitability for 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 meet 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 have been manufactured/made. 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

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 5.4 to illustrate the different approaches.

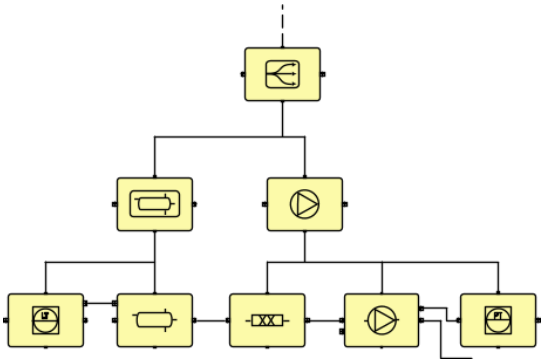


Figure 5.4: An example of an IMF Model

The approach to select for a 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

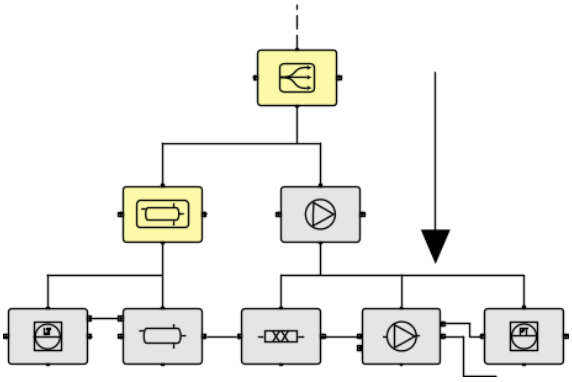


Figure 5.5: A Top-down modelling approach.

The top-down approach shown in Figure 5.5, 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 Blocks with Terminals yet to be modelled.

5.4.2 A Bottom-up Modelling Approach

The bottom-up approach shown in Figure 5.6, 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 Blocks with Terminals yet to be modelled.

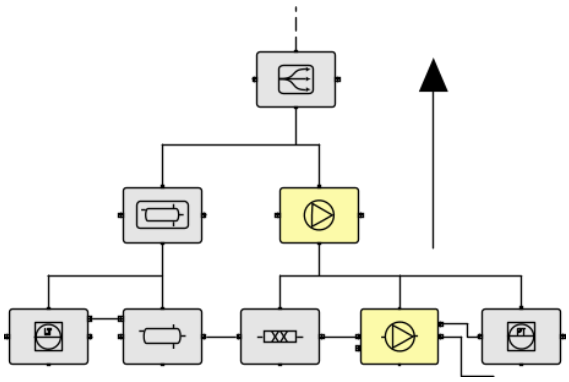


Figure 5.6: A bottom-up modelling approach.

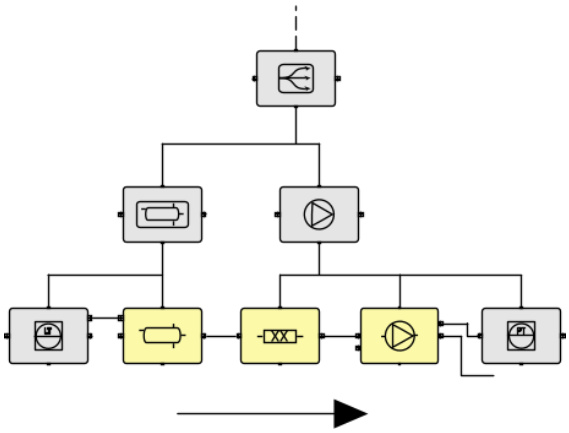


Figure 5.7: A follow-stream modelling approach.

5.4.3 A Follow-Stream Modelling Approach

The follow-stream approach shown in Figure 5.7, allows full attention to 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 Blocks with Terminals yet to be modelled.

5.4.4 A Follow-thread Modelling Approach

As illustrated in Figure 5.8, 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

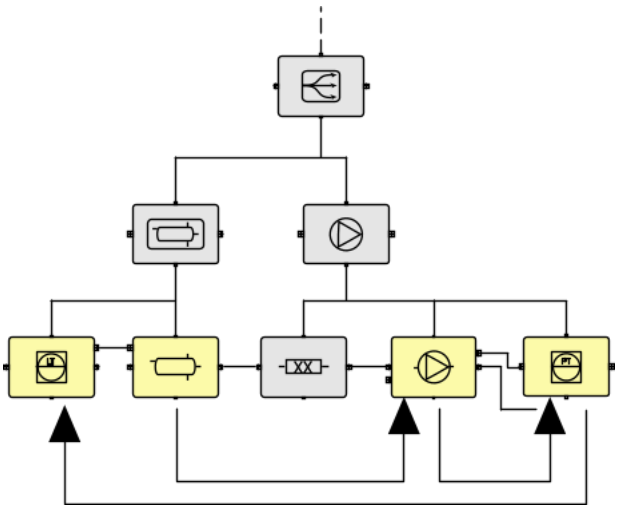


Figure 5.8: A follow-thread modelling approach.

model will require a bottom-up integration modelling effort. Grey coloring indicates systems that do not have primary focus or that do not yet exist.

5.5 How to Create and Connect Blocks with Terminals

The building blocks of an IMF model are the various types of Blocks with Terminals. The Blocks with Terminals are fetched from a library, or more precisely they are created from definitions called IMF Types that reside in the IMF Type library. When a Block with Terminals 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 the 'Pumping'.

To compose a model requires creating Blocks with Terminals and connecting them together. This is done by placing them into the breakdown hierarchy and connecting them together between terminals, and by defining relations to Blocks with Terminals in other aspects. Depending on the modelling approach chosen, note that not all these different types of connections need to be defined initially.

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.

When selecting the IMF Type, its aspect must be chosen (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

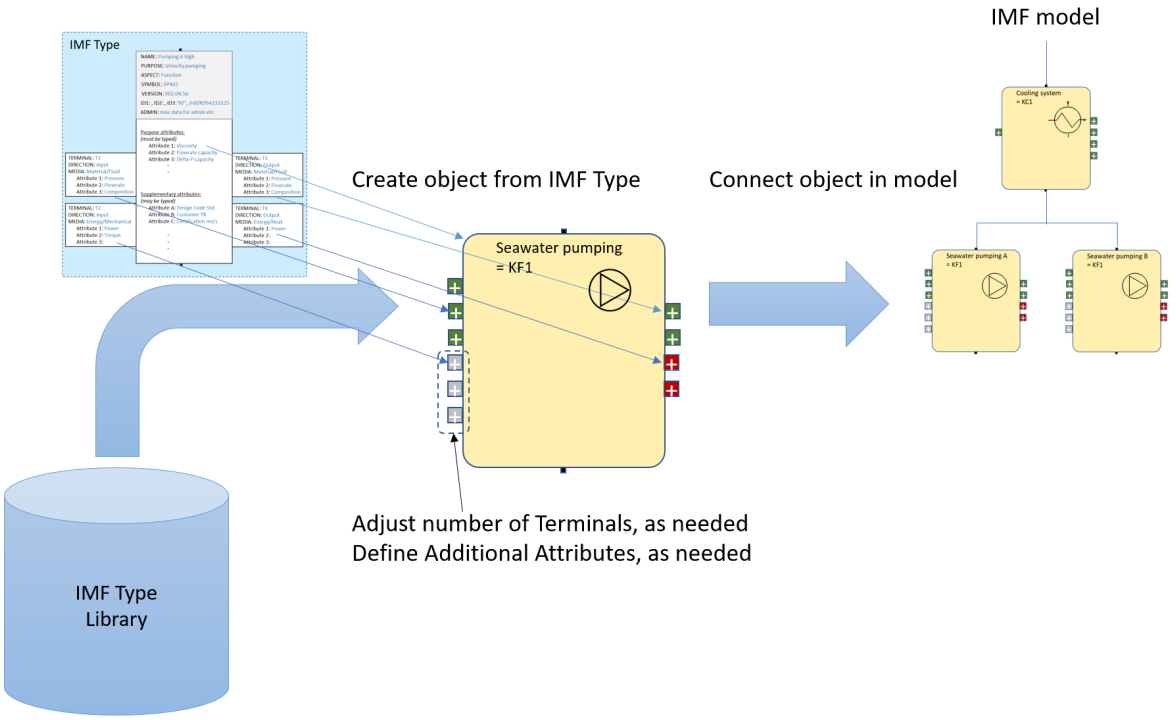


Figure 5.9: Using an IMF Type from a library.

mandate; see Chapter 6. Other IMF Type libraries than the common industry library is likely to be developed, sometimes with limited sharing. They may be needed due to specialization or intellectual property protection or other reasons. IMF is designed to support such a variety of IMF Type libraries.

5.5.3 How to Place the Block with Terminals into the Model

When a Block with Terminals has been 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 hierarchical relations.
- Into the topology of existing aspect blocks - effectively creating topology relations.
- Into the relation between Blocks with Terminals of different aspects - effectively creating inter-aspect 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 Blocks and Terminals

When a new Block with Terminals is created during modelling it will have one or several attributes of specific 'Quality'. This is provided as part of the IMF Type definition. 'Quality' is such as 'Pressure', 'Temperature', 'Speed' etc. It is possible during the modelling to increment as needed the number of (amount) attributes of the same type, as well as specify further specialization of attribute values, such as Quantity Datum, e.g., 'Pressure *Design, Maximum*', and sub-quality, e.g., 'Pressure Design Maximum, *Cavity*. It is also possible to add entirely new attributes to the Block with Terminals, when that is needed

to hold information that is not part of the basic IMF Type definition. Usually this is supplementary information which is not strictly required, but is convenient to include.

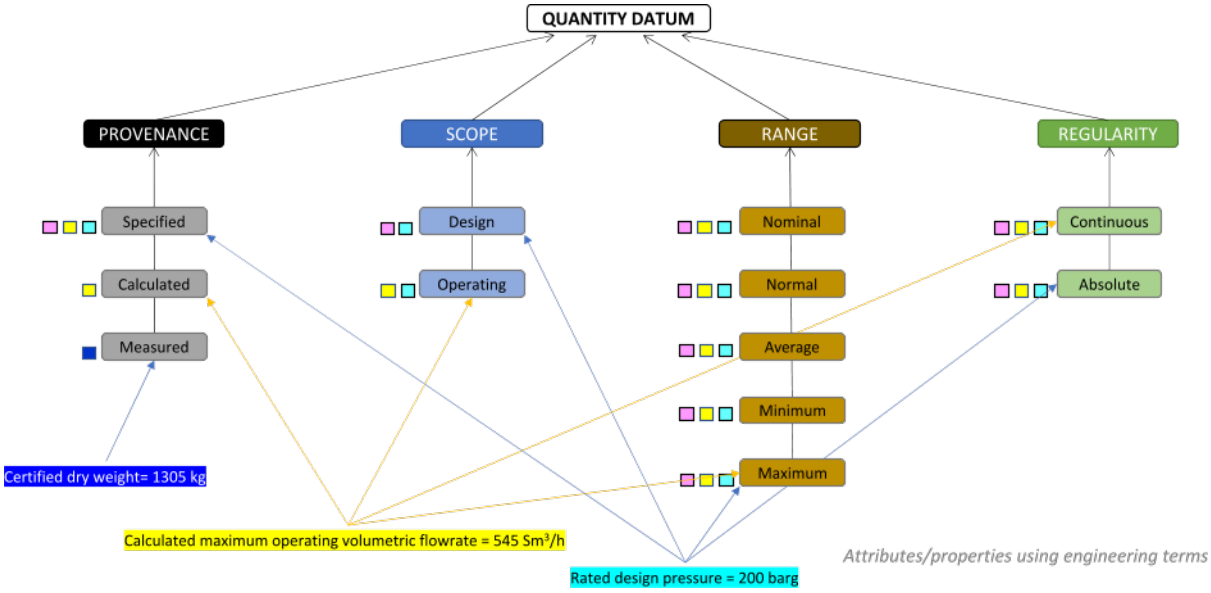


Figure 5.10: Typing of attribute values to provide context.

Assigning attribute values to Quantity Datum classes can be done as shown in Figure 5.10. 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 value Datum types are selected, they are ready to be given values and units of measure can be 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 an actual value, to provide a sufficiently detailed 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 results in.

The above Quantity Datum is based on the concept of quality and quantity datum, and the types of quantity datums from ISO 15926-14.

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 *few or none of the* 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 Terminals

When a new Block with Terminals is created it will have one or several Terminals of each type. This is provided as part of the IMF Type definition. A Terminal can represent an Input, Output or Nodirection, and is defined 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

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 attribute values can be further 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 can be 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 Blocks with Terminals

Blocks will have Terminals, and they are the means for modelling how streams flow. An Output Terminal of one Block needs ultimately to be connected to the Input Terminal of another Block in order to model a stream. At the high level of the model the connection between Output and Input may simply be viewed as a transport connection, whereas at lower levels more details could emerge, revealing that this transport connection comprises one or several pipes, shafts, or cables – possibly with some devices, each having specific characteristics.

5.5.9 Iterate on Creating and Editing Blocks with Terminals

As more and more Blocks with Terminals 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 Blocks with Terminals 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. It is therefore important to understand what the potential far-reaching effects of a change could be. A change to a Block with Terminals could mean that other Blocks with Terminals 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 part of. To alleviate this, the tool used for modelling may have mechanisms that warn about such consequences.

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.

The Location aspect is more distinct and is entirely different from the other aspects. Because this aspect will hold area 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 the location. The Installed aspect is solely used for documenting real world physical objects, and how they fulfil the specifications given in the Product aspect.

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 5.11.

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

1. The logic 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 for 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

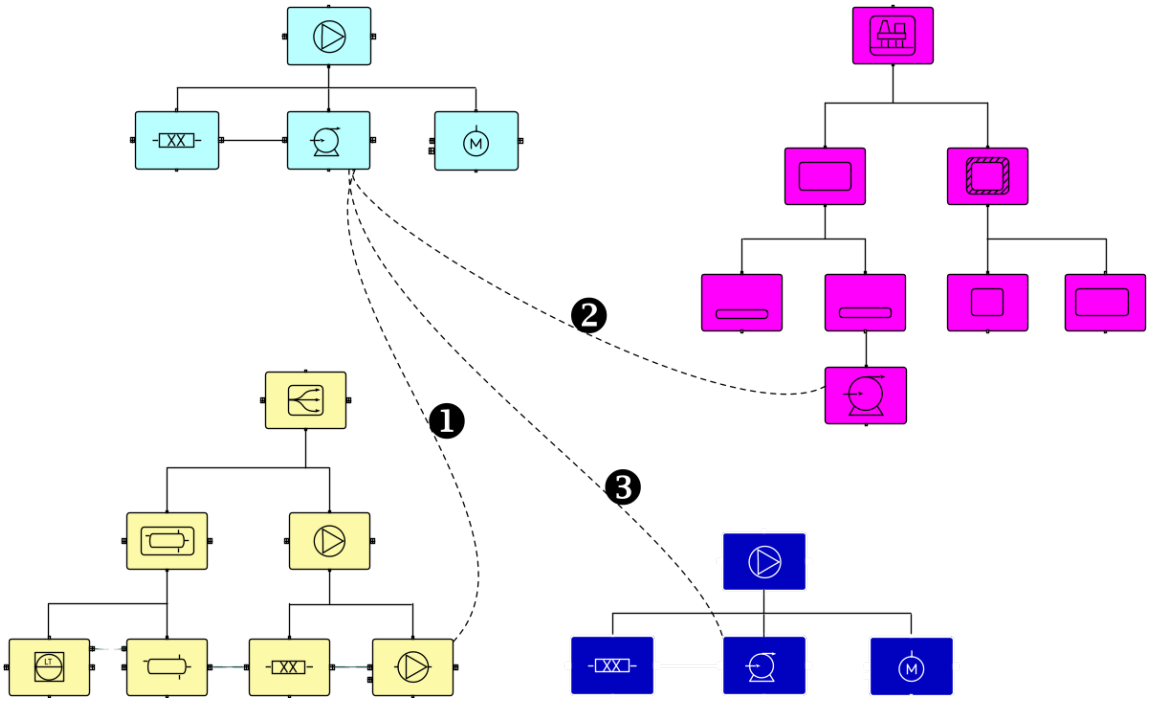


Figure 5.11: Inter-aspect relations

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, the requirements are given by the parent Block. E.g., when a pump is located in a process area it is the requirements of that process area that apply to the pump.

1. 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 Blocks with Terminals will have relations to other aspects. A Block with Terminals 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 Block with Terminals representing the requirements, and the Product Block with Terminals 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 Block with Terminals representing the solution, and the Location Block representing the spatial properties of the solution, *and* by connecting the Location Block 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 Installed Block with Terminals representing the actual component, equipment, or assembly, and the Product Block with Terminals 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. New aspects can be created based on simple rules and components (planned in a later version).

Connecting relations to these possible future Blocks 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 5.12. More advanced mechanisms for supporting this are planned for a later version of IMF. The integration can be between models in different Aspects (1) or between different multi-aspect models (2).

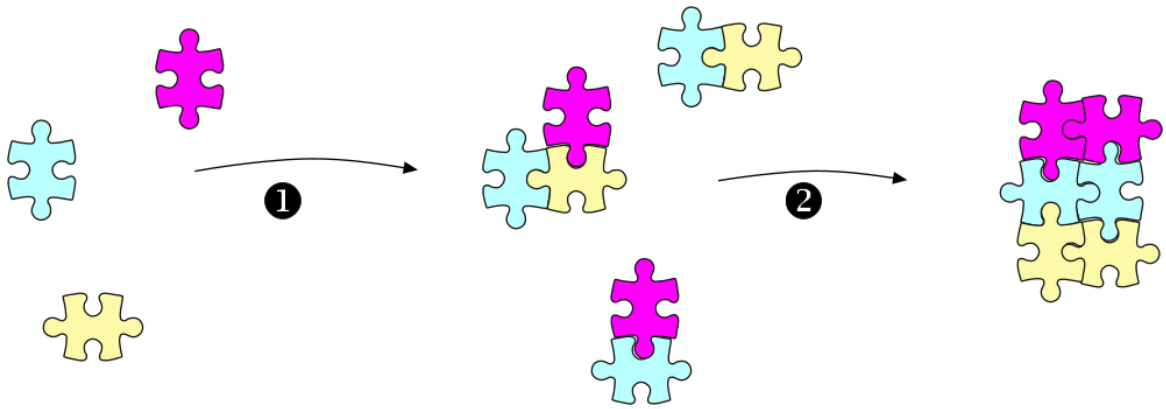


Figure 5.12: The concept of model integration

This supports a work process where the design work is split into 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.

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 the figures.

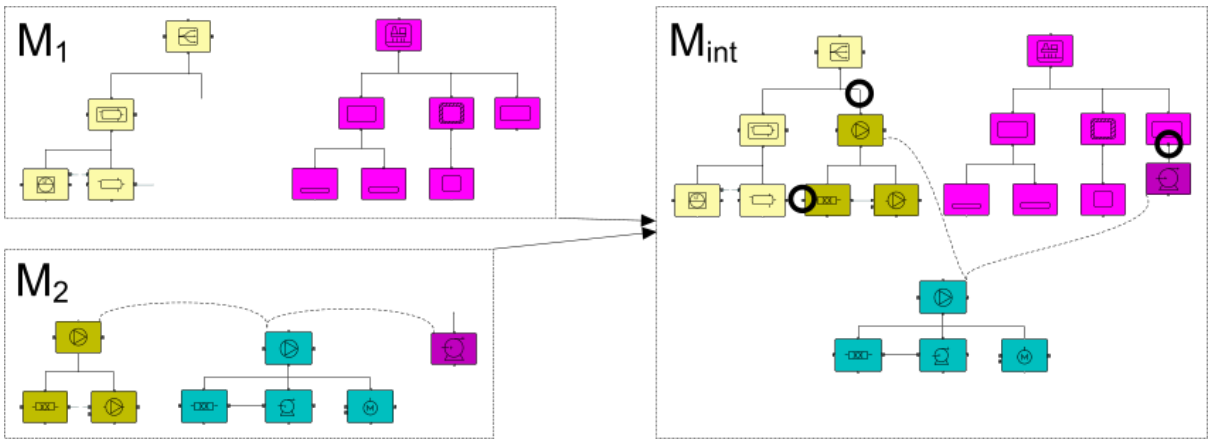


Figure 5.13: Model integration

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 through appropriate modelling.

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 interest 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 5.14.

The Function aspect hierarchy defines the system break-down structure, and the streams define how the systems are connected together. Each of the Function Blocks with Terminals define the requirements to what they represent, e.g., the “Pumping” Function Block with Terminals 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 Block with Terminals, 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 a high level by one “Separation system” Function Block with Terminals. At this level the stated requirements could be limited to the required capacity to perform separation of crude oil of a specified quality.

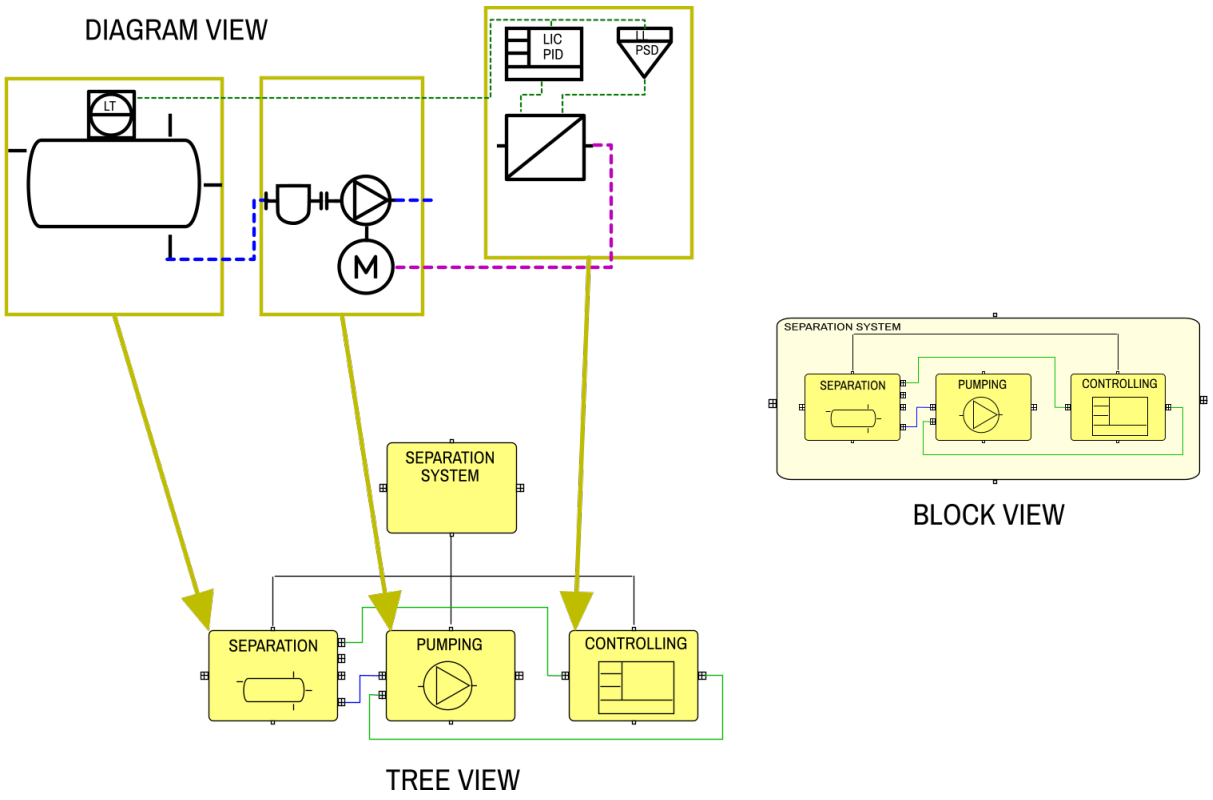


Figure 5.14: Separation system modelled in Function aspect

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 Block with Terminals.

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 5.15. No decisions regarding a solution specification are made, and therefore the entire model is in the Function aspect.

This particular requirement thread ties high level Process requirements to high level Electro requirements. Other requirement threads (not shown) similarly lead 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 invited.

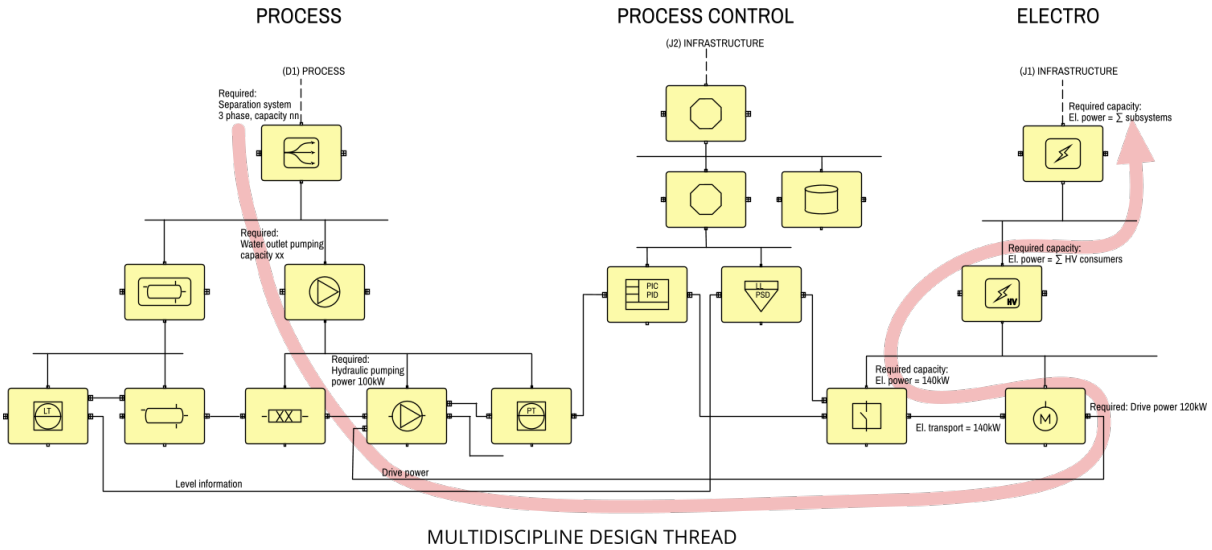


Figure 5.15: Modelling directed by the thread of requirements.

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 5.16.

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. Each part shown is represented by a Product Block with Terminals, 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 the 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 5.17 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 in such a document format. The purpose of modelling this architecture in the Location aspect is to specify all such information in the context of the location.

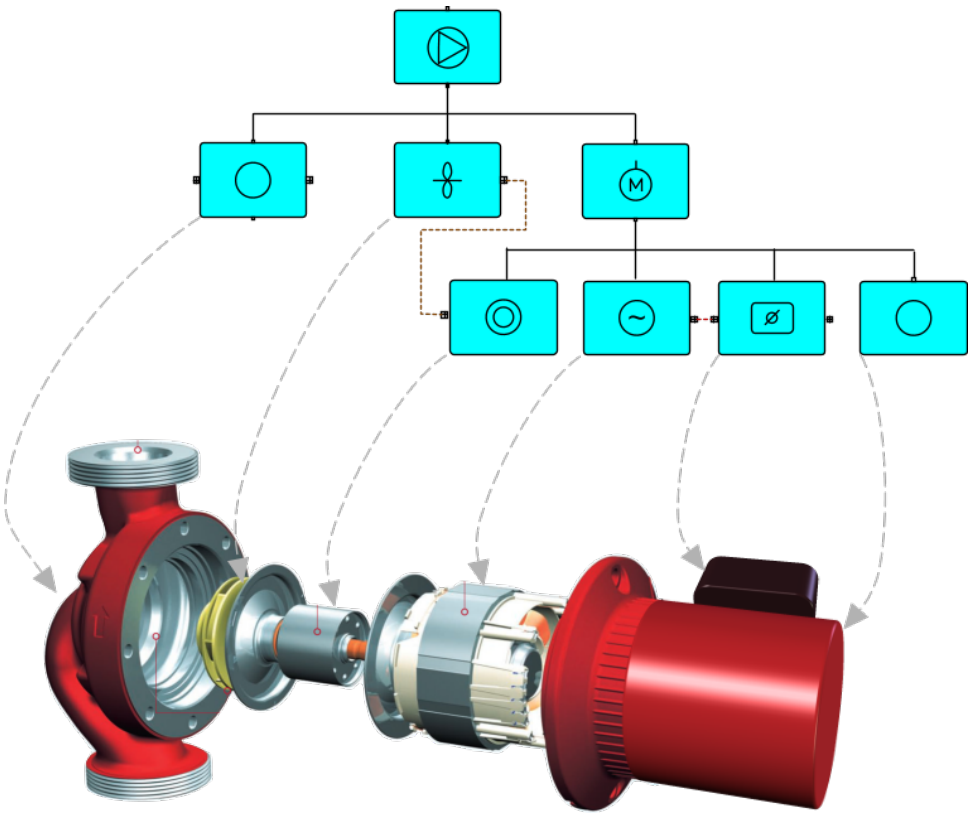


Figure 5.16: Pump Product model

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.

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.

5.9 Employing Re-usable Design Patterns

Usually, the design of industrial facilities is done by configuring a selection of pre-existing solutions, or *Design Patterns*. Whenever it is an option to re-use pre-existing designs, there is a need to do this as efficiently as possible. To accommodate this, IMF Complex Types offers a means to encode such typical Design Patterns in the form of a template - at the level of granularity and detail desired.

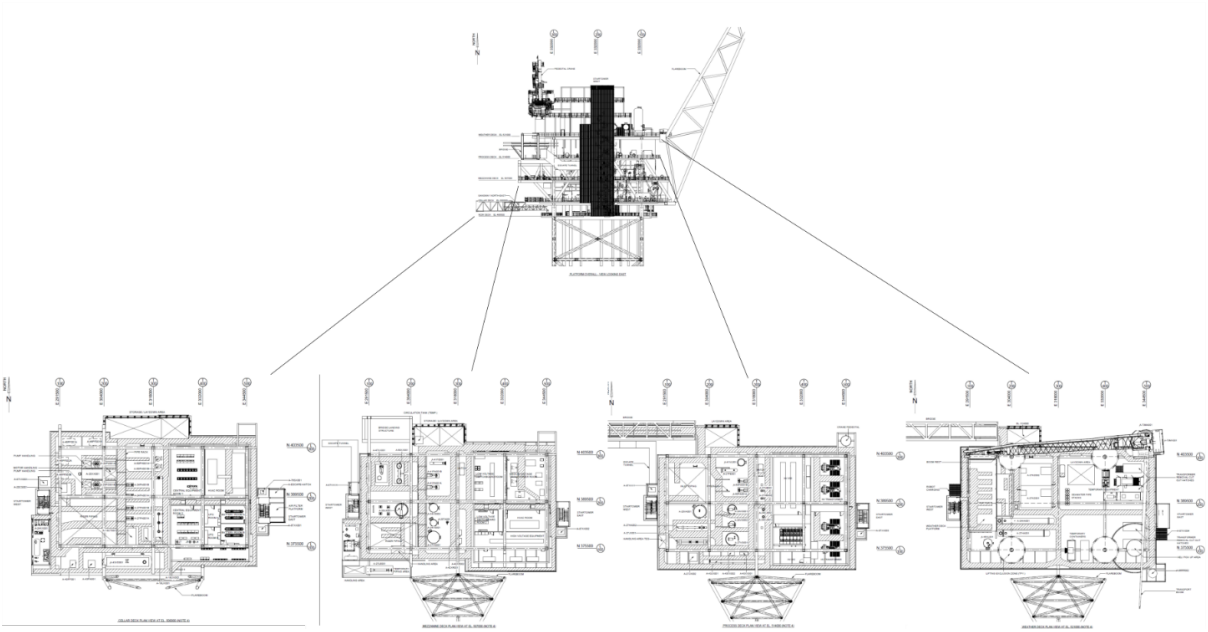


Figure 5.17: Plot Plans for a Platform

5.9.1 Design Patterns in Engineering

Design Patterns in engineering refers to a typical design which is likely to be re-used. It may have emerged as a de facto typical resulting from repeated use within a particular domain, or it may be more formalized such as being specified in a standard. It may be typical across the industry, or it may be a proprietary design re-used within a company. It may be complex and large, such as a Gas Compression Package, or it may be very limited in scope, such as an Electrically Driven Pump. In most cases it is valuable to include more than one Aspect to fully describe a Design Pattern. Also, it may be valuable to specify the break-down structure within each Aspect. As such a IMF Complex Type may appear similar to a small Facility Asset Model. However, a Design Pattern is not a *copy* of a previous design, instead it is a description of that which is repeated across several previous and similar designs, thus emerging as a typical. To achieve such a description a break-down structure may need to be included, but the individual elements in the break-down structure may only require the attribute 'Purpose' to be specified. Optionally Type and Attribute information can be included for the elements in the break-down structure, or it can be deferred to a later stage, depending on the work process.

5.9.2 Design Patterns represented as IMF Complex Types

IMF Complex Types differ from (basic) IMF Types: they allow more than one Aspect to be specified, and they allow specifying a break-down structure for each of the Aspects included. As an example, the IMF Complex Type for the 'Electrically Driven Pump' as a Design Pattern could specify both the Function and Product Aspects, with the main Purpose, Attributes and Terminals, and furthermore specify the break-down into Electric Motor, Shaft& Coupling, and Pump Unit - with the Purpose= Drive, Transport, and Pump, respectively. This would reflect the general Design Pattern that is used every time an 'Electrically Driven Pump' is incorporated into an IMF Model but would not include specifics related to the particular application.

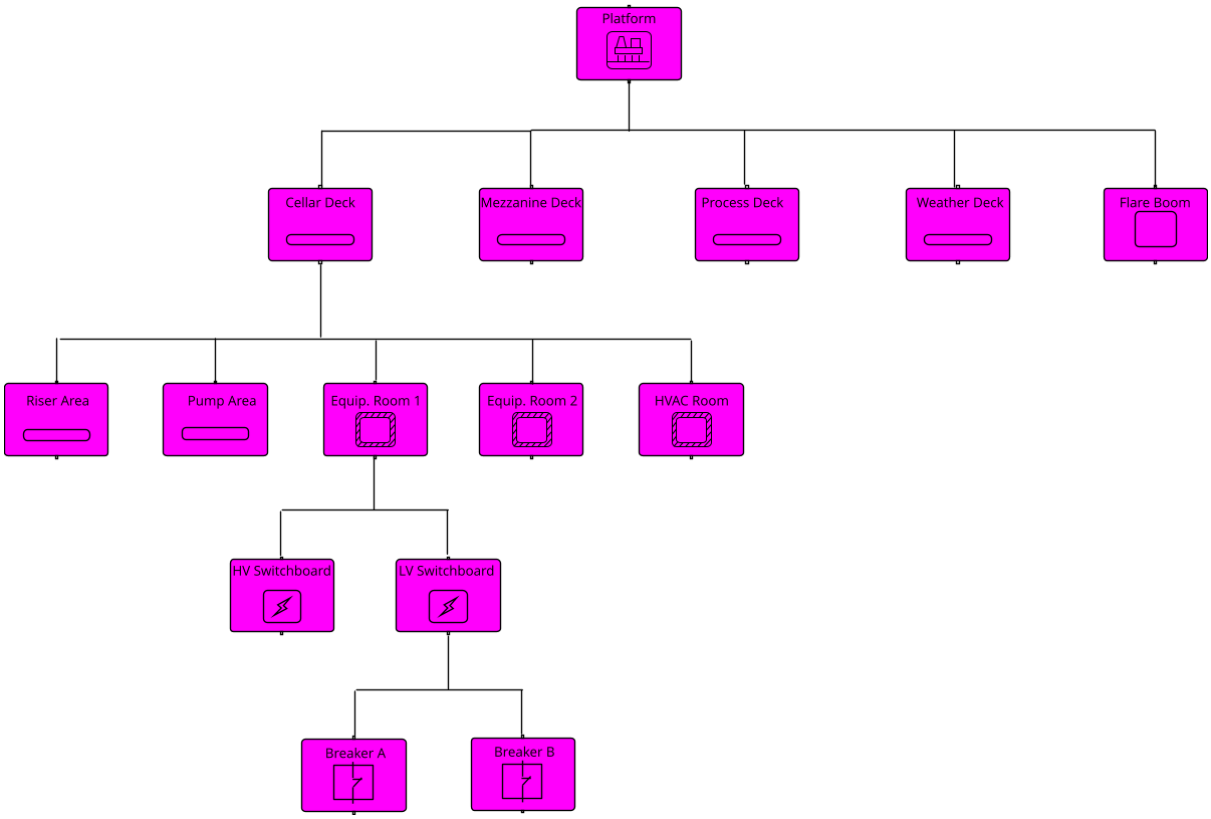


Figure 5.18: Model in Location aspect

5.9.3 Modelling Using IMF Complex Types

When modelling a facility asset where a part of the model can be created by re-using an established Design Pattern, this is done by creating an instance of the Design Pattern from a template; an IMF Complex Type. These are fetched from a library, or more precisely they are created from definitions called IMF Complex Types that reside in the IMF Type library. When such a Design Pattern is created it defines a minimum of terminals, attributes, break-down structures, and possibly also some topology, but the full details related to the particular application are not defined. These details may be completed at this stage or be deferred to later stages in the design process. The Design Pattern must then be tied into the model breakdown hierarchy where appropriate, and be connected between terminals, such that it becomes an integrated part of the overall model.

5.9.4 Deferred setting of attributes

To defer or postpone the setting of attributes can be a powerful feature when modelling. It allows deciding break-down structure before details of the individual elements in the break-down structure are known. The minimum information can be limited to stating the Purpose of each Block with Terminals and how they are related in a break-down hierarchy. This can be progressively enriched at later stages by deciding which IMF Type each Aspect Object shall be, followed by deciding on Terminals, Attributes, and topology information. The break-down structure can also be further developed into a higher level of granularity as design progresses.

Chapter 6

How to Specify IMF Types

This chapter gives an introduction and step by step guidelines on 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 The need for IMF Types

From an engineers perspective, the following is a valid questioning:

- What type of equipment is this? It is a pump.
- What type of pump is it? It is a centrifugal pump.
- What type of centrifugal pump is is it? It is a two-stage centrifugal pump.
- What type of two-stage centrifugal pump is it? It is a flange mounted two-stage centrifugal pump.

This questioning could go on, resulting in an ever increasing granularity of typing. As the granularity increases, the level of detail, as well as the precision increases. But the range of application decreases. In practical application, there is a need for both high level IMF Types to cover a wide range of use cases, as well as more detailed IMF Types for use cases that are more specialized. The engineer needs to be able to use IMF Types to state 'what type of', and for this the IMF Type must have a set of attributes that together characterizes 'this type of', but a further need is for the IMF Type also to be a template holding information which will be re-used every time an instance is created (during modelling). Thus, some attributes need to be strict - defining the type, whereas other attributes are non-strict - convenient template information only.

6.2 What is an IMF Type?

IMF Types are blueprints for building blocks used to model a Facility Asset. An IMF Type is a definition from which Blocks with Terminals are created. Blocks with Terminals are the building blocks. IMF Types are *not* models, meaning they do not comprise blueprints for break-down structures. IMF *Complex* Types will instead serve that purpose.

6.2.1 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.

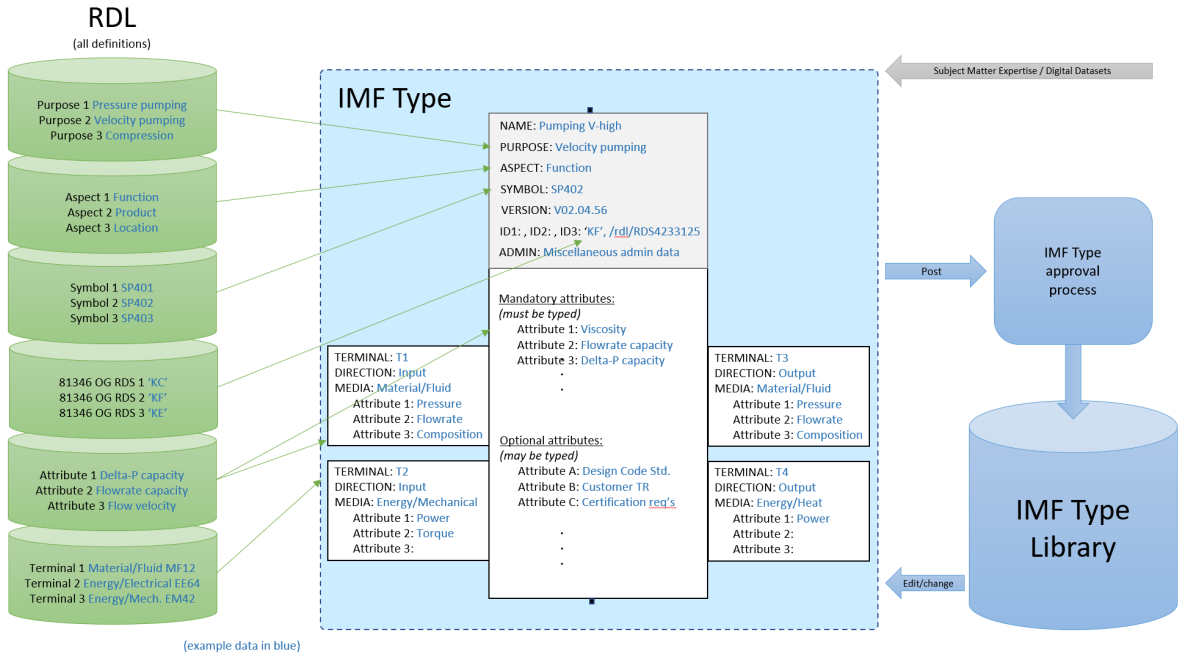


Figure 6.1: Creating and using a Block with Terminals from an IMF Type

6.2.2 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 Datasheets or conventional Datasheets.

6.2.3 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 is the chosen perspective. Therefore, Aspect Blocks 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.

6.2.4 IMF Type Attributes

An IMF Type comprises a range of attributes. These attributes are either 'Mandatory' or 'Optional'.

Mandatory Attributes Attributes that are essential to characterize the IMF Type. Data contents is a condition for later model validation. To understand which attributes should be mandatory, two guidelines apply: 1) Identify the attributes that are needed to make the 'Purpose' (i.e. the intended Activity) of the IMF Type as specific as possible. As an example, an IMF Type with Purpose= 'Pumping' will as a minimum need attributes about Flow and Delta Pressure, as this enables the 'Pumping' to be specified at a minimum level. 2) Identify the Mandatory Attributes of any similar IMF Types and make a comparison. There must be a difference in the set of Mandatory Attributes in order to represent a differentiation. As an example, if some other IMF Type, also with the Purpose= 'Pumping', also have the same Mandatory Attributes about Flow and Delta Pressure, then further differentiation is required. Assuming this example IMF Type is about variable speed pumping, then Minimum Speed and Maximum Speed would be appropriate extra Mandatory Attributes.

Optional Attributes Attributes that are *not* essential to characterize the IMF Type, but are nevertheless valuable. Typically they represent supplementary information, and can range from being links to reference documents - to being about particular detail data that is of interest when modelling a facility asset. Referring to the 'Pumping' example, an Optional Attribute could be Design Code or a Link to a Pump Curve document.

Additional attributes not part of the IMF Type, but added later, during modelling Attributes that are added to individual objects during modelling, but that do not have any relevance to the characteristics of the IMF Type from which the object originates. Such attributes can be freely added, as long as they are not in conflict with any of the Mandatory Attributes. One example, based on the imagined IMF Type for 'Pumping', could be Duty Configuration (e.g. '3x50 percent'), which is an attribute which is needed only when the pump is part of a set.

6.3 Creating an IMF Type

6.3.1 Target Group

The primary target group for creating IMF Types is the engineering domain SME. To be able to create types, the Reference Data needed must be available. 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.

6.3.2 Proactive versus Reactive Workflow

To maintain a comprehensive IMF Type Library requires 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.

6.3.3 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 represent. 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 component, equipment, or assembly.
- **In Location:** In this Aspect the spatial attributes are defined, including dimensions, relative location, and conditions and requirements within the space.
- **In Installed:** This Aspect is the recorded/documented attributes of a manufactured, delivered or installed component, equipment, or assembly.

6.3.4 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', i.e., to force an increase flow of-, and/or pressure of a fluid. 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 that require input of mechanical energy.

- **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 when meaning 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'.

Note: With the aspect 'Location' we here refer to the spatial perspective which can be measured along X, Y, Z, e.g., Rooms and Areas. A different aspect is the 'Logical Location' which is not defined in this document version. It represents a logical ordering, with a break-down structure that is not identical to the

physical break-down, and where X, Y, Z coordinates are not necessarily relevant. This could be used to define zones having defined activity spaces or states. Examples are Fire Detection, Wifi Coverage, etc.

- **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 'Drive' for an electrical motor.

6.3.5 Defining Attributes

One special property of an IMF Type is 'Purpose'. The Purpose is a verb which describes the essence of the type, such as 'Pumping', but alone it does not provide a sufficient specification, which is why Mandatory 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. Optional Attributes may also be defined as needed, typically to support data that will be repeatedly entered for objects based an IMF Type, even if it is not essential information. Attributes are selected from the Reference Data library.

6.3.6 Assigning the IMF Type to a Class

An IMF Type definition is a specialization of a more general IMF Type - 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 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 – hierarchy - 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 to state that 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.

6.3.7 Defining Terminals

Terminals define what is inputting to- and outputting from an Aspect Block. The main properties for a Terminal are direction (Input, Output, Nodirection) 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, electrical current, power.

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 needed.

6.3.8 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.

Mandatory 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 Mandatory Attributes such as ‘Flow, volumetric’, ‘Delta Pressure’, etc.

Optional 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 optional attribute:

- Client requirements such as a specific Technical Specification which states that there shall be a 50 percent 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.

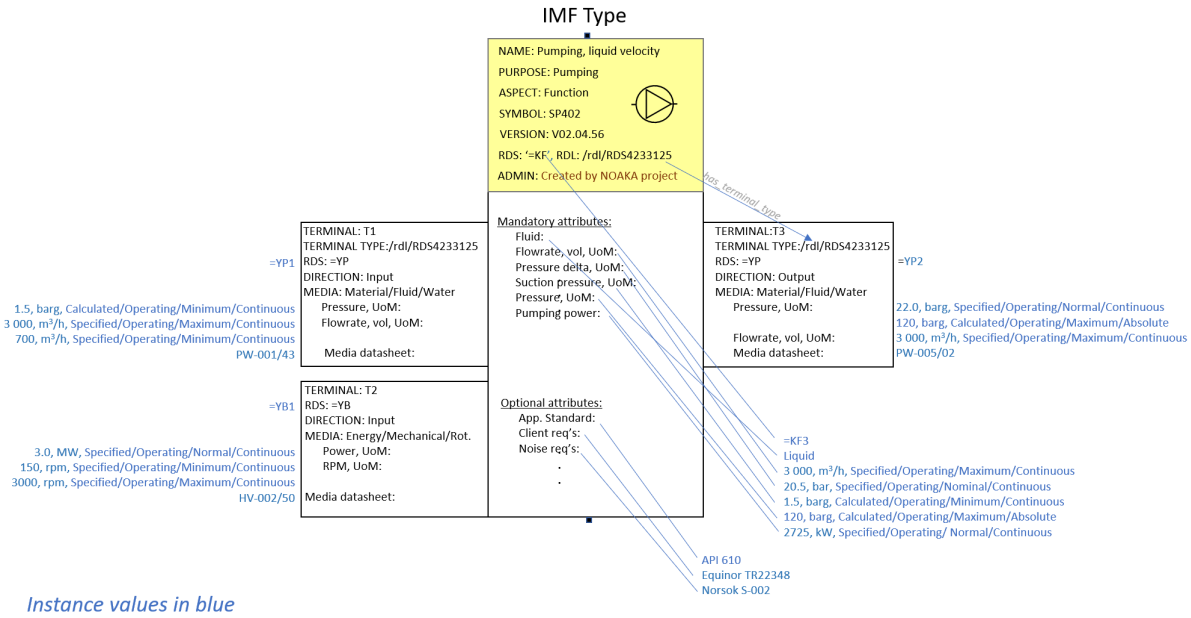


Figure 6.2: IMF Type example – Function aspect

6.3.9 IMF Type in the Product Aspect

In this Aspect it is specified how to solve and fulfill a need by means of a component, equipment, or assembly. Mandatory 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 Mandatory Attributes such as ‘Flow capacity’, ‘Pumping Power’, etc.

Optional Attributes are for specifying needs that are not directly related to the Purpose.

Example(s) of Optional Attributes:

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

Terminals in the Product aspect specify the *physical connection* 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 an 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, and would include attributes about flange size, cross section, etc.

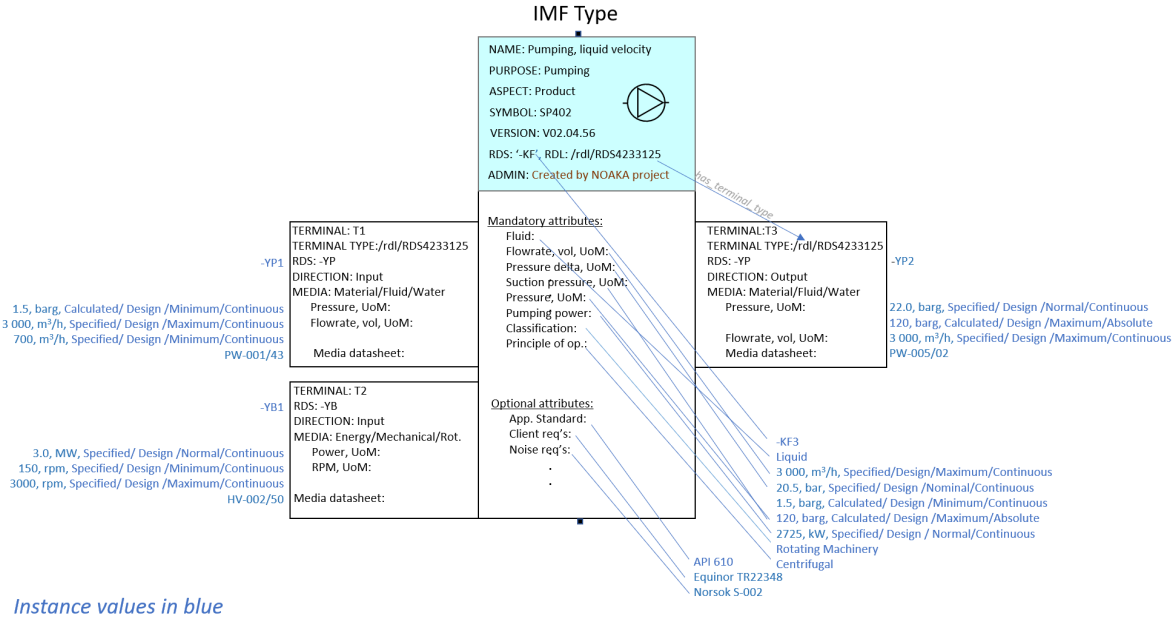


Figure 6.3: IMF Type example - Product aspect

6.3.10 IMF Type in the Location Aspect

In the Location Aspect the spatial properties are defined, including dimensions, relative location, conditions and requirements within the subject area. Mandatory 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.

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

Example(s) of area properties:

- Noise requirements that will apply to the emission of noise from any component located in this location.

In the Location aspect Terminals are not used.

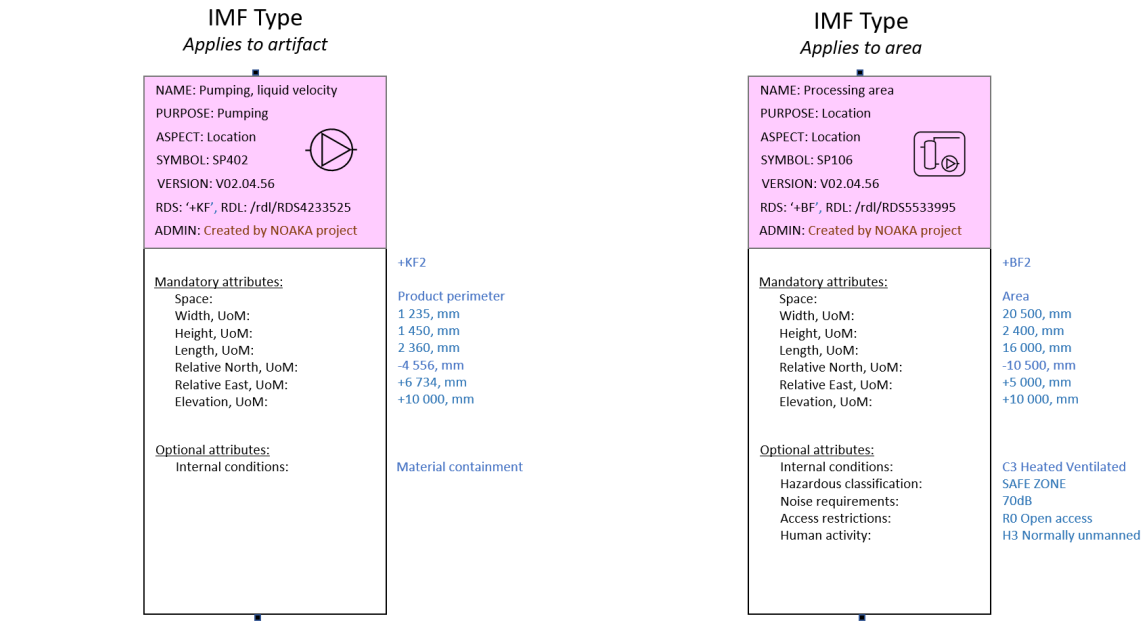


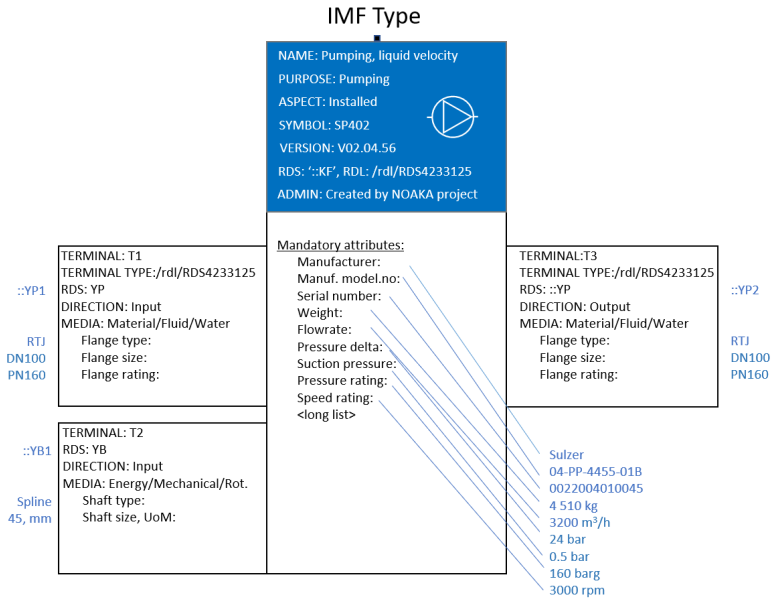
Figure 6.4: IMF Type example – Location aspect

6.3.11 IMF Type in the Installed Aspect

This Aspect is the manufactured, delivered or installed component, equipment, or assembly, with its recorded/documented properties. The Mandatory 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 physical connection of inputs to- or outputs from the delivered or installed product.



Instance values in blue

Figure 6.5: IMF Type example – Installed aspect

6.4 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 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.

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 different 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 the IMF language, which can be transformed into an RDF data set according the PCA RDL ontologies without losing any information.

6.5 [EXPERIMENTAL] Open versus Closed IMF Type

The default is that an IMF Type definition is *Open*. That is to say that an object instantiated from it can be freely extended during modelling by Additional Attributes or by increasing number of Terminals, the only

restriction being that the Additional Attributes must not be in conflict with the Mandatory Attributes. Even this limited restriction must be enforced by the user, as there are no mechanisms to prevent it, or invalidate it. With an IMF Type which is instead *Closed*, restrictions can be set for what are formally allowed additions, and underlying mechanisms will enforce these restrictions. The result would be a stronger type, with less variance allowed.

6.6 [EXPERIMENTAL] What is an IMF Complex Type?

Whereas the IMF Type provides a template for *one* aspect, and for *one* object, the IMF Complex Type provides a template for *several* aspects, breakdown structure(s) of *several* objects, and information about how they are interconnected (topology). As such, the IMF Complex Type can be likened with a small IMF facility asset model.

6.6.1 IMF Complex Type and its Role in Information Modelling

The use case for IMF Complex Types is to encode design patterns that are frequently employed when building IMF facility asset models. The IMF Complex Type provides a template for creating instances of such design patterns or even subsystems, and thus support efficient re-use of standardised design patterns.

6.6.2 IMF Complex Type Information Content

To understand what is the information content of an IMF Complex Type, it is helpful to think of it as a more or less empty model to be later filled in with more details. It comprises break-down structure(s) of IMF Types. It must include this breakdown structure for all the relevant aspects. It may also include information about topology. Further information is optional. Defining more details serve to reduce variance in how a design pattern is implemented, but will also reduce flexibility during modelling.

Chapter 7

The IMF Language Formalized

The formal IMF Language is defined using semantic web technologies. These artefacts are published at http://ns.imfid.org.

The terms of the language are defined by a Web Ontology Language (OWL) ontology, called the IMF Ontology. The ontology includes all the terms of the IMF Language described with different metadata, including a textual definition, and their formal semantics which details how a term is related to other terms in the language. The grammar of the IMF Language is specified by a set of SHACL Shape specifications. They describe the required and permissible ways terms in the IMF Ontology may be used.

The normative description of the IMF Language is found in the OWL ontology and SHACL Shapes published at http://ns.imfid.org. An informative extract from these artefacts is given in Section 7.1. The formalisation of IMF Types are presented in Section 7.2.

7.1 IMF Ontology

The IMF Ontology defines the vocabulary for creating IMF models, containing classes such as [Block], [Terminal], [Aspect] and [Attribute]; and properties such as [part of] and [connected to].

The ontology is designed to meet the following functional requirements:

- The users of IMF should be subject matter experts (SME). This means that the IMF language and ontology must be easy to learn and simple to use, hence it must contain a limited set of vocabulary terms and grammar rules.
- The language must provide incremental value, and hence must support modelling in an incremental manner.
- The language must 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 SMEs to express different levels of precision as part of iterations of their design.
  - Enable use in different phases of a project (concept, detail engineering, manufacturing, etc.).
- The language must have the precision of machine interpretation to allow automated verification. This means that the IMF language must be precise and unambiguous, and allow translation into other ontology languages and be able to exploit semantic descriptions in external ontologies and reference data libraries.

The use of this ontology may be validated against the following SHACL shape specifications:

- IMF Terms[1], specifies all the IRIs in the IMF namespace that is defined by the IMF Ontology. Use this to ensure that you are using the correct IRIs for the terms in the IMF Ontology.
- IMF Model Grammar[2], specifies constraints on the use of the IMF Ontology. Use this to ensure that you are using the IMF Ontology grammatically correctly.

Terms in the ontology are created following these principles:

- All terms should validate against the SHACL shape description: http://ns.imfid.org/20230630/imf-ontology-grammar.shacl.ttl[3]
- All terms in the ontology are given a status using the `vs:term_status` property from the 'Term-centric Semantic Web Vocabulary Annotations' vocabulary[4] to indicate their level of maturity using one of the values 'unstable', 'testing', 'stable' and 'archaic'. In this version of the ontology the highest status is 'testing'. Some terms are 'unstable'; these terms also contain the word 'unstable' in their label to make this clear.
- All IRIs in the ontology are placed in the namespace `http://ns.imfid.org/imf`. The localnames of IRIs follow these naming conventions:
  - Classes use `UpperCamelCase`;
  - Properties use `lowerCamelCase`, with additional rules:
  - object properties between IMF Classes usually start with a verb, e.g., `has`,
  - object properties for reified properties start with `the`,
  - object properties to external libraries, datatype properties, and annotation properties do not start with a verb;
  - Individuals use `lowerCamelCase`.
- When referring to a term in the ontology, the following format is used: [prefLabel], where prefLabel is set by the property `skos:prefLabel`, e.g., we will write [Block] to refer to the class `imf:Block`.

The ontology makes use of these vocabularies for its definition:

- SKOS[5]
- PAV[6]
- VANN[7]
- Term-centric Semantic Web Vocabulary Annotations[8]

Versions of this ontology and other semantic web resources for IMF are published at an URL following this schema: `http://ns.imfid.org/[yyyy-mm-dd]/[resource-name]`.

The ontology defines the following terms, categorised by their OWL type:

Classes `imf:Aspect`, `imf:AspectElement`, `imf:Attribute`, `imf:AttributeQualifier`, `imf:Block`, `imf:BreakdownPoint`, `imf:ConnectionPoint`, `imf:Element`, `imf:FunctionBlock`, `imf:FunctionElement`, `imf:FunctionTerminal`, `imf:InputTerminal`, `imf:InstalledBlock`, `imf:InstalledElement`, `imf:InstalledTerminal`, `imf:LocationBlock`, `imf:LocationElement`, `imf:LocationTerminal`, `imf:Model`, `imf:OutputTerminal`, `imf:ProductBlock`,

[1] <http://ns.imfid.org/20230630/imf-terms-grammar.shacl.ttl>
[2] <http://ns.imfid.org/20230630/imf-model-grammar.shacl.ttl>
[3] <http://ns.imfid.org/20230630/imf-ontology-grammar.shacl.ttl>
[4] <https://www.w3.org/2003/06/sw-vocab-status/note.html>
[5] <https://www.w3.org/TR/skos-primer/>
[6] <https://pav-ontology.github.io/pav/>
[7] <https://vocab.org/vann/>
[8] <https://www.w3.org/2003/06/sw-vocab-status/note.html>

imf:ProductElement, imf:ProductTerminal, imf:ProvenanceQualifier, imf:RangeQualifier, imf:RegularityQualifier, imf:ScopeQualifier, imf:Terminal, imf:TerminalQualifier

Object properties imf:asFunction, imf:asInstalled, imf:asLocation, imf:asProduct, imf:associativeRelation, imf:classifier, imf:connectedTo, imf:externalReference, imf:hasAspect, imf:hasAttribute, imf:hasAttributeQualifier, imf:hasElement, imf:hasInputTerminal, imf:hasOutputTerminal, imf:hasPart, imf:hasTerminal, imf:hasTerminalQualifier, imf:hierarchicalRelation, imf:interAspectRelation, imf:intraAspectRelation, imf:medium, imf:partOf, imf:predicate, imf:purpose, imf:symbol, imf:theConnected, imf:theInput, imf:theOutput, imf:thePart, imf:theWhole, imf:uom

Data properties imf:value, skos:notation

Annotation properties imf:color, imf:prefix

Individuals imf:absoluteQualifier, imf:averageQualifier, imf:calculatedQualifier, imf:continuousQualifier, imf:designQualifier, imf:functionAspect, imf:inputFlow, imf:installedAspect, imf:locationAspect, imf:maximumQualifier, imf:measuredQualifier, imf:minimumQualifier, imf:nominalQualifier, imf:normalQualifier, imf:operatingQualifier, imf:outputFlow, imf:productAspect, imf:specifiedQualifier

The terms are presented below, categorised in informal groups according to topic:

- Model
- Generic relations
- Elements
- Attributes
- Aspects

Each group contains an informal diagram of the group's terms. Each term is presented with a selection of metadata, its basic OWL ontology definitions and the SHACL shape constraints defined for the term. The listing does not contain all OWL axioms and SHACL shape definitions as it is difficult to represent in condenced form, please consult the OWL ontology and SHACL shape description for all details.

7.1.1 Model

Model (unstable)

IRI imf:Model
Definition A [Model] is a collection of [Element]s.
Usage note A [Model] is a construct for organising [Element]s into an identified collection which is useful for, e.g., provenance, exchange and integration.
The set of [Element]s of a [Model] is expressed using the property [has Element]. A [Model] can contain any number of [Element]s.
[Model] has the status 'unstable'; more detailed use cases and experience is required to work out its details.
Type owl:Class
SHACL Definition

path	min	max	severity	message
imf:hasElement	1		sh:Warning	The Model contains no Elements.

has element (unstable)

IRI `imf:hasElement`
Definition [has element] is a relation from a [Model] *M* to an [Element] *E* to specify that *M* contains *E*.
Type `owl:ObjectProperty`
Domain `imf:Model`
Range `imf:Element`

7.1.2 Generic relations

associative relation

IRI `imf:associativeRelation`
Definition [associative relation] is a generic relation that relates resources in an associative (or non-hierarchical) structure.
Usage note This is a generic property that is not intended to be used directly, rather use one of its subproperties. Generic properties like this are introduced to add structure to the properties of the ontology, and to be able to express generic class constraints.
Type `owl:ObjectProperty`
Superproperty `skos:related`
Subproperties `imf:connectedTo`, `imf:hasTerminal`, `imf:theConnected`, `imf:thePart`, `imf:theWhole`

external reference

IRI `imf:externalReference`
Definition [external reference] is a generic relation that relates a resource *X* to a resource in an external ontology or reference data library to describe *X*.
Usage note This is a generic property that is not intended to be used directly, rather use one of its subproperties. Generic properties like this are introduced to add structure to the properties of the ontology, and to be able to express generic class constraints.
Type `owl:ObjectProperty`
Subproperties `imf:classifier`, `imf:medium`, `imf:predicate`, `imf:symbol`, `imf:uom`

hierarchical relation

IRI `imf:hierarchicalRelation`
Definition [hierarchical relation] is a generic relation that relates resources in a hierarchical or tree-shaped structure.
Usage note This is a generic property that is not intended to be used directly, rather use one of its subproperties. Generic properties like this are introduced to add structure to the properties of the ontology, and to be able to express generic class constraints.
Type `owl:IrreflexiveProperty`, `owl:ObjectProperty`
Superproperty `skos:semanticRelation`
Subproperties `imf:hasPart`, `imf:partOf`

7.1.3 Elements

Block

IRI `imf:Block`

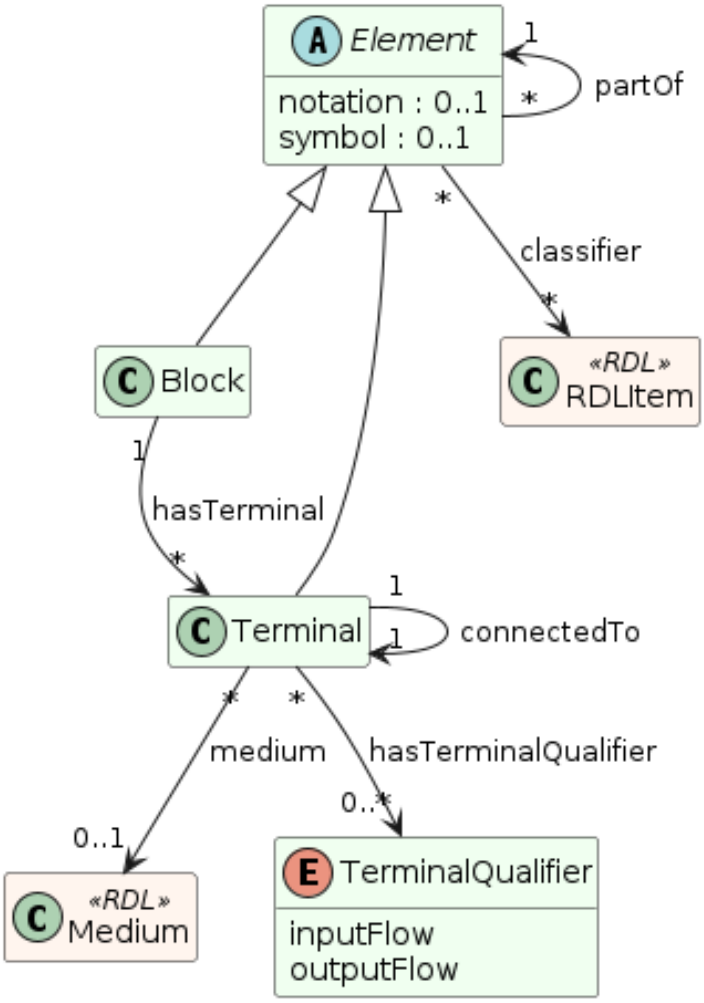


Figure 7.1: Ontology group: elements

Definition A [Block] represents an abstraction over a system or a system element as per ISO/IEC/IEEE 15288.

Usage note A [Block] is a basic building block of the IMF language. A [Block] can represent anything which is of interest and which is deemed convenient to treat as a system or system element. A [Block] sets the boundaries of what it abstracts over—at any granularity level. This could be a whole industry plant, a pump system, a measuring function, or a location of interest.

A [Block] interfaces with other [Block]s via its [Terminal]s (expressed with the property [has terminal]). A [Block] can have any number of [Terminal]s.

A [Block] is visualised as a rectangular box.

Type `owl:Class`

Superclass `imf:Element`

Subclass `imf:FunctionBlock`, `imf:InstalledBlock`, `imf:LocationBlock`, `imf:ProductBlock`

SHACL Definition

path	min	max	severity	message
imf:hasTerminal				The Block has a Terminal which is not a Terminal.
inverse of: imf:hasTerminal	1		sh:Warning	The Block has no Terminals.
imf:partOf				The Block has a part which is not a Block.
imf:hasPart				The Block is part of a non-Block.

Breakdown Point (unstable)

IRI `imf:BreakdownPoint`

Usage note A [Breakdown Point] represents a reified [has part]/[part of] property instance, using the properties [the whole] and [the part] to relate to the [Element]s that are related by the [has part]/[part of] property.

[Breakdown Point] has the status 'unstable'; its details are yet to be worked out.

Type `owl:Class`

Superclass `imf:Element`

SHACL Definition

path	min	max	severity	message
imf:thePart	1	1		The Breakdown Point must have exactly one part.
imf:theWhole	1	1		The Breakdown Point must have exactly one whole.

Connection Point (unstable)

IRI `imf:ConnectionPoint`

Usage note A [Connection Point] represents a reified [connected to] property instance, using the property [the connected] or subproperties [the input]/[the output] to relate to the [Terminal]s that are related by the [connected to] property.

[Connection Point] has the status 'unstable'; its details are yet to be worked out.

Type `owl:Class`

Superclass `imf:Element`

SHACL Definition

path	min	max	severity	message
imf:theInput		1		The Connection Point has more than one input.
imf:theOutput		1		The Connection Point has more than one output.
imf:theConnected		2		The Connection Point has more than two connected.

Element

IRI `imf:Element`

Definition An [Element] is a generic entity for modelling systems and system elements. An [Element] is described by its relations to other [Element]s, e.g., [part of] and [connected to], and by its [Attribute]s (through the property [has attribute]).

Usage note [Element] is a very generic concept and direct instantiation of [Element] is probably a mistake. A better option is to use a subclass of [Element] that specifies whether the [Element] is either a [Block] or [Terminal], and its [Aspect], e.g., [Function Block].

Note An [Element] should be expressed with the following metadata:

- a preferred label (using `skos:prefLabel`);
- optionally additional alternative labels (using `skos:altLabel`);
- a (textual) description (using `dc:description`);
- optionally a source of origin from which information about the element is taken (using `dc:source`);
- a version number (using `pav:version`);
- a created timestamp (using `pav:createdOn`);
- its creator, i.e., a person (using `createdBy`);
- optionally contributors to its creation (using `pav:contributedBy`);
- optionally the time of latest update (using `pav:lastUpdateOn`).

Type `owl:Class`

Subclass `imf:Block`, `imf:BreakdownPoint`, `imf:ConnectionPoint`, `imf:Terminal`

SHACL Definition

path	min	max	severity	message
inverse of: imf:hasElement	1		sh:Warning	The Element is not contained in a Model.
imf:purpose	1		sh:Warning	The Element has no purpose.
imf:partOf		1		The Element has more than one parent.

Input Terminal

IRI `imf:InputTerminal`

Definition An [Input Terminal] is a [Terminal] that accepts input (and not output).

Usage note An [Input Terminal] is equivalent to a [Terminal] that has the [Terminal Qualifier] [input flow].

Type `owl:Class`

Superclass `imf:Terminal`

Output Terminal

IRI `imf:OutputTerminal`

Definition An [Output Terminal] is a [Terminal] that accepts output (and not input).

Usage note An [Output Terminal] is equivalent to a [Terminal] that has the [Terminal Qualifier] [output flow].

Type `owl:Class`

Superclass `imf:Terminal`

Terminal

IRI `imf:Terminal`

Definition A [Terminal] is an [Element] that represents a point of interaction or communication for exactly one [Block] (through the relation [has terminal]), and hence specifies an input and/or output that the [Block] produces and/or receives. A [Terminal] may be qualified by [Terminal Qualifier]s. A [Terminal] is visualised as a square with rounded corners containing a plus sign, attached to its [Block]. An [Input Terminal] is placed to the left of its [Block], while an [Output Terminal] is

placed to the right of its [Block].

Type `owl:Class`

Superclass `imf:Element`

Subclass `imf:FunctionTerminal`, `imf:InputTerminal`, `imf:InstalledTerminal`, `imf:LocationTerminal`, `imf:OutputTerminal`, `imf:ProductTerminal`

SHACL Definition

path	min	max	severity	message
imf:partOf		1		The Terminal has more than one parent.
imf:connectedTo		1		The Terminal is connected to more than one Terminal.
imf:connectedTo	1		sh:Warning	The Terminal has no connection (to a different Terminal).
imf:medium		1		The Terminal has more than one medium.
imf:medium	1		sh:Warning	The Terminal has no medium.

Terminal Qualifier

IRI `imf:TerminalQualifier`

Definition [Terminal Qualifier] is a feature or characteristic of a [Terminal].

Type `owl:Class`

Instances `imf:inputFlow`, `imf:outputFlow`

classifier

IRI `imf:classifier`

Definition [classifier] is an [external reference] that relates an [Element] *E* to a resource *X* in an external ontology or reference data library such that *X* classifies *E* – or equivalently, *E* is an instance of *X*.

Type `owl:ObjectProperty`

Superproperty `imf:externalReference`

Subproperties `imf:purpose`

connected to

IRI `imf:connectedTo`

Definition [connected to] is an [associative relation] and an [intra-aspect relation] that relates a [Terminal] *T1* to a [Terminal] *T2* to specify that *T1* is connected to *T2*. [connected to] specifies a topology of [Block]s by relating the [Terminal]s of [Block]s.

Usage note [connected to] is used for describing how [Block]s are interconnected and indicates how they interact, i.e., how the [medium] of the connected [Terminal] flows between [Block]s.

[connected to] is a 1-1 relationship: a [Terminal] may only be [connected to] one other [Terminal], that is, a [Terminal] may not be [connected to] multiple [Terminal]s and it may not be [connected to] itself.

[connected to] is visualised with a line between the [Terminal]s.

Type `owl:FunctionalProperty`, `owl:InverseFunctionalProperty`, `owl:IrreflexiveProperty`, `owl:ObjectProperty`

Superproperty `imf:associativeRelation`, `imf:intraAspectRelation`

Domain `imf:Terminal`

Range `imf:Terminal`

has input terminal

IRI `imf:hasInputTerminal`

Definition [has input terminal] specialises the [has terminal] property to relate [Block]s to [InputTerminal]s.
Type owl:ObjectProperty
Superproperty imf:hasTerminal
Range imf:InputTerminal

has output terminal

IRI imf:hasOutputTerminal
Definition [has output terminal] specialises the [has terminal] property to relate [Block]s to [Output Terminal]s.
Type owl:ObjectProperty
Superproperty imf:hasTerminal
Range imf:OutputTerminal

has part

IRI imf:hasPart
Definition [has part] is a [hierarchical relation] and an [intra-aspect relation] that is the inverse relation of [part of]. See also [part of].
Usage note Use [has part] when you want to specify the [Element]s that are part of an [Element] *E* "on *E*", rather than using the inverse property [part of] to state that the [part of] relationship to *E* for every [Element] that is part of *E*.
Type owl:ObjectProperty
Superproperty imf:hierarchicalRelation, imf:intraAspectRelation, skos:narrower

has terminal

IRI imf:hasTerminal
Definition [has terminal] is an [associative relation] and an [intra-aspect relation] that relates a [Block] *B* to a [Terminal] *T* to specify that *T* is a terminal of, or belongs to, *B*. [has terminal] is an inverse functional property which means that a [Terminal] may only belong to one [Block].
Type owl:InverseFunctionalProperty, owl:ObjectProperty
Superproperty imf:associativeRelation, imf:intraAspectRelation
Subproperties imf:hasInputTerminal, imf:hasOutputTerminal
Domain imf:Block
Range imf:Terminal

has terminal qualifier

IRI imf:hasTerminalQualifier
Definition [has terminal qualifier] relates a [Terminal] *T* to a [Terminal Qualifier] *Q* to specify that *T* has the qualifier *Q*.
Type owl:ObjectProperty
Domain imf:Terminal
Range imf:TerminalQualifier

input flow

IRI imf:inputFlow

Definition [input flow] is a [Terminal Qualifier] that indicates that the flow is incoming to the [Terminal] it qualifies.
Type `imf:TerminalQualifier`, `owl:NamedIndividual`

medium

IRI `imf:medium`
Definition [medium] is an [external reference] that relates a [Terminal] *T* to a resource *X* to specify that *X* is the medium that flows through *T*.
Example [medium] is typically used to relate a [Terminal] to a resource that specifies one of the following:

- Material (Fluid, Solids),
- Energy (Mechanical, Electrical, Thermal),
- Force (Mechanical),
- Information(Electrical, Optical, Wireless)

Type `owl:ObjectProperty`
Superproperty `imf:externalReference`
Domain `imf:Terminal`

output flow

IRI `imf:outputFlow`
Definition [output flow] is a [Terminal Qualifier] that indicates that the flow is outgoing from the [Terminal] it qualifies.
Type `imf:TerminalQualifier`, `owl:NamedIndividual`

part of

IRI `imf:partOf`
Definition [part of] is a [hierarchical relation] and an [intra-aspect relation] that relates an [Element] *E1* and an [Element] *E2* to specify that *E1* is part of *E2*, or equivalently that *E2* has *E1* as a part. [part of] is used to specify a breakdown hierarchy of [Element]s.
Usage note [part of]/[has part] is used for describing an [Element] *E* by splitting *E* into parts *Es*, which again can be broken down into parts, to form a tree-shaped breakdown structure.
Formally this is expressed by specifying [part of] as a functional and irreflexive property. This means that any [Element] may only be part of one other [Element], that is: an [Element] may not be part of multiple [Element]s and it may not be part of itself.
[part of] is visualised with an arrow pointing from the child (the part) to the parent (the whole).
Type `owl:FunctionalProperty`, `owl:IrreflexiveProperty`, `owl:ObjectProperty`
Superproperty `imf:hierarchicalRelation`, `imf:intraAspectRelation`, `skos:broader`
Domain `imf:Element`
Range `imf:Element`

purpose

IRI `imf:purpose`
Definition [purpose] is a [classifier] that relates an [Element] *E* to a resource *X* in an external ontology or reference data library such that *E* has the purpose *X*.
Usage note [purpose] should be used to state the main purpose of an [Element]. Any additional purposes can be stated using [classifier].

Example Use [purpose] to state that the main activity of a [Function Block] is to perform pumping, by relating the [Function Block] to, e.g., the external resource PUMPING https://data.posccaesar.org/rdl/RDS9657917[9]:
`ex:myFunctionBlock imf:purpose <https://data.posccaesar.org/rdl/RDS9657917> .`
We then say that the [Function Block] has the purpose PUMPING. In the case that the [Block] also performs, e.g,. a mixing activity, this can be stated by relating the [Block] to an external resource that represents this activity using the property [classifier].
Type `owl:ObjectProperty`
Superproperty `imf:classifier`

the connected (unstable)

IRI `imf:theConnected`
Usage note See [Connection Point].
Type `owl:ObjectProperty`
Superproperty `imf:associativeRelation`, `imf:intraAspectRelation`
Subproperties `imf:theInput`, `imf:theOutput`
Domain `imf:ConnectionPoint`
Range `imf:Terminal`

the input (unstable)

IRI `imf:theInput`
Usage note See [Connection Point].
Type `owl:ObjectProperty`
Superproperty `imf:theConnected`
Range `imf:OutputTerminal`

the output (unstable)

IRI `imf:theOutput`
Usage note See [Connection Point].
Type `owl:ObjectProperty`
Superproperty `imf:theConnected`
Range `imf:InputTerminal`

the part (unstable)

IRI `imf:thePart`
Usage note See [Breakdown Point].
Type `owl:FunctionalProperty`, `owl:ObjectProperty`
Superproperty `imf:associativeRelation`, `imf:intraAspectRelation`
Domain `imf:BreakdownPoint`
Range `imf:Element`

the whole (unstable)

IRI `imf:theWhole`
Usage note See [Breakdown Point].

[9] <https://data.posccaesar.org/rdl/RDS9657917>

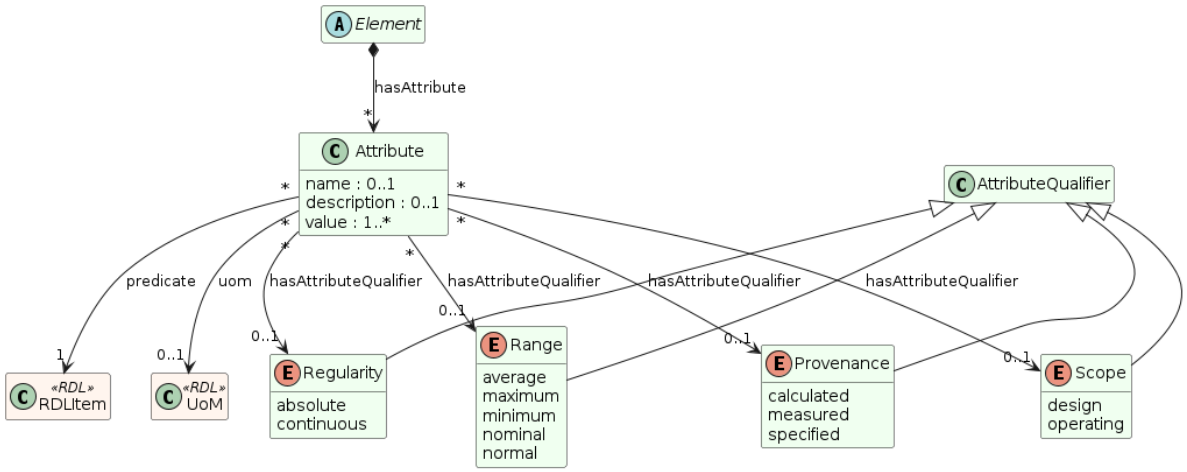


Figure 7.2: Ontology group: attributes

Type `owl:FunctionalProperty`, `owl:ObjectProperty`
Superproperty `imf:associativeRelation`, `imf:intraAspectRelation`
Domain `imf:BreakdownPoint`
Range `imf:Element`

7.1.4 Attributes

Attribute

IRI `imf:Attribute`
Definition An [Attribute] describes a quality, property or characteristic of a resource by setting a [predicate], a [value], and optionally a [unit of measure]. An [Attribute] may be qualified by [Attribute Qualifier]s.
Example Maximum ambient operating temperature: 120 K.
Measured weight: 1250 kg.
Specified color: red.
Type `owl:Class`
SHACL Definition

path	min	max	severity	message
imf:predicate	1	1		The Attribute must have exactly one predicate.
imf:value	1		sh:Warning	The Attribute has no value.
imf:uom		1		
imf:hasAttributeQualifier		1		The Attribute has more than one regularity qualifier.
imf:hasAttributeQualifier		1		The Attribute has more than one range qualifier.
imf:hasAttributeQualifier		1		The Attribute has more than one provenance qualifier.
imf:hasAttributeQualifier		1		The Attribute has more than one scope qualifier.
imf:hasAttributeQualifier	1		sh:Warning	The Attribute has no qualifier.

Attribute Qualifier

IRI `imf:AttributeQualifier`
Type `owl:Class`
Subclass `imf:ProvenanceQualifier`, `imf:RangeQualifier`, `imf:RegularityQualifier`, `imf:ScopeQualifier`

Provenance Qualifier

IRI imf:ProvenanceQualifier
Type owl:Class
Superclass imf:AttributeQualifier
Instances imf:calculatedQualifier, imf:measuredQualifier, imf:specifiedQualifier

Range Qualifier

IRI imf:RangeQualifier
Type owl:Class
Superclass imf:AttributeQualifier
Instances imf:averageQualifier, imf:maximumQualifier, imf:minimumQualifier, imf:nominal-Qualifier, imf:normalQualifier

Regularity Qualifier

IRI imf:RegularityQualifier
Type owl:Class
Superclass imf:AttributeQualifier
Instances imf:absoluteQualifier, imf:continuousQualifier

Scope Qualifier

IRI imf:ScopeQualifier
Type owl:Class
Superclass imf:AttributeQualifier
Instances imf:designQualifier, imf:operatingQualifier

absolute qualifier

IRI imf:absoluteQualifier
Type imf:RegularityQualifier, owl:NamedIndividual

average qualifier

IRI imf:averageQualifier
Type imf:RangeQualifier, owl:NamedIndividual

calculated qualifier

IRI imf:calculatedQualifier
Type imf:ProvenanceQualifier, owl:NamedIndividual

continuous qualifier

IRI imf:continuousQualifier
Type imf:RegularityQualifier, owl:NamedIndividual

design qualifier

IRI imf:designQualifier
Type imf:ScopeQualifier, owl:NamedIndividual

has attribute

IRI `imf:hasAttribute`
Definition [has attribute] is a relation between an [Element] *E* and an [Attribute] *A* that specifies that *E* has the attribute *A*.
Type `owl:ObjectProperty`
Domain `imf:Element`
Range `imf:Attribute`

has attribute qualifier

IRI `imf:hasAttributeQualifier`
Definition [has attribute qualifier] is a relation between an [Attribute] *A* and an [Attribute Qualifier] *Q* to specify that *A* is qualified by *Q*.
Type `owl:ObjectProperty`
Domain `imf:Attribute`
Range `imf:AttributeQualifier`

maximum qualifier

IRI `imf:maximumQualifier`
Type `imf:RangeQualifier`, `owl:NamedIndividual`

measured qualifier

IRI `imf:measuredQualifier`
Type `imf:ProvenanceQualifier`, `owl:NamedIndividual`

minimum qualifier

IRI `imf:minimumQualifier`
Type `imf:RangeQualifier`, `owl:NamedIndividual`

nominal qualifier

IRI `imf:nominalQualifier`
Type `imf:RangeQualifier`, `owl:NamedIndividual`

normal qualifier

IRI `imf:normalQualifier`
Type `imf:RangeQualifier`, `owl:NamedIndividual`

operating qualifier

IRI `imf:operatingQualifier`
Type `imf:ScopeQualifier`, `owl:NamedIndividual`

predicate

IRI `imf:predicate`
Definition [predicate] is an [external reference] that relates an [Attribute] *A* to a resource *X* to specify that *A* has the predicate *X*.

Example The [predicate] of an [Attribute] "Weight: 1250 kg" could be expressed as WEIGHT http://data.posccaesar.org/rdl/RDS356894[10].
Type `owl:ObjectProperty`
Superproperty `imf:externalReference`
Domain `imf:Attribute`

specified qualifier

IRI `imf:specifiedQualifier`
Type `imf:ProvenanceQualifier`, `owl:NamedIndividual`

unit of measure

IRI `imf:uom`
Definition [unit of measure] is an [external reference] that relates an [Attribute] *A* to a resource *X* to specify that *A* has the unit of measure *X*.
Example The [unit of measure] of an [Attribute] "Weight: 1250 kg" could be expressed as KILOGRAM http://data.posccaesar.org/rdl/RDS1328669[11].
Type `owl:ObjectProperty`
Superproperty `imf:externalReference`
Domain `imf:Attribute`

value

IRI `imf:value`
Definition [value] relates an [Attribute] *A* to literal value *v* to specify that *A* has the value *v*.
Example The [value] of an [Attribute] "Weight: 1250 kg" could be expressed as the literal `"1250"^^xsd:integer` (or using a different appropriate literal datatype).
Type `owl:DatatypeProperty`
Domain `imf:Attribute`

7.1.5 Aspects

Aspect

IRI `imf:Aspect`
Definition An [Aspect] describes a context for interpreting [Element]s. An [Aspect] specifies a *perspective*, an *interest*, and a *modality*.

- *Perspective* refers to from which viewpoint the [Element] is interpreted.
- *Interest* refers to the scope for which the information is intended used.
- *Modality* refers to the form in which information is recorded.

Example For examples, see the instances of [Aspect] defined in this ontology, e.g., [function aspect].
Note Aspect is a core concept of ISO/IEC 81346 and the IMF ontology includes aspects that are found in ISO/IEC 81346: function, location and product. We anticipate that more aspects will be added to support future modelling needs.
Type `owl:Class`
Instances `imf:functionAspect`, `imf:installedAspect`, `imf:locationAspect`, `imf:productAspect`

[10] <http://data.posccaesar.org/rdl/RDS356894>
[11] <http://data.posccaesar.org/rdl/RDS1328669>

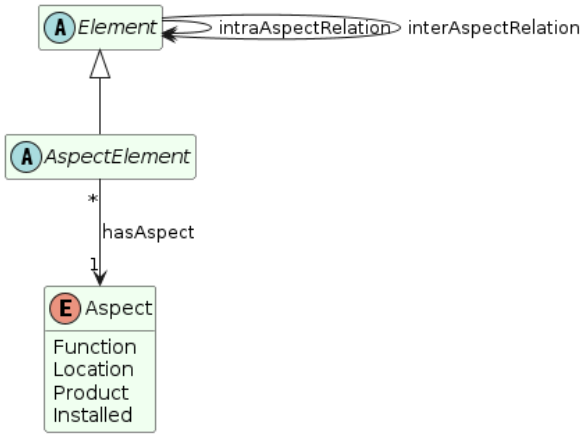


Figure 7.3: Ontology group: aspects

Aspect Element

IRI `imf:AspectElement`
Definition An [Aspect Element] is an [Element] that has exactly one [Aspect].
Usage note Avoid direct instantiation of [Aspect Element], use instead one of its subclasses.
The [Aspect] of an [Aspect Element] is visualised by filling the shape of the [Element] with the color of the [Aspect].
Type `owl:Class`
Subclass `imf:FunctionElement`, `imf:InstalledElement`, `imf:LocationElement`, `imf:Product-Element`
SHACL Definition

path	min	max	severity	message
imf:hasAspect	1	1		The AspectElement must have exactly one Aspect.

Function Block

IRI `imf:FunctionBlock`
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type `owl:Class`
Superclass `imf:Block`, `imf:FunctionElement`

Function Element

IRI `imf:FunctionElement`
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type `owl:Class`
Superclass `imf:AspectElement`
Subclass `imf:FunctionBlock`, `imf:FunctionTerminal`

Function Terminal

IRI `imf:FunctionTerminal`

Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:FunctionElement, imf:Terminal

Installed Block

IRI imf:InstalledBlock
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:Block, imf:InstalledElement

Installed Element

IRI imf:InstalledElement
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:AspectElement
Subclass imf:InstalledBlock, imf:InstalledTerminal

Installed Terminal

IRI imf:InstalledTerminal
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:InstalledElement, imf:Terminal

Location Block

IRI imf:LocationBlock
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:Block, imf:LocationElement

Location Element

IRI imf:LocationElement
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:AspectElement
Subclass imf:LocationBlock, imf:LocationTerminal

Location Terminal

IRI imf:LocationTerminal

Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:LocationElement, imf:Terminal

Product Block

IRI imf:ProductBlock
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:Block, imf:ProductElement

Product Element

IRI imf:ProductElement
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:AspectElement
Subclass imf:ProductBlock, imf:ProductTerminal

Product Terminal

IRI imf:ProductTerminal
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:Class
Superclass imf:ProductElement, imf:Terminal

as function

IRI imf:asFunction
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:ObjectProperty
Superproperty imf:interAspectRelation
Range imf:FunctionElement

as installed

IRI imf:asInstalled
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type owl:ObjectProperty
Superproperty imf:interAspectRelation
Range imf:InstalledElement

as location

IRI imf:asLocation

Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type `owl:ObjectProperty`
Superproperty `imf:interAspectRelation`
Range `imf:LocationElement`

as product

IRI `imf:asProduct`
Note This resource is programatically generated. Please consult its defining resources for textual descriptions.
Type `owl:ObjectProperty`
Superproperty `imf:interAspectRelation`
Range `imf:ProductElement`

color

IRI `imf:color`
Definition [color] is a relation from a resource *X* to a string identifying a hexadecimal color *c* to specify that the *c* is a color that is associated with *A*.
Usage note [Aspect]s are often associated with a distinct color, which is used in graphical presentations of IMF data.
Example For examples, see the instances of [Aspect] defined in this ontology, e.g., [function aspect].
Type `owl:AnnotationProperty`
Range `xsd:string`

function aspect

IRI `imf:functionAspect`
Definition [function aspect] is an [Aspect] about the intended activity of [Element]s, i.e., the activity an [Element] performs or is designed to bring about. [function aspect] has *perspective* "Activity", *interest* "System design", and *modality* "Specification".
Type `imf:Aspect`, `owl:NamedIndividual`

has aspect

IRI `imf:hasAspect`
Definition [has aspect] is a relation from an [Element] *E* to an [Aspect] *A* that specifies that *E* has the [Aspect] *A*.
Type `owl:ObjectProperty`
Domain `imf:Element`
Range `imf:Aspect`

installed aspect

IRI `imf:installedAspect`
Definition [installed aspect] is an [Aspect] about the information of [Element] instances. [installed aspect] has *perspective* "Artefact", *interest* "Built", and *modality* "Description of individual".
Type `imf:Aspect`, `owl:NamedIndividual`

inter-aspect relation

IRI `imf:interAspectRelation`
Definition [inter-aspect relation] a generic relation between [Element]s *E1* and *E2* such that *E1* and *E2* do not share any [Aspect]s.
Usage note [inter-aspect relations] is not used directly, rather use one of its subproperties. An [inter-aspect relation] is used to relate [Element]s that represent the same system/system element, but in different [Aspect]s.
Type `owl:ObjectProperty`
Subproperties `imf:asFunction`, `imf:asInstalled`, `imf:asLocation`, `imf:asProduct`
Domain `imf:Element`
Range `imf:Element`

intra-aspect relation

IRI `imf:intraAspectRelation`
Definition [intra-aspect relation] a generic relation between [Element]s *E1* and *E2* such that *E1* and *E2* share at least one [Aspect].
Usage note [intra-aspect relations] is not used directly, rather use one of its subproperties. An [intra-aspect relation] is used to relate [Element]s that represent different (but related) systems/system elements within the same [Aspect].
Type `owl:ObjectProperty`
Subproperties `imf:connectedTo`, `imf:hasPart`, `imf:hasTerminal`, `imf:partOf`, `imf:theConnected`, `imf:thePart`, `imf:theWhole`
Domain `imf:Element`
Range `imf:Element`

location aspect

IRI `imf:locationAspect`
Definition [location aspect] is an [Aspect] about the spatial envelope (e.g., geometry, size and shape) of [Element]s. [location aspect] has *perspective* "Location", *interest* "Geometry and Position", and *modality* "Specification".
Type `imf:Aspect`, `owl:NamedIndividual`

prefix

IRI `imf:prefix`
Definition [prefix] is a relation from an [Aspect] *A* to a string *s* to specify that *s* is the prefix associated with *A*. Such prefix strings are typically used for identifying the aspect when constructing reference designation system (RDS) identifiers.
Example For examples, see the instances of [Aspect] defined in this ontology, e.g., [function aspect].
Type `owl:AnnotationProperty`
Domain `imf:Aspect`
Range `xsd:string`

product aspect

IRI `imf:productAspect`

Definition [product aspect] is an [Aspect] about the specification of a (physical) solution/implementation of [Element]s. [product aspect] has *perspective* "Artefact", *interest* "Built", and *modality* "Specification".
Type `imf:Aspect`, `owl:NamedIndividual`

symbol

IRI `imf:symbol`
Definition [symbol] is an [external reference] that associates a resource *X* with a typical graphical representation of *X*.
Type `owl:ObjectProperty`
Superproperty `imf:externalReference`

7.2 IMF Types

An *IMF Type* is formally specified as a SHACL Shape constraint over the IMF Language as defined by the IMF Ontology. The IMF Type SHACL Shape should describe the required properties and values for properties using the constructs available in the SHACL language. This includes specifying, e.g., mandatory and optional properties, fixed values, and value ranges.

A rudimentary grammar, (also) in the form of a SHACL Shape constraint, for IMF Types is defined and made available at `http://ns.imfid.org`. The grammar extends the standardised SHACL shape specification published by W3C. The IMF Type grammar specifies three kinds of types: `BlockType`, `TerminalType` and `AttributeType`. IMF Types that follow the specified grammar can be translated to an OWL Class representation, and to prototypical instance data following the IMF Ontology. The translation tools are found from `http://ns.imfid.org`.

The OWL representation of an IMF Type represent the class of instances of the type, and instances of a type should explicitly be typed as member of this class. The IMF SHACL Shape for a type should use the same IRI as for the OWL class, but appended with `Shape`. Example: An IMF Type for a pumping function could be represented by an *SHACL shape* with IRI `ex:PumpingFunctionShape` and an *OWL Class* with IRI `ex:PumpingFunction`. An instance of the type, e.g,. `abc:myPumpingFunction123`, should be explicitly be typed as member of the OWL Class `ex:PumpingFunction` and it should validate against the SHACL Shape `ex:PumpingFunctionShape`.

The OWL representation of a IMF Type is useful for checking the consistency of the type, to classify instances according to types, and to discover specialisation/generalisation relationships between types by reasoning over a set of types in OWL format and the IMF Ontology.

The prototypical type instance generated from an IMF type can be useful as a starting point for specifying customised instances of the type.

To simplify the creation of IMF Types, a rudimentary tabular format for specifying IMF types is defined using OTTR templates and implemented as an annotated spreadsheet format. Figure 7.4 presents an overview of these OTTR templates. The contents of the spreadsheets in this format can be translated to SHACL Shape descriptions that follow the IMF Type Grammar. The OTTR templates are available at `http://ns.imfid.org`.

An example that demonstrates the use of all the IMF semantic resources is available from `http://ns.imfid.org`.

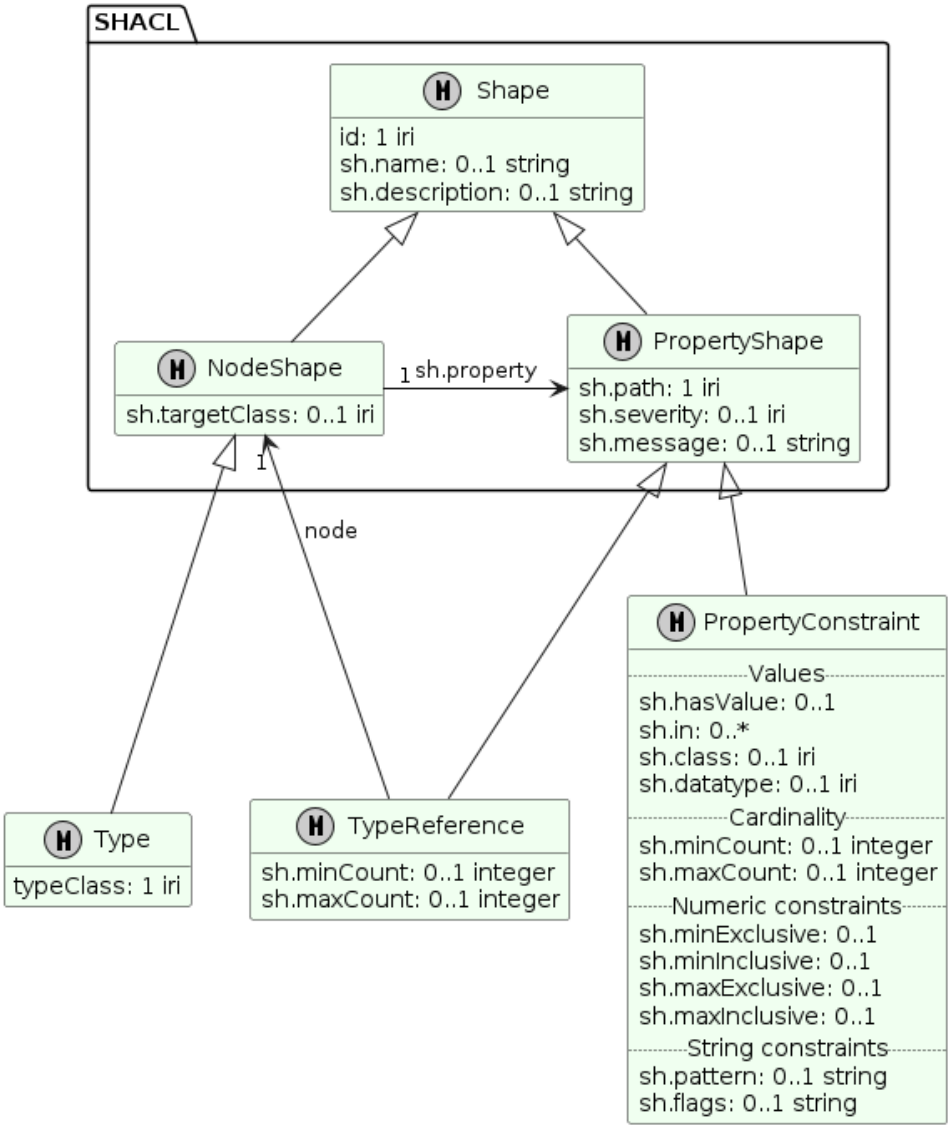


Figure 7.4: IMF Type OTTR Template patterns

Bibliography

[1] *Petroleum, petrochemical and natural gas industries - Collection and exchange of reliability and maintenance data for equipment*. Standard. International Organization for Standardization, 2016.

[2] *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*. Standard. International Organization for Standardization, 2004.

[3] *81346-1: Industrial systems, installations and equipment and industrial products - Structuring principles and reference designations - Part 1: Basic rules*. Standard. International Organization for Standardization/International Electrotechnical Commission, 2022.

[4] *'81346 - O&G - Reference Designation System for Oil and Gas*. Standard.

[5] *Systems and software engineering - System life cycle processes*. Standard. International Organization for Standardization/International Electrotechnical Commission, 2015.

[6] *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*. Standard. International Organization for Standardization.

Appendix A

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 granularity of the model, i.e. how far down into details the partOf breakdown is performed, shall be to the same level as the breakdown of the facility asset for the purpose of procurement or construction
7. 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.
8. The system information element shall include data about its purpose.
9. 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.
10. 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.
11. Descriptors and identifiers shall be included to: support interoperability with external systems, and user interaction

12. The interoperability shall as a minimum include support for loss-less transfer of Facility Asset information between applications
13. The Information Model shall include version control mechanisms to support engineering workflows such as review and revision control to support contractual obligations.
14. 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.

Appendix B

Changelog

The following changes have been made from draft version 2.0 released in November 2022 to address the comments received during the review process:

- Chapter 2 in this version covers what was Chapter 3 in the previous and includes also the problem statement and the material in the previous chapter on IMF eco-system.
- Chapter 3, previously Chapter 2, which includes background information on system, aspect, and information model, has been rewritten and extended.
- Chapter 4 is an engineering friendly extract of Chapter 4 of the previous version.
- Chapter 5, which describes how to model, has been updated.
  - Updated to reflect comments received from external draft review.
  - Updated to reflect IMF changes as specified in Chapter 7.
- Chapter 6, which describes how to create IMF Types has been updated.
  - Updated to reflect comments received from external draft review.
  - Updated to reflect IMF changes as specified in Chapter 7.
- Chapter 7, which includes the formal specification of the IMF language, has been restructured. The IMF ontology has been updated according to best practices for ontology engineering, including textual definitions for all central terms. The ontology has been updated to accommodate received comments; see below. To simplify the formal definition of the IMF language, the structural definition has been removed in favour of relying on only the semantic technologies OWL and SHACL.

  The following changes has been done to the formal specification of the IMF language:

  - The inter-aspect relation fulfilledBy has been replaced with multiple relations named as Function, asProduct, asLocation, asInstalled to make the relations easier to communicate and more aligned with the definitions. The definitions of the inter-aspect relations have been loosened to allow functional elements to be related to multiple products, which allows for more pragmatic modelling.
  - InterfacePoints have been taken out. We expect the functionality that interface points represented to be re-introduced in the next major release of the IMF manual.

  - BiTerminals have been taken out. Instead use Terminal, that is a Terminal with no specified direction. The requirements on Terminals with a direction have been loosened: the connectedTo relation is relation between Terminal, no further requirements for InputTerminals and OutputTerminals are specified.
  - connectedTo is only a binary relation (and not also a ternary relation)
  - The notion of reified relationships (BreakdownPoint and ConnectionPoint) is introduced as a holder for attributes for relationships.
  - partOf relationships are allowed also between Terminals, and not only Blocks.
  - The ontology contains a changelog of the IRI that have been removed and added between the previous version.
- The specification of IMF Types has been updated.

Appendix C

Way Forward

The following topics in the scope of IMF are not fully covered in this version, but are identified as future work:

- The interplay between breakdown and topology.
- Detailing a concept of type to apply to complex and composite structures such as packages and design codes, and the integration of such models into larger model structures.
- A visual syntax for types to extend Chapter 4.
- 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.
- Exchange formats for IMF data and types must be updated and aligned with ongoing tool development.
- Semantic verification, integrity checking and interpretation of IMF models by means of the Industrial Data Ontology.