Information Modelling Framework Manual

Version 0.3.0

2024-05-29

A Note on Versioning

The Information Modelling Framework (IMF) is and has been in development since the early 2020's. During this time the IMF Manual, of which this document is a version of, has been released in incremental versions. These versions have followed a loose versioning scheme.

We wish now to follow the semantic versioning scheme (`https://semver.org/`) and indicate that IMF is in rapid development by using semantic versioning's major version 0, e.g., `0.y.z`. This and all previous versions should therefore be considered as minor version of major version 0. Hence, older versions that do not have major version 0, should be prefixed with major version 0, e.g., the IMF Manual released 2023-07-05 has the semantic version 0.2.1.

We anticipate that version 1.0.0 will be released in the fall of 2024.

Outline and Readers Guide

This document introduces IMF and explains how IMF can be used to enable new ways of working for engineering of facility assets, from the design begins and until the facility asset 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 ecosystem 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 gives a complete overview of the IMF language.

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.

For intended users of IMF the entire document 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

Chapter 1

Scope

This document contains a specification of the Information Modelling Framework (IMF), 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 resources used when building IMF models.

❹ Semantic verification mechanisms and tools that offer computer-based validation, integrity checking, and verification of the IMF model.

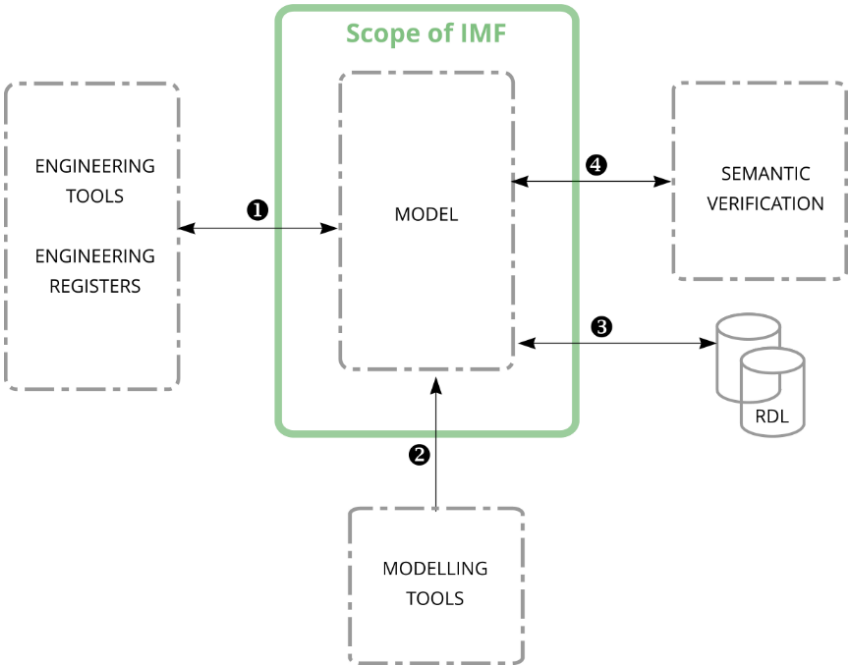


Figure 1.1: Framing and scoping of the IMF.

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 a modelling methodology.
- A definition of a methodology for creating or modifying modelling building blocks.
- Guidelines that enable subject matter experts (SMEs) to be users of the framework, supporting and improving existing discipline design workflows.
- 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.
- Open protocols for exchange of facility asset information and formats that enable automated verification and augmented engineering extensions.
- Examples of shared libraries and use of existing standards.
- Examples of engineering tools and engineering registers that may hold attributes and objects and describe the facility asset.

This document and the development of IMF is intended to serve as a reference for:

- Understanding how to implement a new way of working using information models.
- The basis for a DNV recommended practice on the subject of creating and using of industrial facility assets information models.
- Implementing tools and applications for information modelling.
- Alignment of industry reference data libraries.
- Implementation of cross-industry interoperability based on IEC/ISO 81346-1 [2] (structuring principles) and ISO 15926-14 [5] (reference data and verification).
- Computer-based exchange, automation, and verification of information models.

Chapter 2

A New Way of Working using Information Models

2.1 Problem Statement

Facility assets in the energy industry are becoming increasingly complex. There is a need for better communication between organisations, 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; and
- 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 to a portfolio of applications, and a lot of company-specific requirements that are tuned to fulfil needs for a certain portfolio of applications.

Figure 2.1 illustrates the problem of today's way of working. The figure shows 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.

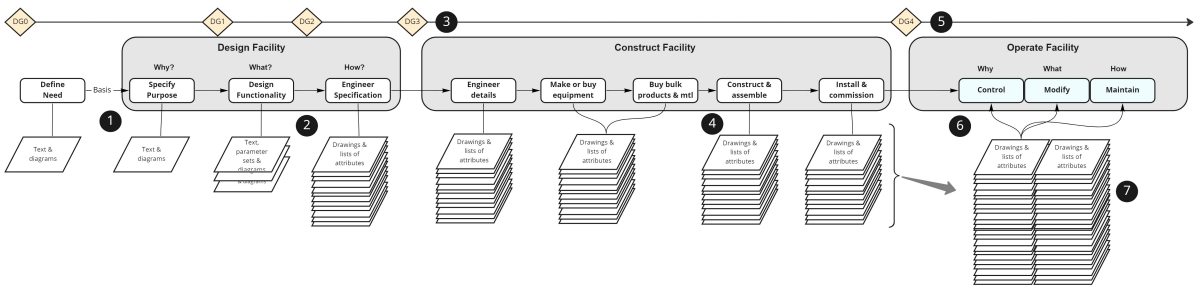


Figure 2.1: Current documentation practice during an industrial investment and development project.

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 specialised 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 labour-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 lacks 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.
- Utilisation 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.

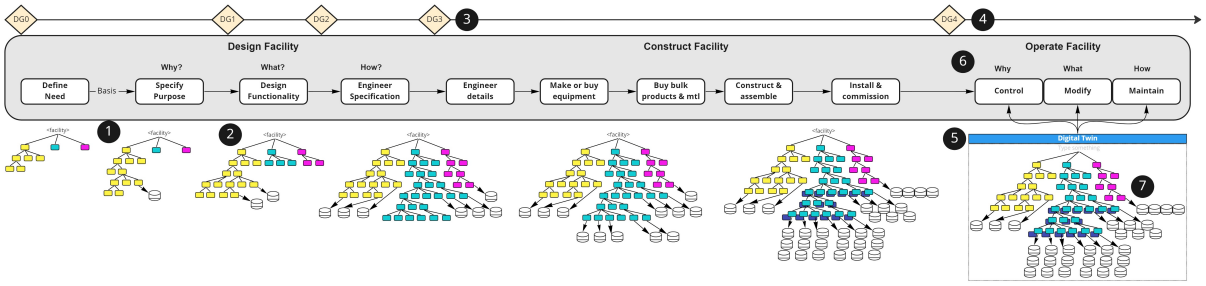


Figure 2.2: Using information models instead of documents during an industrial investment and development project.

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-models*, containing the historical context and all 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:

- 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 knowledge representation and semantic technology. For example, a knowledge engineer manages the validation and integrity verification of the model of a design.

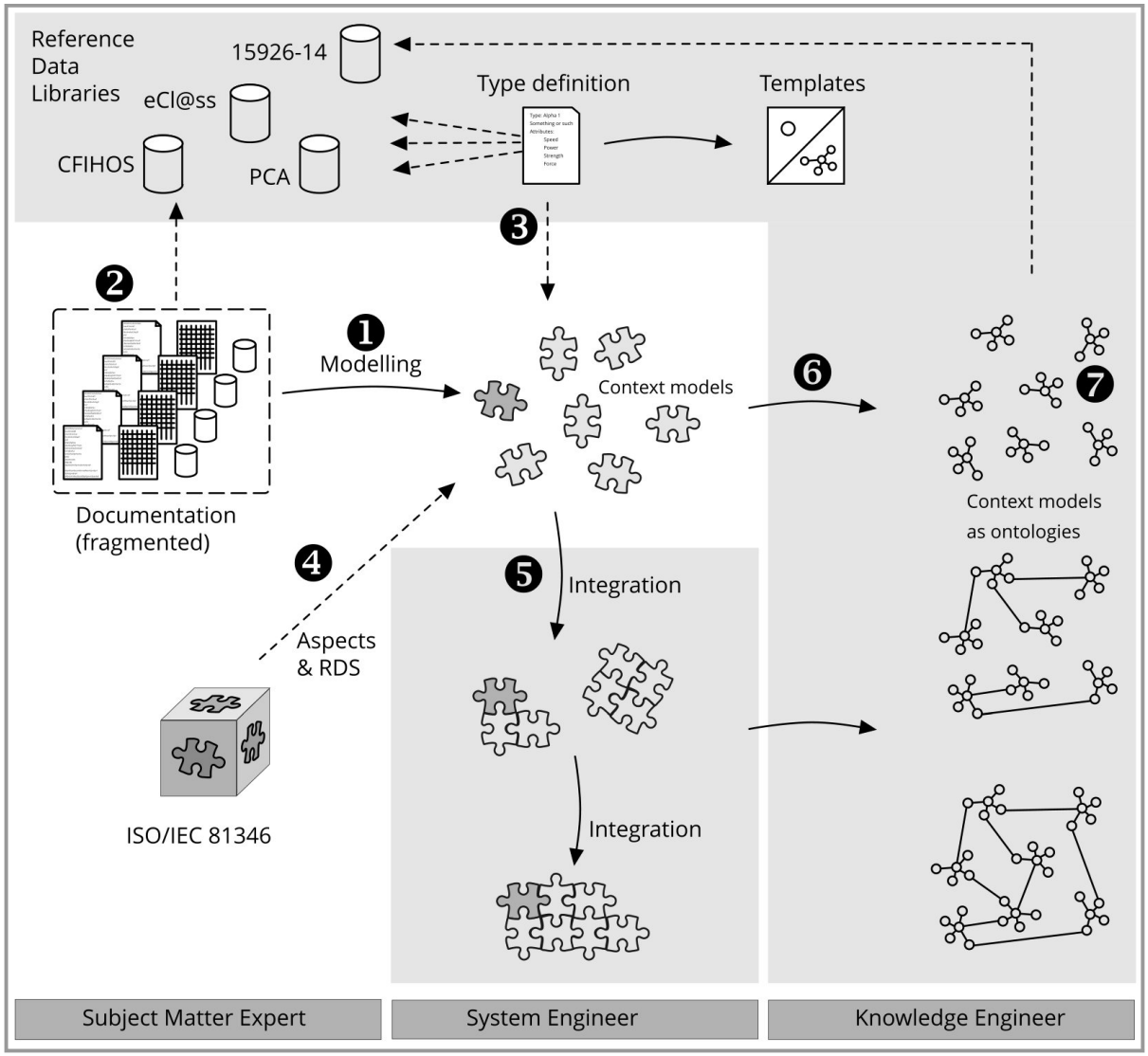


Figure 2.3: Different expertise and how they enable information modelling of facility assets.

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 the data engineer has conventionally 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 utilised to verify and validate the integrity of the model.

SMEs model information that today is available only in fragments, commonly recorded in 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 rules and structures of IMF that build on the IEC/ISO81346 O&G standard [3] ❹.

IMF does not require modelling to be done in one single model, something which often is implied when referring to data-centric or model-centric ways of working. Instead, the modelling can be distributed across 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 into semantically enriched models that enable the knowledge engineers to exploit verification techniques such as automated reasoning ❼.

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

Chapter 3

Fundamental Concepts for IMF

The chapter provides readers with an engineering background with an understanding of the IMF approach to asset modelling. Section 3.1 addresses the system concept. It identifies the need for additional structuring principles to support development and use of information models of engineered systems. Section 3.2 introduces the concept of aspect. In IMF aspects are used to express the context for modelling and to provide the structuring principles that give unambiguous system breakdowns and topologies. Section 3.3 introduces the fundamental building blocks for information models using IMF. Section 3.4 addresses different formal languages of IMF that are developed for different user groups and purposes.

3.1 System

IMF is a language for describing assets, and in particular for describing engineered systems. In the Systems Engineering standard ISO/IEC/IEEE 15288 a system is defined as a combination of interacting elements organised to achieve one or more stated purpose [4]. Figure 3.1 graphically illustrates this definition.

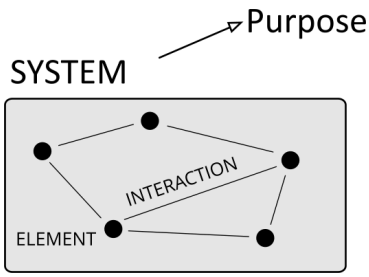


Figure 3.1: The system concept from ISO/IEC/IEEE 15288, illustrated as a group of elements interacting within a boundary to achieve a purpose.

As an example, the elements of a cooling system may be equipment such as pumps, pipes, chillers, etc. The purpose of the cooling system can be described as heat exchanging.

Below are observations about purpose, elements, and interaction, followed by an observation about the system concept as a structuring principle. These observations motivate key ideas behind IMF.

3.1.1 Purpose

A system purpose can be described by stating an activity that the system is designed to bring about. Closely related to purpose is the concept of a system function, understood as a capability of the system to realize its purpose. In IMF a system function is described indirectly by describing the activity in which the function is realized, ie. by stating system purpose.

An engineered system will typically have medium at input and output, where medium is understood to be either material, energy, force, or signal carrying information. Its purpose will be described as an activity that transforms input media state to output media state.

3.1.2 System elements

System elements can themselves be systems. In consequence, a 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*.

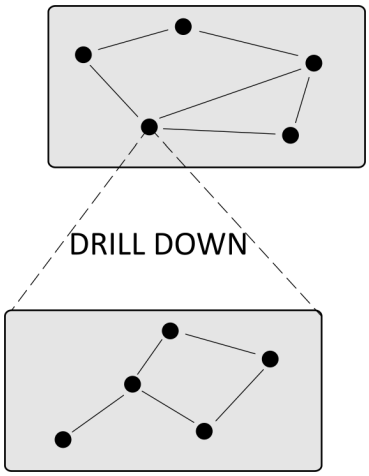


Figure 3.2: Illustration of a system breakdown

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, an engineering contractor may take the pump as an equipment 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.

3.1.3 System interaction

A system *topology* describes how system elements are interconnected and hence how they interact. In IMF it is assumed that system elements interact through media.

Different disciplines will typically have focus on 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. The functions provided by a complex system typically arise from interaction across different disciplines. Such systems can be broken down into sub-systems where the system elements interact through only one single media type. This situation is illustrated in Figure 3.3.

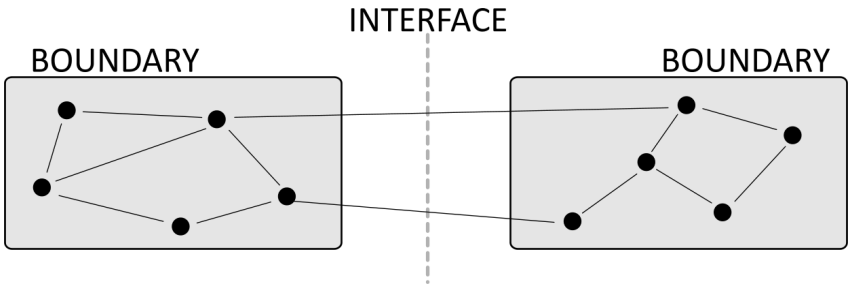


Figure 3.3: Illustration of the interaction of system elements inside and outside of system boundaries.

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.

The interface between two systems specifies how the systems interact. In IMF an interface is broken down to a set of connectors. Each connector details the flow of a medium between two elements, e.g., how much cooling medium the pump shall supply to the heat exchanger. When IMF is used to describe functional requirements to a system, information on a connector can be understood as a description of a contract between the systems elements. 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.

3.1.4 The system concept as a structuring principle

An information model that describes an engineered system must describe its purpose, identify the elements that lie within the system boundary, classify these elements, and describe how these elements interact with each other and with elements outside the system boundary for the system to achieve its purpose.

The system concept can be used as a structuring principle for developing information models of engineered systems. Repeated use of the principles of system breakdown and interaction, as a recursive scheme, gives rise to patterns recurring at progressively smaller scales both vertically (system breakdown) and horizontally (system interactions) as is illustrated in Figure 3.4.

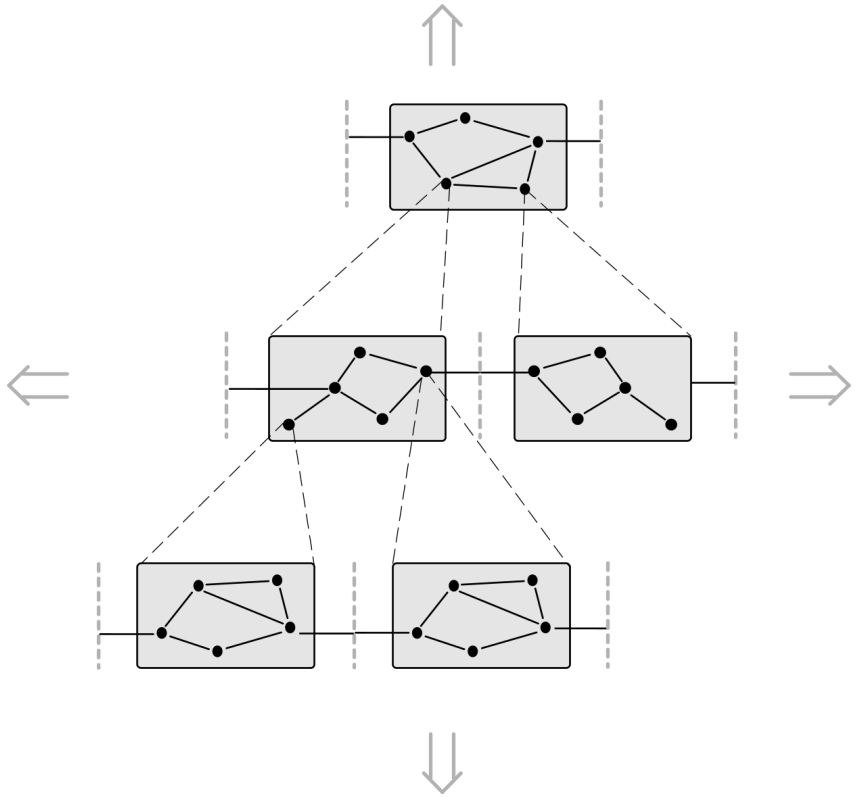


Figure 3.4: Illustration of system breakdown and topology.

Note that:

- A system may be broken down in more than one way, and hence give rise to different groups of system elements;
- A system may be a system element of more than one system, and hence be part of more than one breakdown,
- A system may be connected to other systems through more than one topology.

This ambiguity complicates both the design of information models, and access to technical information through them, since a user must manage different breakdowns and topologies without confusion.

Example: Consider a cooling system with, among other things, a pump connected to a heat exchanger. The cooling system can be drilled down in several ways:

- Functionally, the pump and the heat exchanger are both contributing to a function the cooling system and are hence contained in a function breakdown.
- The pump can be delivered on a skid as part of a pumping package and hence be part of a bill of material breakdown of the engineering implementation. The heat exchanger may be delivered in a different package.
- The pump can be located in an explosive area and is hence contained in an area breakdown.

There are several topologies in play in the cooling system, for instance:

- The cooling medium of a pump can be represented by a connector with information about the cooling medium.
- The terminal of the pump is functionally connected to the terminal of the heat exchanger.
- The terminal of the pump is physically connected to the pipes that transport the cooling medium through flanges that connect by means of mechanical force.

The IMF approach explained below is to use aspects to separate different ways of breaking down and connecting so that breakdown and connectivity are not allowed to cross aspect. Each element of a given aspect will then be part of a unique breakdown hierarchy. This gives the intuitive operations of zooming in and zooming out a precise meaning that can be implemented by traversing a breakdown hierarchy upwards or downwards. Also, a terminal of an aspect will be connected to a unique connector. Aspects can thus be used to resolve the ambiguity inherent in the system concept.

3.2 Aspect

IMF is a language that can be used to describe assets from different points of view. It is thus assumed that information about an asset is comprised of information from a multitude of views. An IMF model aims to preserve these views and put them in relationships, and thus build them into the structure of IMF information models.

3.2.1 Definition of aspect

An aspect is defined figuratively as a point of view. The first task in making an IMF model is to define the aspects that will be used in the model. The definition of an aspect has to answer three questions:

- What is the information domain of the view? There are three predefined information domains: Activity, Implementation, Space.
- What is the modelling interest? There two two predefined interests: Product Lifecycle and Project Lifecycle.
- What is the mode of existence of what is modelled? There are two predefined modalities: Intended and Actual.

The three dimensions in the definition of aspects are complementary. They are also extensible in the sense that more terms can be added when more detailed aspects are needed.

3.2.1.1 Information domain

Information domain refers to what is captured in a view. An information domain can be thought of as a filter on information. IMF has three predefined information domains and allows users to introduce more:

- *Activity*: Information about, e.g., process, motion, change. System purpose is meant to be captured in terms of the Activity information domain;

- *Implementation*: Information about physical objects or software. A system view of elements as physical objects or software can be described in terms of the Implementation information domain;
- *Space*: Information about, e.g., geometry, geographical position, what is located within what. Information about the space that an asset occupies is meant to be captured in terms of the Space information domain.

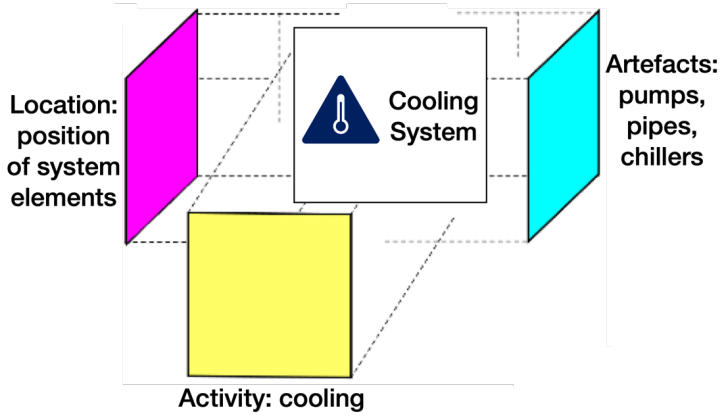


Figure 3.5: Different information domain views on the same system.

Figure 3.5 illustrates how the information domains can be applied to information about a cooling system as delivered from a supplier. In an Activity information domain view, a cooling system is the cooling activity that realises the cooling function. In an Implementation information domain view, a cooling system is comprised of physical system elements, such as pumps, pipes, and chillers. In a Space information domain view, a cooling system is the geometric extension of each of its physical system elements.

Elements of, respectively, the Activity, Implementation, and Space information domains stand in different breakdown structures:

- An Activity can be broken down into sub-activities;
- An Implementation can be broken down into sub-assemblies;
- A Space can be broken down into sub-spaces.

Elements of the respective information domains are connected in different topologies:

- Two elements of the Activity information domain can be connected through a medium;
- Two elements of the Implementation information domain can be physically connected by being mounted together;
- Two elements of the Space information domain can be connected by being adjacent or overlapping.

3.2.1.2 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. An interest will in many

cases be used to capture a specific segment of technical information.

In this IMF Manual there are two predefined terms for interests related to positions in the lifecycle of engineered assets.

- Project lifecycle: This is used when the interest is in identifying needs in project exection and in operation and maintenance;
- Product lifecycle: This is used when the interest is from a supplier or manufacturer viewpoint.

Users can add more fine-grained terms whenever needed. The value chain may include more steps that users may need to distinguish between like, e.g., contractor and client in the project execution, or sub-suppliers in the product lifecycle. The project execution may introduce terms for milestones like, e.g., as-built and as-commissioned.

A fine-grained set of interest terms will provide users with narrow contexts. This will, however, also give rise to many different aspects in the language, and a potential need for integrating information across aspects.

3.2.1.3 Modality

The modality refers to the mode of existence of what is modelled. There are two predefined terms.

- Intended: Information that expresses needs, specifications, or similar engineering related information.
- Actual: Information about physical assets.

Users may add other modalities. A Required modality may be useful in engineering. In a digital twin context, a Simulated modality may be useful.

3.2.2 IMF aspects

An IMF aspect is defined as as a triple: <*Information domain*, *Interest*, *Modality* >.

Table 3.1 lists the six combinations of predefined information domains and interests for modality Intended. Here are some examples of what kind of information that can be captured in each of these aspects:

- Activity need: Functional requirements resulting from simulation. Breakdown of activities to a level of detail where a system function can be identified with capability to deliver the needed activity. Activity needs are typically activity breakdown hierarchies identified by the disciplines with interfaces to one another that bring details on various sorts of medium.
- Area need: Definition of activity spaces and detailed positions and coordinates. The information will typically be organised in breakdown hierarchies with information about how they are interrelated, e.g., where they overlap or where they are adjacent.
- Plant: Components and how they are physically assembled and connected.

Table 3.1: IMF aspects with modality Intended.

Aspect short name	Information domain	Interest	Modality
Activity need	Activity	Project Lifecycle	Intended
Area need	Space	Project Lifecycle	Intended
Plant	Implementation	Project Lifecycle	Intended
Inherent function	Activity	Product Lifecycle	Intended
Product location	Space	Product Lifecycle	Intended
Product type	Implementation	Product Lifecycle	Intended

Table 3.2: Definition of four aspects with no specific interest, and their prefixes and colours.

Aspect	Information domain	Interest	Modality	Prefix	Colour
Function Aspect (F)	Activity	none	Intended	=	yellow
Location Aspect (L)	Space	none	Intended	+	magenta
Product Aspect (P)	Implementation	none	Intended	-	cyan
Installed Aspect (I)	Implementation	none	Actual	::	dark blue

- Inherent function: The capability of a product individual for bringing about an intended activity; the purpose of the product.
- Product location: The space that product individual occupies.
- Product type: The configured product template.

Example. In the lifecycle of an asset it may be important to distinguish between client needs and what is delivered by suppliers. IMF allows users to distinguish between client and supplier interests and thus make information related to each of them be expressed in different aspects. Note that information about client needs is not identical to product information from suppliers, even when this is information about the same asset and both client and supplier are using, e.g., the Activity information domain.

3.2.3 Reserved Aspects and ISO/IEC 81346

The IMF manual uses reserved names for four aspects that have undefined interest, see Table 3.2. Colours are included in the table and are used in later figures in the IMF Manual to show the aspect.

Figure 3.6 illustrates these aspects applied to a pumping system.

- The *Function Aspect* (yellow) ❶ is used to specify the information about activity that realises the function of the pumping system.
- The *Product Aspect* (cyan) ❷ is used to specify the intended implementation of the pumping system.
- The *Location Aspect* (magenta) ❸ is used to specify the size and shape of the specified pump.

- The *Installed Aspect* (dark blue) ❹ is used to give information about the actual implementation, e.g., the serial number, run hours, and status of an actual pump installed in a plant.

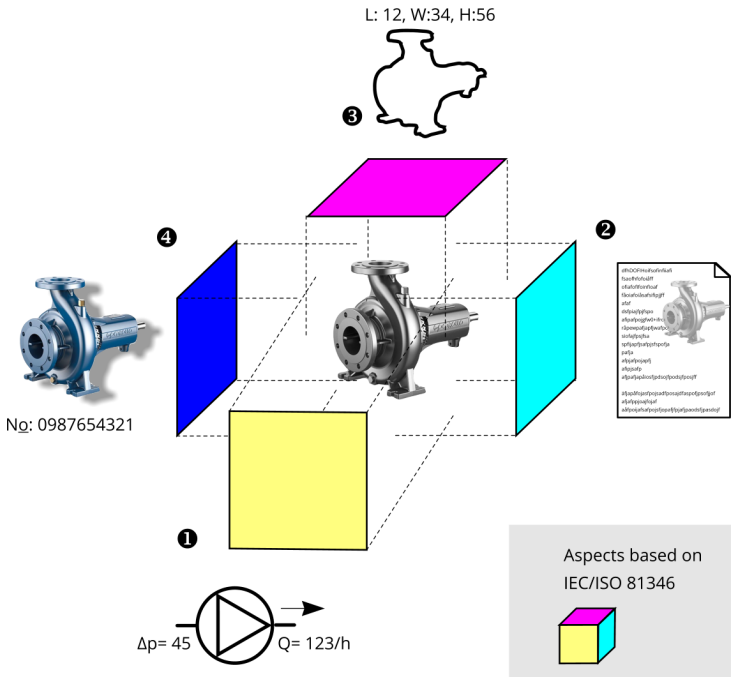


Figure 3.6: The four reserved aspects are illustrated: the Function Aspect (yellow), the Product Aspect (cyan), the Location Aspect (magenta), and Installed aspect (dark blue).

The Function, Product, and Location aspects of Table 3.2 are identical to the core aspects of ISO/IEC 81346. These are the main differences between ISO/IEC 81346 and IMF on the treatment of aspects:

- ISO/IEC 81346 does not have the IMF concept of interest. It does, however, allow introduction of new aspects.
- ISO/IEC 81346 does not address individuals, and hence does not have the Installed aspect in Table 3.2.
- ISO/IEC 81346 has a type aspect. IMF has a concept of type and a concept of classifier. The IMF concepts are not identical to the ISO/IEC 81346 type.

3.2.4 Describing systems using aspects

Section 3.1.4 identifies limitations of the system concept as a structuring principle. The IMF approach to overcoming these limitations is to define breakdown and topologies only within aspects, and allow users to flexibly flip between aspects. According to IMF principles,

- An element of one aspect cannot be drilled down to an element of a different aspect. A breakdown structure hence stays within an aspect;
- An element in a breakdown structure can have at most one parent.

- An element of one aspect cannot be connected to an element of a different aspect. A topology hence stays within an aspect.

A system will typically have different breakdowns and topologies in different aspects. But within one and the same aspect, there will be no ambiguity:

- An element can have at most one breakdown;
- An element is part of at most one breakdown;
- Two elements are connected through a topology in at most one way.

As a result, breakdown structures within aspects can be used to unambiguously zoom in when more detail is needed, and zoom out when a more comprehensive view is needed. This feature substantially facilitates navigation in technical information.

The definition of aspect provides users with flexibility to add new aspects when this is needed for, e.g., an element in a breakdown structure to have just one parent. A user can in particular add an interest and form new aspects with this interest so that each element of this aspect has a unique breakdown.

Note that the reserved aspects Function, Product, Location, and Installed will as a rule not be sufficient for meeting the condition that an element in a breakdown structure can have at most one parent.

In order to flip between aspects, IMF supports introduction of inter-aspect relations. One basic inter-aspect relation is the proxy relation. Two elements are said to be proxies if they are different views of the same thing, e.g., different views of the same system.

3.3 Information model

An information model is an *information artefact*, i.e. information that is constructed in order to describe something. IMF models are information models about assets that consist of *elements* and *descriptors* put into relationships by means of *references*, *relations*, and *instantiations* of *types*.

Both elements and relationships result from applying structuring principles that help organising engineering knowledge and technical information. In this process identification of elements, types, and descriptors is closely intertwined with identification of relationships.

The full IMF language is presented in Chapter 4. The aim of this section is to clarify principles that underlie some of the language constructs in IMF.

3.3.1 Element

The IMF language contains three types of element: Block, Connector, Terminal.

The elements are abstract forms that are useful for expressing simplified and aggregated views of technical information. Typically these views are constructed by engineers using elements representing concepts such as groups, systems, and specifications, and relationships between them.

- A block is used to represent anything of interest such as a system or a complex assembly that can be divided into parts.

- A connector is used to represent information about how two blocks connect. When two blocks connect, they connect through a connector. A connector can in particular contain information about medium.
- A terminal is used to express information about state of medium at the boundary of a block.

Blocks and connectors are associated with the concept of a boundary, intuitively meaning that any elements is, or is not, contained a boundary. More precisely:

- A block has a boundary.
- A connector has a boundary.

The connector is an element that can be present in a model even if it is not connected to a block. This is not the case for terminals:

- A terminal can only be placed at the boundary of a block. A connector cannot have terminals at its boundary.
- A terminal does not have a boundary.

3.3.2 Descriptor

A descriptor is a term defined in IMF that is meant to be shared across models and use cases. Using references (cf. Section 3.3.3), both elements and other descriptors can refer to descriptors. IMF contains two types of descriptors:

- Aspect, with the associated descriptor types InformationDomain, Interest, and Modality, cf. Section 3.2;
- Attribute, including Attribute Group and Attribute Qualifier, cf. Section 4.2.10.

3.3.3 Reference

A reference is a term that is used to relate elements and descriptors to descriptors or external resources.

A *classifier* is a reference that only elements can have. Classifiers detail what the element at hand expresses information about. A classifier refers to an external resource such as a class in a reference data library or in an industry standard.

All IMF elements have a mandatory classifier called the primary classifier. They can also have any number of auxiliary classifiers.

Relationships between classifiers, such as the subclass relationship, are not expressed in IMF, but can be expressed in a reference data library. We may think of relationships between classifiers as objective expressions about what something is, e.g., that a pump is a rotating equipment. If the reference data library is structured in, e.g., subclass hierarchies, this structure can exploited for verification and quality checking of IMF models.

An element can have a reference to an Aspect descriptor and thereby specify that it is of the given aspect. If an element is of an aspect, then this aspect is unique.

Aspect details the view on technical information that is expressed through information about the element. Aspects regulate how users can build relationships between elements, e.g., by breaking something down in parts. We may think of such relationships as expressing subjective views, shared by actors with the same interest, on the structure of technical information.

An element may optionally have references to attribute descriptors.

3.3.4 Relation

IMF is designed to facilitate operations on asset information in the engineering workflow through relations between elements. Three such operations are:

- Flipping between abstraction layers, facilitated by *part of* relationships;
- Tracing medium flow, facilitated by *connected to* relationships;
- Tracing requirement-solution threads, facilitated by *fulfills* relationships.

3.3.4.1 The partOf relation

Different abstractions are typically organised in breakdown structures. Something identified as an element in one source of technical information can be part of a group or system in another. The partOf relationship relates elements in breakdown structures.

The concept of a boundary can provide intuitions on how division into parts is understood in IMF:

- A block can contain blocks, connectors, and terminals inside its boundary.
- A block can contain terminals at its boundary.
- A connector can contain blocks, connectors, and terminals inside its boundary, but a connector cannot contain a terminal at its boundary.
- A terminal can contain other terminals, but a terminal cannot contain blocks or connectors.

A breakdown hierarchy is represented as a sequence of partOf relationships such that if A is partOf B, then A is contained in B.

In IMF an element can have at most one partOf parent. Ie., if A is partOf B, and C is different from B, then A cannot also be partOf C. The partOf relation is so-called functional.

The partOf relation cannot relate elements of different aspects.

3.3.4.2 The connectedTo relation

Engineering artefacts such as products, systems, models, or elements in breakdown hierarchies will in general be connected in different ways using media types such as material, energy, force, or information. Traces of the flow of a medium type are typically depicted in topology diagrams.

In IMF a block and a connector can be connected via a terminal. Information about media on connectors gives rise to topologies understood as a chain of connectors that represents the flow of medium.

The connectedTo relation cannot relate elements of different aspects.

3.3.4.3 The fulfills relation

IMF uses the fulfills relation to state a requirement-solution relationship between two elements. It can relate elements of different aspects.

Engineering consists of a series of decisions whereby a solution space is successively constrained. An element of a design specification can be a solution to some requirements and itself pose new requirements for solutions as part of the gradual evolution of the engineering of an asset. Information in specifications such as data sheets can in many cases be traced back to elements of requirement specifications referring, e.g., to industry standards.

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, in procurement

- Client or contractor may provide a datasheet about some product to specify the required product requirement. The client datasheet will select information from the aspect Activity need, Area need, and Plant.
- Supplier provides a datasheet about an offered solution based on some product catalogue of sub-supplier. The supplier datasheet will select information from the Inherent function, Product location, and Product type aspects.

The evolving of information typically follows that in Figure 3.7: 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. These relationships can be expressed in an IMF model using fulfills relationships.

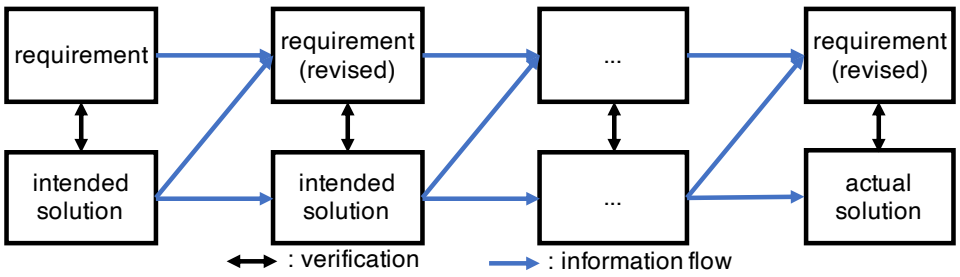


Figure 3.7: The three modalities and a typical evolving flow

3.3.5 Type and type instantiation

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. These patterns are information structures that partially detail engineering solutions. They can be found in libraries, standards, or other industry shared resources.

An *IMF type* is a predefined template for an information structure. It will typically contain elements with classifiers and attribute information, with or without values, potentially with a range of admissible values that an attribute may take. It will typically also contain information about connectors, and, in complex cases, details about a breakdown into additional elements and a topology over those elements.

Types in IMF are meant to reside in a library from which they can be accessed, instantiated, and reused. 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.

An element in an IMF model can be traced to types of which it is an instance. An element can in general be an instance of several types, and a type can instantiate several elements and relationships between them.

3.4 IMF concept and languages

IMF allows users to construct information models using only three element types. A restricted set of relations allows the user to construct clearly defined breakdown hierarchies, topologies that represent flow of medium, and requirement-solution chains through the information model. Models can be composed by combining reusable types. By adding classifiers to elements, classification hierarchies provided by external reference data libraries can be exploited.

The IMF concepts are provided in different syntactic forms to different user groups and for different purposes:

- A visual language
- A formal RDF specification language
- An exchange protocol using AAS
- A formal semantics in formal logic

The focus of this document is the visual syntax.

Visual language. The purpose of the visual language is to provide an easy to use language to the discipline experts to facilitate creation of information models, navigation in such models, and formulation of interesting views of a model.

The visual language is meant to be used in applications supporting creation and manipulation of a simple set of graphical elements and relationships for building graphs using a restrictive set of templates. The resulting graphs are meant to be composed of recognizable links between the elements that should be intuitively simple to navigate through. When a specific view is of interest, the user should be able to select elements and relationships of that view.

RDF representation. The RDF representation captures the meaning of an IMF model in a more precise way than the visual language. RDF is a W3C recommendation (ie. a standard for the semantic web) for specifying graphs. It enables the representation of complex constraints that

are useful for developers of applications for the visual language; such constraints can help users avoid creating patterns in the model that go against the intended meaning.

An expert in RDF can also write a graph pattern and, using the W3C recommended query language SPARQL, pose this as a query to an RDF graph; the query will then return patterns in the graph that match the pattern of the query. If the discipline expert specifies an interesting view, the RDF expert can hence implement this view by writing the corresponding query, execute this over the model, and visualize the query result to the user as a view of the information model.

Exchange protocol. Information about assets is fragmented; it is created in and exchanged between silos arising between disciplines and along the value chain. When IMF models are developed in silos, there is need for serialisations of the models that can be effectively exchanged.

RDF is a serialisation format for the semantic web that can be used as exchange protocol when it is supported. Otherwise it is practical to use protocols that the industry has already implemented. This can be supported in IMF by specifying a translation from the RDF representation to the serialisation format at hand that preserves model structures, i.e. a structure preserving translation.

The Asset Administration Shell is an industry implemented protocol for exchange of information between supplier and client as part of the Industry 4.0 program. The IMF project aims to support exchange of model information using AAS by providing a profile to AAS. This is not included in this version of the IMF manual.

Formal semantics. By using formal logic, including first-order logic, the meaning of IMF models can be expressed as rules and constraints even more concisely than in the RDF representation. The formal semantics is also referred to as the interpretation of IMF.

Application developers can use the formal semantics to build logic into applications. Since formal semantics supports automated reasoning, the formal semantics can also be exploited for powerful integrity checking and formal verification.

In particular the formal semantics can provide a bridge to the Industrial Data Ontology (IDO), which is particularly designed to exploit the automated reasoning capabilities of the OWL, the web ontology language recommended by W3C. When elements of an IMF model are classified using resources in a reference data library that conforms to IDO, off the shelf automated reasoning tools can already today provide very efficient and advanced integrity checking. IMF is designed to enable such use of IDO and associated automated reasoning services.

Chapter 4

IMF Language Overview

This specification gives a formal description of the Information Modelling Framework (IMF) language. Its purpose is to provide an abstract and implementation independent specification of the IMF language, and with that give an introduction to the language and provide a common basis for practical and theoretical applications of IMF, such as specifying logical formalizations, serialization formats, and software applications and services that use IMF.

The specification is based on the developments of the following documents:

- **IMF Language: Logic Formalization**, which gives a first-order logic formalization of the IMF language.
  https://gitlab.com/imf-lab/spec/imf-language-logic-formalisation
- **IMF Language: Guidelines for Visualizing IMF Models**, which defines a graphical language for drawing IMF models with nodes, edges and colours.
  https://gitlab.com/imf-lab/spec/imf-language-visualisation
- **IMF Language: Semantic Technologies Implementation**, which specifies an implementation of the IMF language using the W3C semantic technology recommendations OWL, SHACL and RDF.
  https://gitlab.com/imf-lab/spec/imf-ontology

Notation

This document uses the following notation.

Ontology

The IMF vocabulary is written using the following typing:

- Classes are written using UpperCamelCase and appear as unary predicates in examples.
- Relations are written using lowerCamelCase and appear as binary predicates in examples.
- Individuals are written using lowerCamelCase and appear as constants in examples.

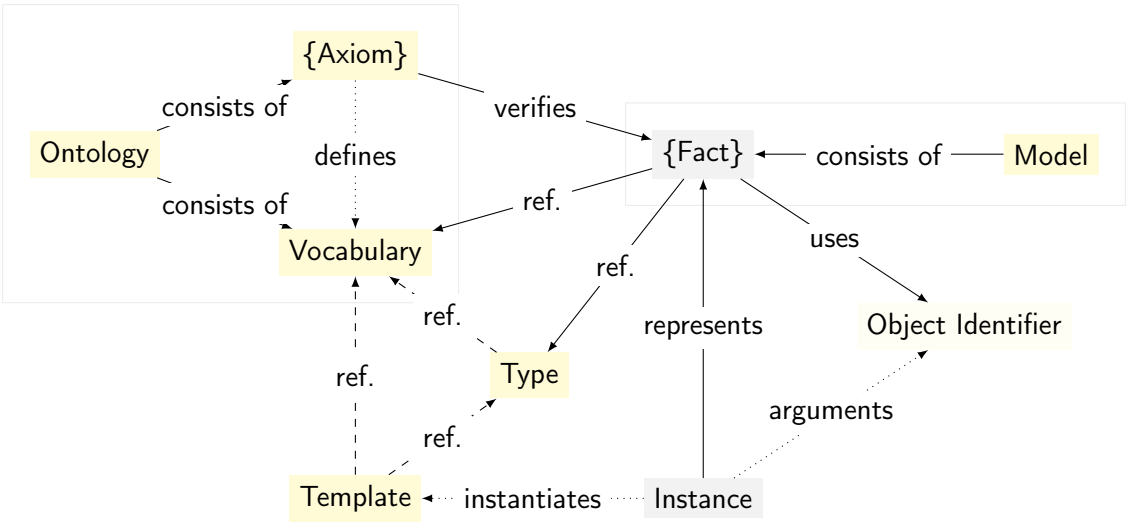


Figure 4.1: The parts of the IMF language and how they are related.

Diagrams

The diagrams that illustrate how classes, relations and individuals in the IMF vocabulary are related are informative only (and not normative). They indicate subclass relationships between classes, membership relationships between individuals and classes, and the domain and range of relations.

4.1 Overview

The IMF language is split into the following main parts: ontology, models, types and templates. Closely related to the IMF language are also (object) identifiers. A high-level description of these parts are described below and informally displayed in Figure 4.1.

4.1.1 IMF Ontology

The ontology comprises

- A vocabulary, i.e., a reserved set of terms, that is used to express facts about the target domain.
- A set of rules over the vocabulary, called axioms, that describe the intended use of the vocabulary and hence how the vocabulary is used to express correct and meaningful facts.

The IMF ontology is described in Section 4.2.

Example 4.1.1. *The IMF ontology contains vocabulary terms such as Block, Terminal, partOf and connectedTo, and axioms such as "A Terminal can only be partOf other Terminals" and "Blocks are not*

Terminals", which can be formalized as follows with these logic formulas:

$$\forall x, y(\mathit{Terminal}(x) \wedge \mathit{partOf}(x, y) \rightarrow \mathit{Terminal}(y))$$
$$\forall x(\mathit{Block}(x) \rightarrow \neg \mathit{Terminal}(x))$$

4.1.2 Identifiers

In the context of the IMF language specification, an object identifier is a name for an object or entity that is described using the IMF language.

A goal of IMF is to be able to clearly identify distinct descriptions of assets, and to integrate data that is typically distributed across multiple systems and diagrams using multiple identifiers.

Example 4.1.2. *Examples of identifiers are tags, system identifiers and database keys.*

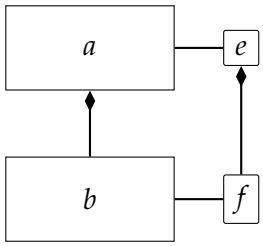
In our simple formal examples, we use a, b, c, ... to represent identifiers.

4.1.3 IMF Models

An IMF model is a language construct for grouping and organizing facts expressed using the IMF vocabulary, i.e., it is essentially just a set of facts, a dataset. A fact is expressed using the vocabulary and identifiers.

Example 4.1.3. *The following is an IMF model:*

$\{\ \mathit{Block}(a), \mathit{connectedTo}(a, e), \mathit{Terminal}(e),$
$\mathit{Block}(b), \mathit{connectedTo}(b, f), \mathit{Terminal}(f),$
$\mathit{partOf}(b, a),$
$\mathit{partOf}(f, e)\ \}$



The model states that a and b are Blocks, and e and f are Terminals. a is connectedTo e, b is connectedTo f, b is partOf a, and f is partOf e. This model complies with the axioms in Example 4.1.1. The model is visualized using the graphics listed in Figure 4.2.

4.1.4 IMF Types and IMF Templates

Informally, *types* are used by SMEs to express typical configurations of model elements, in order to support standardization, reuse, and consistency and uniformity of information models. A type will typically represent a concept that is standard throughout an engineering discipline.

A *template* is used to consistently create a specific configuration of model elements, in order to simplify model construction and maintenance. The configuration that a template represents need not—but can—be standard, and may be created to support modelling consistency and uniformity within only a specific engineering project.

Hence, types and templates are similar, but they differ in their scope; types are expected to serve a wider use than templates, while templates will on the other hand typically represent more detailed configurations, often orchestrating and configuring types for particular modelling use

cases. Instantiating types and templates from carefully curated libraries is a primary way of efficiently creating and maintaining information models of high quality.

This informal understanding of types and templates gives rise to the following more formal definition of IMF types and IMF templates.

An *IMF type* is represented by an expression over the IMF vocabulary that defines a class of all the objects that adhere to the configuration specified by the IMF Type. These objects are called instances of the type or type instances. An *IMF template* is a language construct for representing reusable modelling patterns over the IMF vocabulary. A template instance is a replica of the template's modelling pattern that follows the conditions that may be specified by the template. A template instance represents a set of facts over the vocabulary. Hence, a template instance represents a small IMF model.

There are hence clear similarities between an IMF type and an IMF template. The difference is that an IMF type *describes* a class of objects using the IMF vocabulary, while an IMF template is used to *generate* a set of facts over the IMF vocabulary. An IMF Template may refer to an IMF Type and hence generate facts that are instances of IMF Types.

Example 4.1.4. *A very simple (and artificial) IMF type is Block-with-Terminal-Type:*

$$\mathit{Block\text{-}with\text{-}Terminal\text{-}Type}(x) \rightarrow (\mathit{Block}(x) \wedge \exists y\, \mathit{Terminal}(y) \wedge \mathit{connectedTo}(x, y))$$

This IMF Type specifies the class of all Blocks that have a Terminal.

Example 4.1.5. *A very simple IMF template is Block-with-Terminal-Template:*

$$\mathit{Block\text{-}with\text{-}Terminal\text{-}Template}(x, y) \mapsto \{\mathit{Block}(x), \mathit{Terminal}(y), \mathit{connectedTo}(x, y)\}$$

The x and y are the parameters of the template, and it represents the pattern of a Block with a Terminal connected to it.

An instance of the template, Block-with-Terminal-Template(a, b), expands to the model:

$$\{\mathit{Block}(a), \mathit{Terminal}(b), \mathit{connectedTo}(a, b)\}$$

Note that the object a is an instance of the IMF type Block-with-Terminal-Type from Example 4.1.4.

4.2 Ontology

4.2.1 Overview

A vocabulary is a reserved set of terms which is split into *classes*, *relations* and *individuals*.

4.2.1.1 Classes

The single top-most class in the IMF vocabulary's class hierarchy is Entity; everything is a Entity. Entity is a very generic concept that is used by the ontology to state generic axioms that hold for all objects described by the ontology. Entity should not be directly instantiated, rather instantiate an appropriate subclass of Entity.

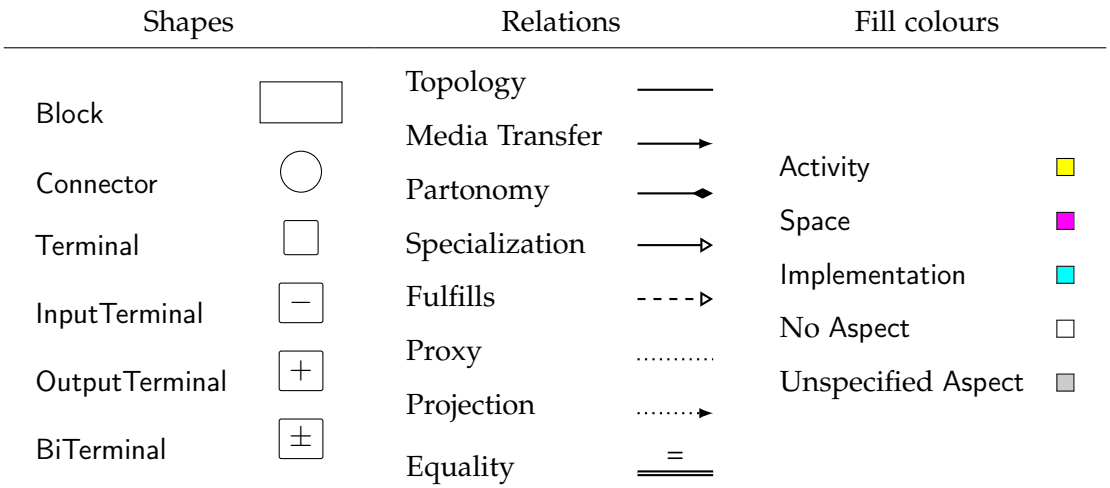


Figure 4.2: All graphical elements.

4.2.1.2 Individuals

The ontology contains a limited number of individuals that represent reserved words in the vocabulary, such as names for Aspects. In this specification these individuals are presented as part of the presentation of the main class they are member of.

4.2.1.3 Relations

All relations are binary, i.e., all relationships relate two objects. As a high-level and informal categorization of relations, we distinguish between:

- *Object* and *datatype* relations. Object relations relate two objects, while datatype relations associate a data value, such as a string, a number or a date, to an object.

For object relations the categorization also includes:

- *Hierarchical* vs. *associative* relations. Hierarchical relations specify an ordering or ranking, while associative relations (or non-hierarchical relations) do not specify a clear ordering and often relate elements of similar rank (as specified by a hierarchical relation).
- *Internal* vs. *external* relations. Internal relations are defined with a domain and range that is specified by IMF classes, while *external relations* do not and will usually relate to objects from external reference data libraries.
- *Logical* vs. *annotation* relations. Logical relations have formal logical meaning which must be considered by logical reasoners, while annotations are typically used for metadata such as creation date and may be ignored by logical reasoners.

Unless otherwise noted, all relations are object type, associative, internal and logical.

4.2.2 Generic relations

Membership and equality relationships are generic relations that can be used to describe and reuse abstractions and categorizations. They are applicable to all Entity-s.

4.2.2.1 Equality

sameAs is a relation between Entity-s, such that sameAs(a, b) expresses that the Entity-s a and b are equal, i.e., a and b are different names/identifiers for the same Entity.

Axioms.

(1) sameAs is a relation between Entity-s.

(2) sameAs is reflexive, symmetric, and transitive.

4.2.2.2 Membership

memberOf is a relation between Entity-s, such that memberOf(a, b) expresses that the Entity a is a member of the Entity b, and hence that b is a collection or group for a. The inverse relation of memberOf is hasMember.

Axioms.

(3) memberOf is a relation between Entity-s.

(4) hasMember is the inverse of memberOf.

4.2.3 InformationArtefacts

Entity has a single subclass InformationArtefact that is the common class for the core categories of the ontology: Element, Descriptor and InformationModel (Figure 4.3). These subclasses are pairwise disjoint.

Element (Section 4.2.5) is the core class of the ontology. All classes and relations in the ontology exist to describe Elements—either directly or indirectly.

A Descriptor is an InformationArtefact that serves to describe or identify a feature of an Entity.

A reference is a generic relation that relates a resource to a Descriptor.

Descriptors are associated with other Descriptors with the reference relation hasCharacteristic.

Axioms.

(1) InformationArtefact is a subclass of Entity.

(2) Element, Descriptor, and InformationModel are subclasses of InformationArtefact.

(3) Element, Descriptor, and InformationModel are pairwise disjoint.

(4) reference is a relation to a Descriptor.

(5) hasCharacteristic is a subrelation of reference that relates Descriptors.

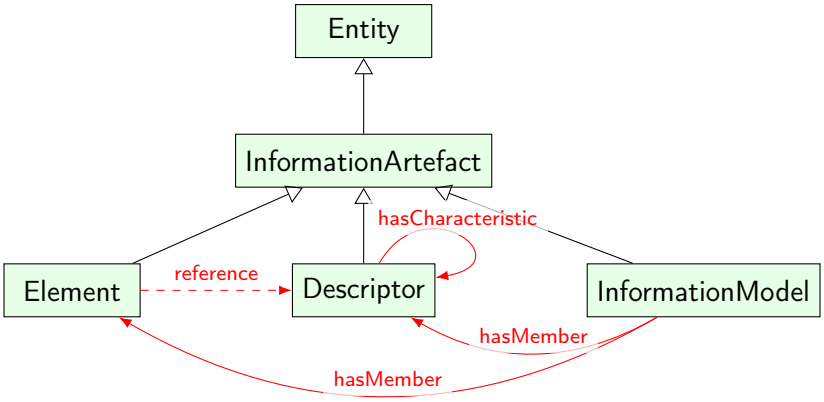


Figure 4.3: InformationArtefact and subclasses.

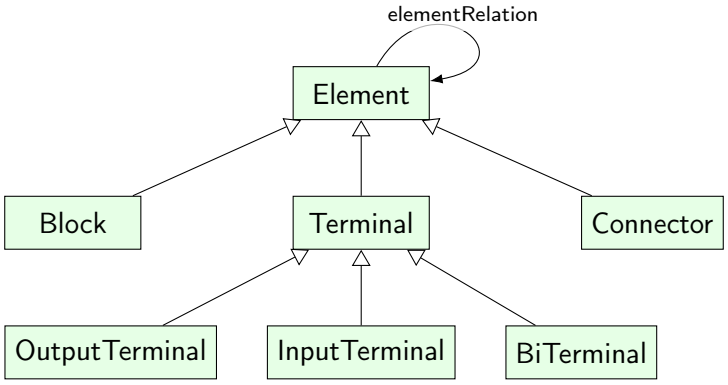


Figure 4.4: Element and its subclasses.

4.2.4 InformationModel

Elements and Descriptors may be collected into and organized by InformationModels, hence an InformationModel represents a set of such InformationArtefacts. Elements and Descriptor are associcated with an InformationModel using the memberOf/hasMember relations.

4.2.5 Elements

Figure 4.4 displays the class Element and its subclasses. Figure 4.5 shows the graphical language for describing Elements.

4.2.5.1 Element

An Element represents the specification of an asset. All assets specifications are represented by an Element. An Element is described through its relations (Section 4.2.8) to other Elements, its Aspect (Section 4.2.6), and it may be assigned Attributes (Section 4.2.10).

An Element is either a Block, a Connector or a Terminal. Individuals rarely instantiate the class Element directly, rather they are stated to be an instance of a subclass of Element. (Element can

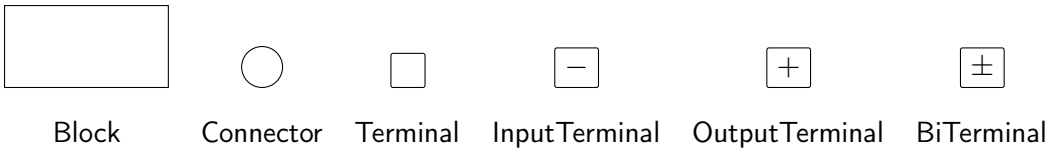


Figure 4.5: All node shapes.

be considered an "abstract class".)

The permissive relationships for an Element are controlled by its type of subclass, and its Aspect.

An elementRelation is a generic relation that relates Elements.

Axioms.

(6) An Element is either a Block, a Connector, or a Terminal.

(7) Block, Connector, and Terminal are pairwise disjoint.

(8) elementRelation is a relation between Elements.

4.2.5.2 Block

A Block represents an abstraction of any entity at any abstraction level. A Block also represents a boundary that separates what is internal and external to the Block. Elements that are part of the Block are internal to the Block and are not directly accessible to Elements outside (the boundary of) the Block. When a Block represents a system, it can be natural to consider the Block as a processing black box that transforms input to output, where the transformation process is detailed by the internals of the Block.

4.2.5.3 Connector

A Connector represents a topological connection between two Blocks, and it will specify conditions on this connection, such as the medium that is transferred between the Blocks.

A Connector can be considered as a Block with infinitely small boundary, i.e., a Block where there is no transformation of the input to the output at the current abstraction level of the Connector. A Connector may be further described by its parts.

4.2.5.4 Terminal

A Terminal represents a point on a Block's boundary where medium may pass, and it will specify conditions on the medium that may pass through it. A Terminal describes the permissible input and/or input of a Block. Any interaction with a Block goes through a Terminal of the Block.

A Terminal depends on a Block, hence a Terminal cannot exist without a Block.

Blocks may interact by transferring media between them (Section 4.2.8.4). A Terminal may refer to direction that specifies if the Terminal is set up to send or receive media. A Terminal that is specified to only receive input is called an InputTerminal. A Terminal that is specified to

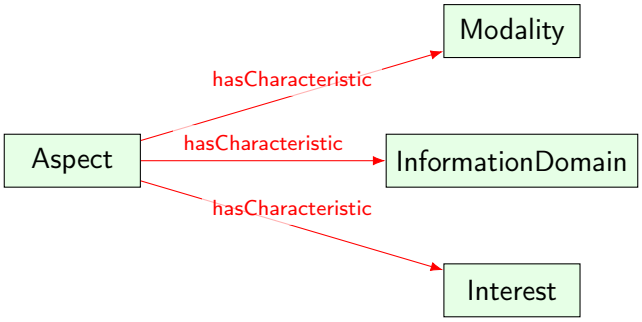


Figure 4.6: Aspect.

only send output is called an OutputTerminal. A Terminal that may both receive input and give output is called a BiTerminal.

Axioms.

(9) direction is a subrelation of reference that is functional relation and associates a Terminal to a transfer direction.

(10) An InputTerminal is a Terminal with direction input.

(11) An OutputTerminal is a Terminal with direction output.

(12) A BiTerminal is a Terminal with direction bidirectional.

4.2.5.5 Classifiers

classifier is a reference from an Element, such that classifier(a, b) expresses that the Element a is classified by the Descriptor b. The relation is typically used to categorize Elements according to identifiers from standard reference data libraries or fixed enumerations for the purpose of connecting to and reusing well-known and shared identifiers and definitions.

Subrelations of classifier define for more specific classification relations. We distinguish between primaryClassifiers that assign *essential* descriptions or features to Elements, and auxClassifiers (auxiliary) that assign *inessential* descriptions or features to Elements.

An Element should have at least one primaryClassifier that identifies the logically necessary types of an Element. Additional classifiers of Elements may be assigned using auxClassifier.

Axioms.

(13) classifier is a subrelation of reference that relates an Element to a Descriptor.

(14) primaryClassifier and auxClassifier are subrelations of classifier.

4.2.6 Aspects

Elements can be described from a specific point of view or aspect, in the IMF language this is represented by Aspects. An Aspect is a Descriptor specified by one Modality, one InformationDomain, and one optional Interest—all of which are also Descriptors.

Modality	Interest	EntityType	Aspect
intended		activity	intendedActivity (functionAspect)
intended		space	intendedSpace (locationAspect)
intended		implementation	intendedImplementation (productAspect)
actual		implementation	actualImplementation (installedAspect)
intended	projectLifecycle	activity	intendedProjectActivity
intended	projectLifecycle	space	intendedProjectSpace
intended	projectLifecycle	implementation	intendedProjectImplementation
intended	productLifecycle	activity	intendedProductActivity
intended	productLifecycle	space	intendedProductSpace
intended	productLifecycle	implementation	intendedProductImplementation

Figure 4.7: Aspects

Aspect/Entity type filter	Colour name	Colour Code
No Aspect	White	`#ffffff`
Unspecified Aspect	Grey	`#cccccc`
Activity	Yellow	`#ffff00`
Space	Magenta	`#ff00ff`
Implementation	Cyan	`#00ffff`

Figure 4.8: Predefined colours for Entity type filters and Aspects.

Modality specifies the mode of existence of what is described. There are two predefined Modality-s: intended and actual.

Interest specifies the modelling interest or context from which the Element is described. There are two predefined Interests: projectLifecycle and productLifecycle

Aspects are associated with an InformationDomain that specifies what type of entities are in described by the Aspect. There are three predefined InformationDomains: activity, implementation and space.

The various combinations of Modality, InformationDomain and Interest give rise to different Aspects; these are listed in Figure 4.7.

Axioms.

(15) Aspect, Modality, Interest and InformationDomain are subclasses of Descriptor.

(16) intended and actual are Modality-s.

(17) projectLifecycle and productLifecycle are Interests.

(18) activity, space and implementation are InformationDomains.

(19) An Aspect is associated by one Modality, optionally one Interest and one InformationDomain via the hasCharacteristic relation.

(20) intendedActivity, intendedSpace, intendedImplementation, intendedProjectActivity, intended-

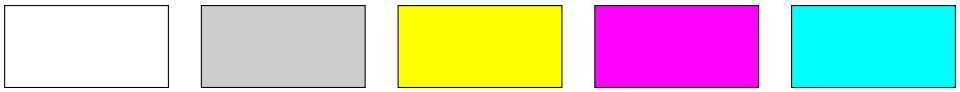


Figure 4.9: Predefined colours demonstrated on Blocks.

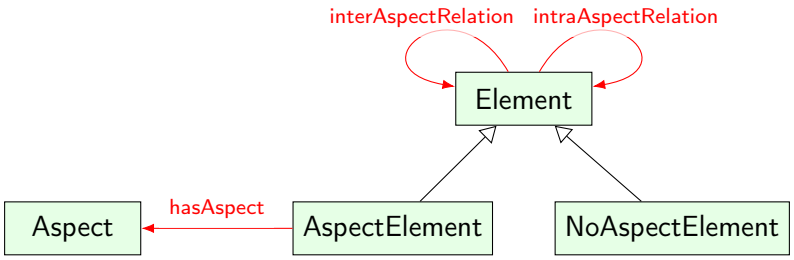


Figure 4.10: Aspects and Elements.

ProjectSpace, intendedProjectImplementation, intendedProductActivity, intendedProductSpace and intendedProductImplementation are Aspects and are specified as in Figure 4.7.

4.2.7 Aspect Elements

An Element may have at most one Aspect. An Element that has an Aspect is called an AspectElement.

It is convenient to be able to identify Elements that carry descriptions that are not of a single Aspect and where the Element's description is not analysed into separate Aspects. Such Elements are called NoAspectElements and have explicitly no Aspect.

Note that an Element where its Aspect is not known is not a NoAspectElement. Such an Element is neither an AspectElement nor a NoAspectElement, simply an Element.

Aspects are used to control the permissive relationships an Element may have, in part by way of the relations intraAspectRelation and interAspectRelation. An intraAspectRelation relates Elements that have the same value assigned as its Aspect, while an interAspectRelation relates Elements that have different values.

Axioms.

(21) hasAspect is a subrelation of reference that is a functional relation from Element to Aspect.

(22) An AspectElement is an Element with an Aspect assigned to it by the relation hasAspect.

(23) A NoAspectElement is an Element with no Aspect.

(24) intraAspectRelation and interAspectRelation are subrelations of elementRelation.

(25) If two Elements are related by an intraAspectRelation they must either share the same Aspect or have no aspect.

(26) If two Elements are related by an interAspectRelation they cannot share the same Aspect.

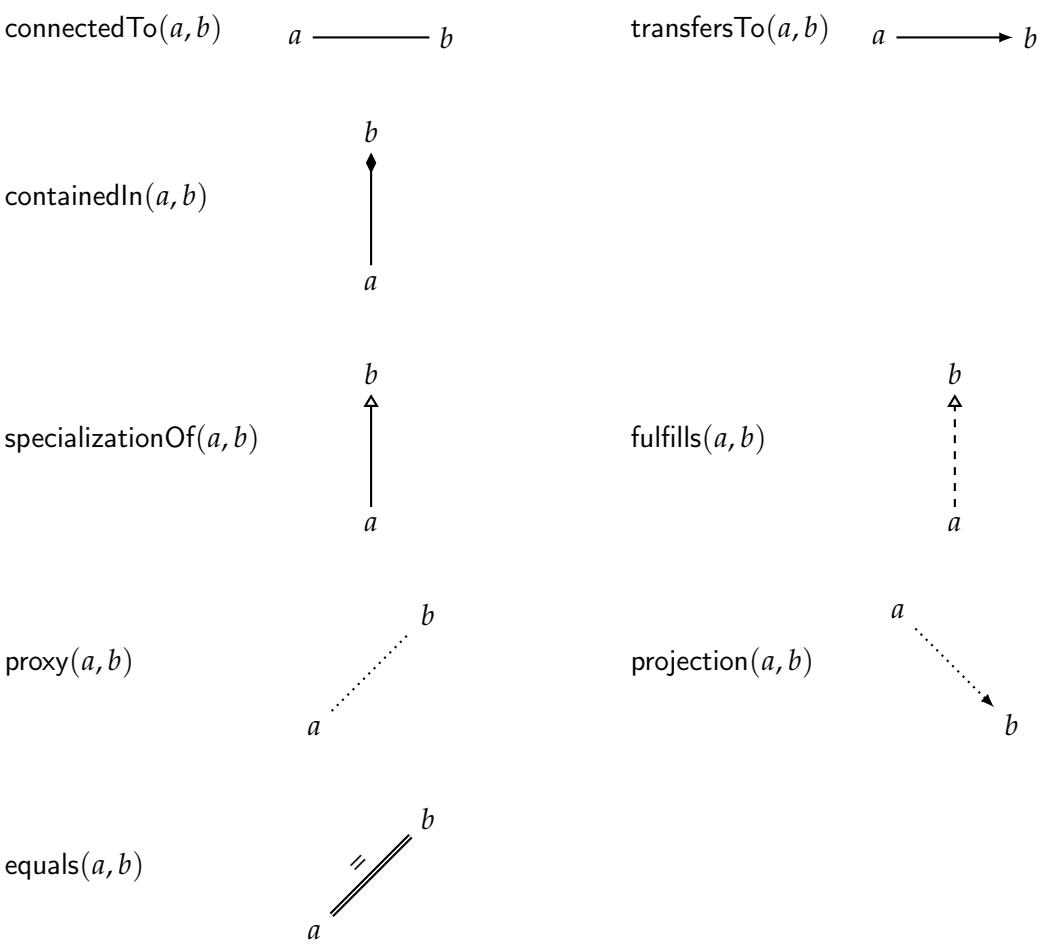


Figure 4.11: All edge types, with their preferred orientation indicated.

4.2.8 Relations

The IMF ontology contains a spesific set of relations for describing Elements. The main relational structures are specialization, topology, partonomy, requirement–solution relationship, and relations between AspectElement that describe different aspects of the same entity. All these relations are subrelations of elementRelation, either directly or indirectly.

Figure 4.11 shows the edges available for visualizing relationships.

4.2.8.1 Specialization

specializationOf is a hierarchical relation and a subrelation of intraAspectRelation, such that specializationOf(a, b) expresses that the Element a is a specialization of the Element b. If a is a specializationOf b, then a inherits *all* the facts expressed for b.

specializationOf is used when specialized Elements are created from one or more generic Elements, or if the shared facts for multiple Elements are to be refactored into new generalized Element(s). Using specializationOf avoids repetition of facts and makes models more compact,

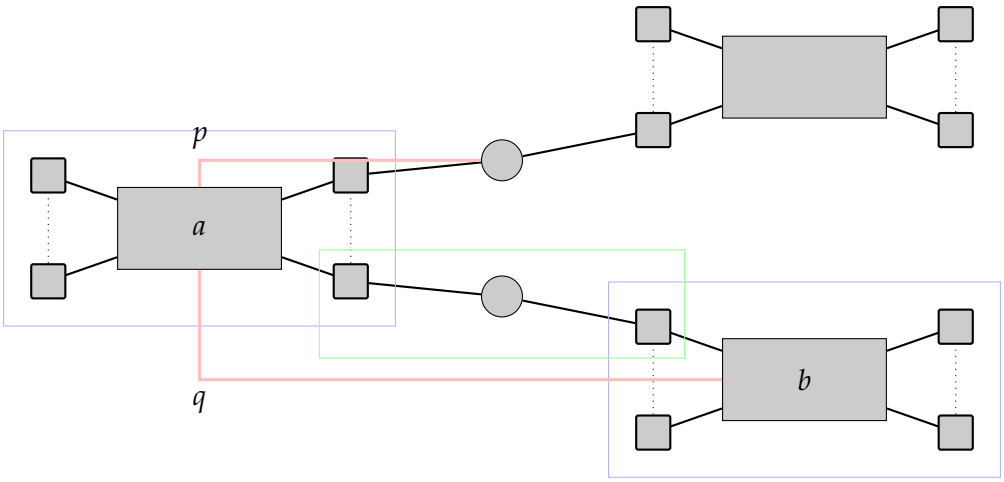


Figure 4.12: Pattern for topologically connected Elements: Blocks are through their Terminals (boxed in blue) connected via Connectors (boxed in green). A connectedTo relationship between a Block and a Connector (the edge labelled p) entails the existence of a Terminal between them. A connectedTo relationship between two Blocks (the edge labelled p) entails the existence of a pattern of the form of the green boxed pattern between them.

possibly at the expense of added cognitive complexity to models.

Axioms.

(27) specializationOf is subrelation of intraAspectRelation.

(28) specializationOf is reflexive, anti-symmetric, and transitive.

(29) If a is a specializationOf b, then a inherits all facts expressed for b.

(30) specializes is the inverse relation of specializationOf.

4.2.8.2 Topology

Topological relations describe how Elements are connected and interact. Connected Elements are described on a similar level of abstraction.

The defining topology relation is connectedTo. connectedTo is an associative relation between Elements, such that connectedTo(a, b) expresses that a is topologically connected to b. adjacentTo is a subrelation of connectedTo, such that adjacentTo(a, b) expresses that a is directly topologically connected to b. That means that there is no element topologically "in between" a and b.

Two "shorthand" relations are also defined: hasTerminal is a subrelation of adjacentTo between Blocks and Terminals, while hasConnector is a subrelation of adjacentTo between Terminals and Connectors.

The difference between these topological relations is that connectedTo allows for expressing the topology of Elements between *any* kind of Elements, e.g., but not limited to, two Blocks, or a Block and a Connector, while the use of adjacentTo and its subrelations is more restricted.

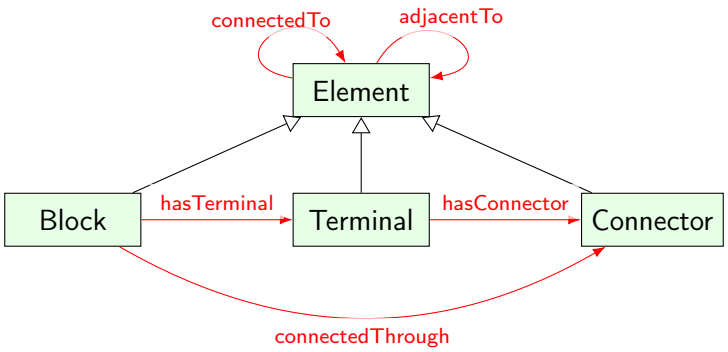


Figure 4.13: Topology relations.

However, a connectedTo relationship is required to be realizable by a chain of adjacentTo relationships. This is to force the following reoccurring pattern in all topological chains of Elements:

$$\ldots \rightarrow \text{Block} \rightarrow \text{Terminal} \rightarrow \text{Connector} \rightarrow \text{Terminal} \rightarrow \text{Block} \rightarrow \ldots$$

Note however that each Element in the chain is not necessarily explicitly expressed, but may be left implicit (Figure 4.12). This can be useful in the case of representing incomplete information which may occur early in the modelling process or when the information to be modelled is at a very high abstraction level where, e.g., the conditions for Terminals and Connectors are not known, but it is clear which Blocks are connected.

The following axioms are defined for topological relations (Figure 4.13):

Axioms.

(31) connectedTo is an intraAspectRelation.

(32) adjacentTo is a subrelation of connectedTo.

(33) connectedTo and adjacentTo are symmetric.

(34) hasTerminal is a subrelation of adjacentTo that associates a Block with its Terminals.

(35) connectedThrough is a relation from a Block to a Connector such that the relation is equal to the composition of hasTerminal and hasConnector.

(36) hasConnector is a subrelation of adjacentTo that associates a Terminal to its Connectors.

(37) Blocks and Connectors may be adjacentTo to only Terminals.

(38) Elements may be connectedTo any number of Elements.

(39) A Block may be adjacentTo any number of Terminals.

(40) A Terminal must be adjacentTo one Block.

(41) A Connector must be adjacentTo exactly two Terminals.

(42) A connectedTo relationship must be realizable by a sequence of adjacentTo relationships.

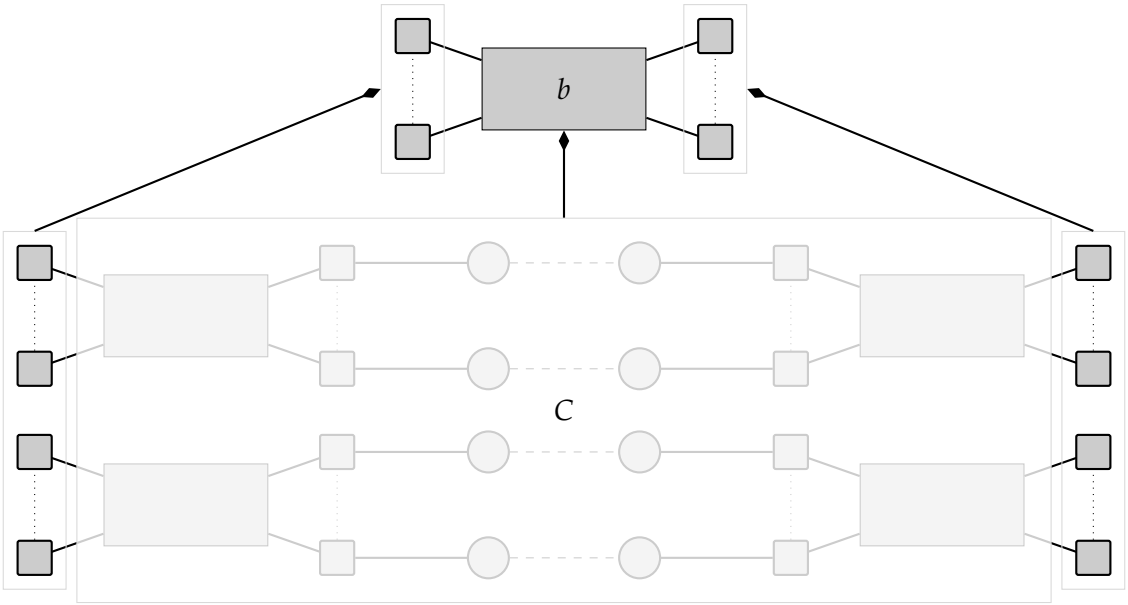


Figure 4.14: Pattern for breakdown hierarchies. The Block b is broken down to a set of Elements (boxed by C) that are all partOf b and form a legal topological structure, and is such that adjacentTo relationships from a part of b can only go to other parts of b or to the parts of b's Terminals. All the Terminals that connect into an Element of C (and not between Elements of C) must be partOf one of b's Terminals. Note that the surrounding boxes used in the diagram are simplifications; relations must go between Elements and not sets of Elements.

4.2.8.3 Partonomy

Partonomy relations relate Elements at different levels of abstraction. An Element can be described by relating it to its *parts*, which are Elements described at a lower level of abstraction, and by it relating to the *whole* it is part of, which is an Element at a higher level of abstraction. This recursive breakdown structure allows Elements to be described at incremental finer—or coarser, level of detail.

The defining partonomy relation is partOf. partOf is a hierarchical relation between Elements, such that partOf(a, b) expresses that a is part of b. In such a relationship, we call a "the part" or "child", while b is called "the whole" or "parent". Any Element is a partOf at most one other Element; the partOf hierarchy forms a tree-structured breakdown. hasPart is the inverse relation of partOf.

containedIn is the transitive closure of partOf, and contains is the transitive closure of containedIn.

The parts of an Element must form a legal topological structure as described in Section 4.2.8.2. Elements can have any kind of Elements as parts, except Terminal whose parts must be Terminals. The parts of an Element are internal to the Element and may not directly connect to external Elements, except to Elements that are partOf the Terminal of the parent Element (Figure 4.14).

The following axioms are defined for partonomy relations (Figure 4.15):

Axioms.

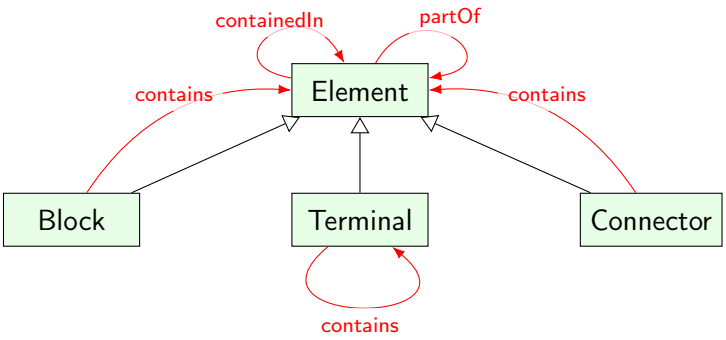


Figure 4.15: Partonomy relations.

(43) containedIn is an intraAspectRelation.

(44) containedIn is transitive.

(45) The inverse relation of containedIn is contains.

(46) partOf is a (non-transitive) subrelation of containedIn.

(47) partOf is functional.

(48) partOf is irreflexive.

(49) hasPart is the inverse relation of partOf.

(50) A containedIn relationship must be realized by a chain of partOf relationships.

(51) A Terminal can only have Terminals as parts.

(52) If two Elements are adjacentTo each other either, then they are each partOf two Elements that are either equal or they are also adjacentTo each other.

4.2.8.4 Media transfer

Topologically connected Blocks may interact by transferring media between them. The media transfer must respect the direction and media specified for the Terminals involved in the transfer.

The defining relation for media transfer is transfersTo. transfersTo is a relation between two Terminals or between two Blocks such that transfersTo(a, b) expresses that (the source Element) a transfers to (the target Element) b. A transfersTo relation between two Blocks must be realized by a transfersTo relationship between Terminals that belong to the Blocks. transfersTo is a non-symmetric subrelation of connectedTo: only Elements that are connectedTo each other can transfersTo each other, with the following restrictions: an OutputTerminal may only transfersTo an InputTerminal, and a BiTerminal may only transfersTo a BiTerminal. An InputTerminal cannot be the source Element in a transfersTo relationship, it may only be the target.

medium is a subrelation of classifier that indicates the medium that a Terminal or Connector handles.

The following axioms are defined for media transfer relations (Figure 4.16):

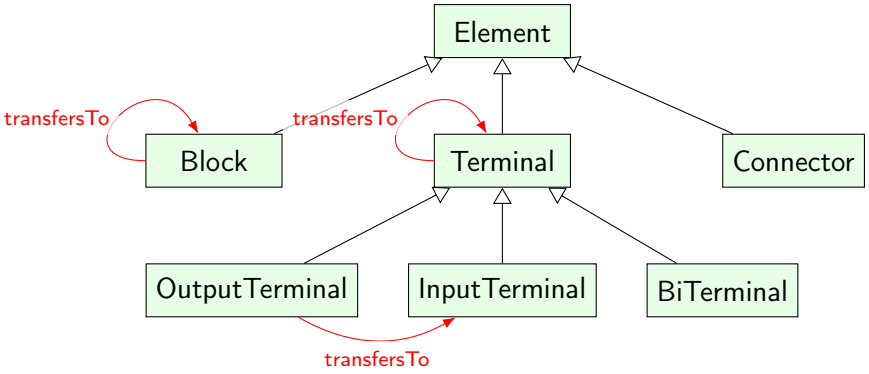


Figure 4.16: Media transfer relations.

Axioms.

(53) medium is a subrelation of classifier that relates a Terminal or a Connector to an external medium classifier.

(54) transfersTo is a relation between Terminals or between Blocks.

(55) transfersTo is a subrelation of connectedTo.

(56) transfersTo is irreflexive.

(57) OutputTerminals may only transfersTo InputTerminals.

(58) BiTerminals may only transfersTo BiTerminals.

(59) InputTerminals may not transfersTo.

(60) A transfersTo relationship between Blocks must be realized by a transfersTo relationships between Terminals.

4.2.9 AspectElement relations

Descriptions of different aspects of entities are represented by different AspectElements. AspectElement relations are relations that relate AspectElement which represent different Aspect of the same entity.

An entity may only be represented by at most one AspectElement for each Aspect, and one NoAspectElement.

There are two AspectElement relations. projection is the relation between a NoAspectElement and an AspectElement, such that projection(a, b) expresses that a is an NoAspectElement representation of an entity and b (called the projection of a) is an AspectElement representations of the same entity. proxy is the relation between two AspectElements, such that proxy(a, b) expresses that a and b are two different AspectElements that are projections of the same NoAspectElement (Figure 4.17).

The following axioms are defined for AspectElement relations:

Axioms.

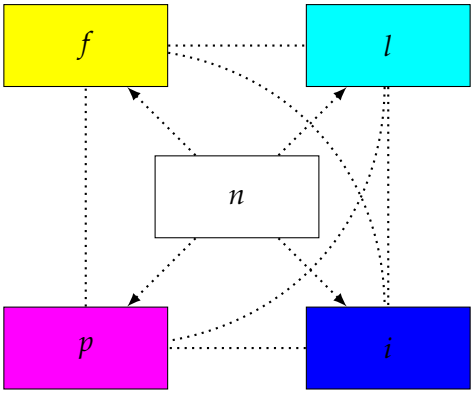


Figure 4.17: Example of cluster of proxy Elements. The AspectElements f, l, p and i are all proxy related, and projections of the NoAspectElement n.

(61) projection is an inverse functional interAspectRelation relation from a NoAspectElement to AspectElement.

(62) proxy is a interAspectRelation between two different AspectElements that are projections of the same NoAspectElement.

4.2.9.1 Requirement–Solution relations

Requirement–Solution relations are used to express that Elements may express requirements that other Element may satisfy and provide solution for.

fulfilledBy is a hierarchical relation between Element, such that fulfilledBy(a, b) expresses that b is intended as a solution to the requirements posited by a.

The following axioms are defined for Requirement–Solution relations:

Axioms.

(63) fulfilledBy is a subrelation of elementRelation.

(64) fulfills is the inverse relation of fulfilledBy.

4.2.10 Attributes

Attributes and AttributeGroups are Descriptors for assigning data values to Elements. The vocabulary for expressing Attributes is shown in Figure 4.18. An Attribute can be specified with a predicate, uom (unit of measure) and a value.

Attributes may be grouped into AttributeGroups, using the relation memberOf or its inverse relation hasMember. memberOf(a, b) expresses that a is member of b.

Both Attributes and AttributeGroups can be qualified by different classifiers (see Section 4.2.5.5) as a means to describe them.

Axioms.

(65) hasAttribute is a subrelation of reference between Element and Attribute.

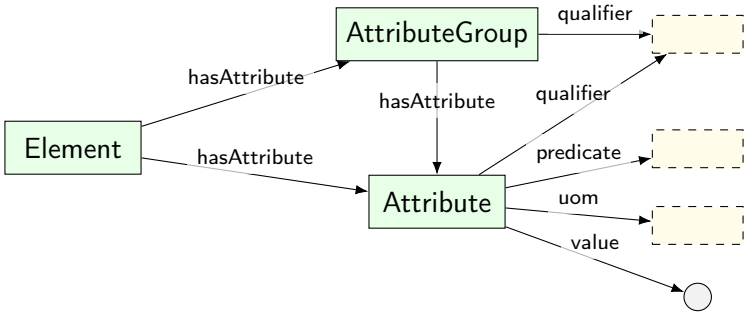


Figure 4.18: Attribute and AttributeGroups.

(66) uom is a subrelation of reference from Attribute.

(67) value is a datatype relation from Attribute to a value.

(68) predicate is a subrelation of classifier from an Attribute.

(69) An Attribute can only be member of an AttributeGroup, and an AttributeGroup can only have Attributes as members.

4.2.11 Other

The ontology contains a few other relations to describe Entity-s:

- symbol is a relation to associate a graphical representation, e.g., a diagram icon, with an Entity.
- color is an annotation relation that is used to associate a resource with a color for use in visualizations.
- notation is a relation that is used to associate an Element with an reference designation system (RDS) code.
- prefix is an annotation relation that is used to associate an Aspect with its RDS prefix.

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 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; the application of IMF modelling spans a much larger range.

5.1 What it Means to Create an IMF Model

To create an IMF model of the simple fluid separation system shown in Figure 5.1, means to specify the functional requirements, the spatial arrangement, and the specification of a realisation of this system. This is done 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), and assigning attributes with values. The level of detail that is put into it—the granularity of the model—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 of abstraction 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, e.g., an overall single line diagram will only contain information about the main pathways of the electrical distribution system.

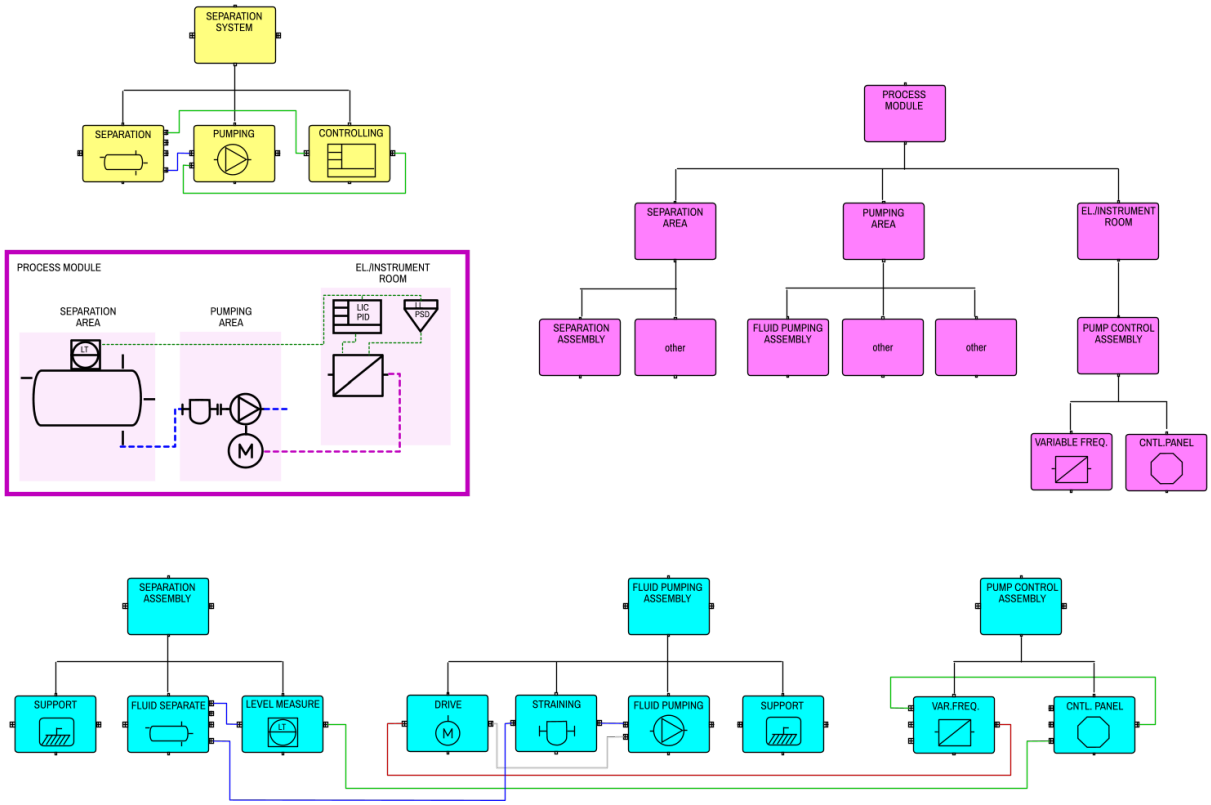


Figure 5.1: A separation system represented as an IMF model with three aspects: Function, Location, and Product.

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 a document-based to a model-based way of working there is a need to bridge between the two paradigms by referring to legacy document types when scoping a modelling exercise. For example, the scope for modelling is to define the same level of information content for the facility asset as typically represented in a process flow diagram (a document type). As the industry gains experience from the transition from document-based to model-based, new work processes can be developed that are directed towards continuously enriching an information model rather than producing fixed format information.

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

1. Defining needs: setting requirements.
2. Specifying solution: 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 characterised by means of their attributes, e.g., the 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 number 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 lifecycle. Figure 5.2 illustrates a separation system, shown as a conceptual diagram and as an imagined real solution.

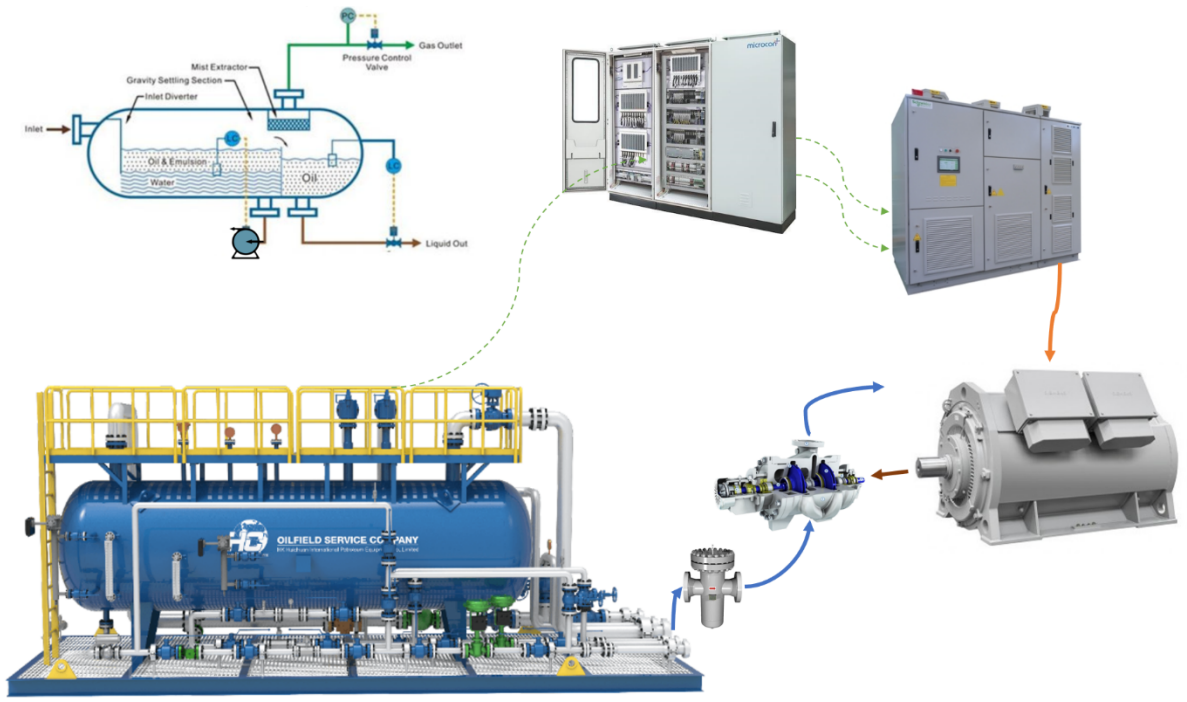


Figure 5.2: A separation system shown as a conceptual diagram and as an imagined real solution.

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. 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

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 [3] 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 when defining the scope, it is important to define the outer interfaces, that is, stating where does this model end (and where another model continues?) Such interfaces include upstream and downstream interfaces as well as interfaces between different disciplines, between aspects, or between IMF models. It is possible, but not recommended, 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.

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 a level of detail that solutions can be specified to fulfil them.

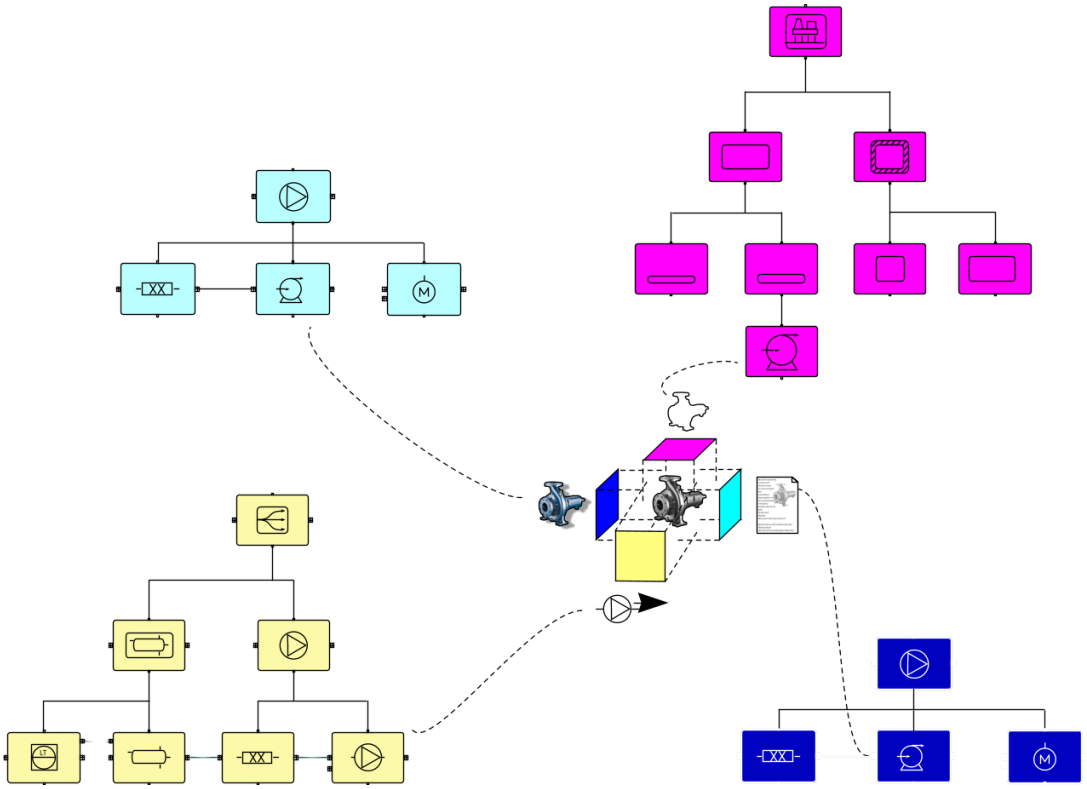


Figure 5.3: A pump system represented as an IMF model using four aspects: Function, Location, Product, and Installed.

5.3.2 The Product Aspect

Choose the Product Aspect when the objective is to specify solutions comprising components, equipment, or assemblies. Often is this 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 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 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. 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 in that 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.

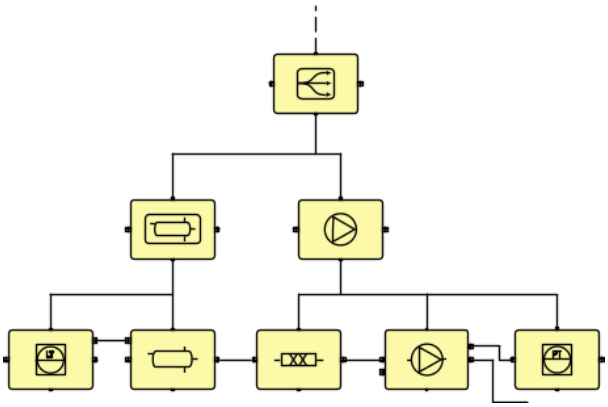


Figure 5.4: An example of an IMF model.

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

5.4.1 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

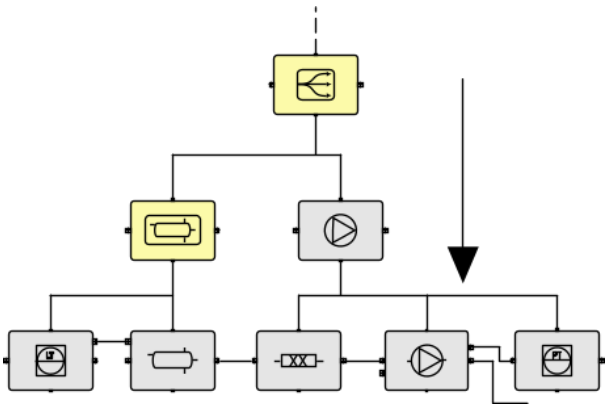


Figure 5.5: A Top-down modelling approach. (Grey colouring indicates blocks yet to be modelled.)

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 specialised expertise on individual sub-systems.

5.4.2 A Bottom-up Modelling Approach

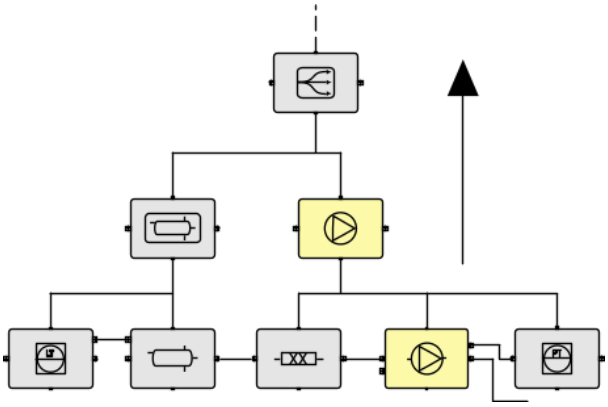


Figure 5.6: A bottom-up modelling approach. (Grey colouring indicates blocks yet to be modelled.)

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 predefined in some way, usually because they represent standardised or off-the-shelf systems.

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

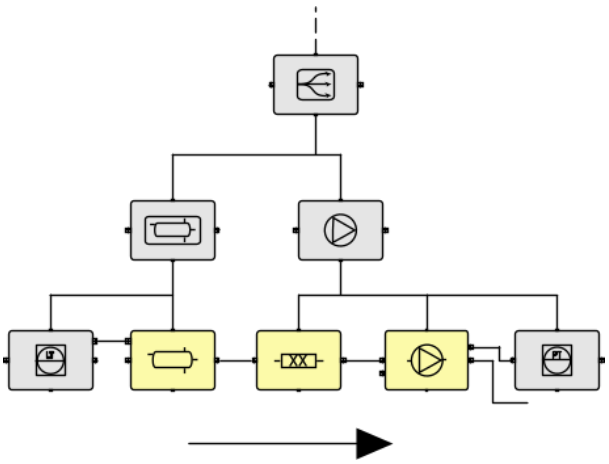


Figure 5.7: A follow-stream modelling approach. (Grey colouring indicates blocks yet to be modelled.)

from this approach is therefore a topology (horizontal connections). To finish such a model will require a bottom-up integration modelling effort.

5.4.4 A Follow-thread Modelling Approach

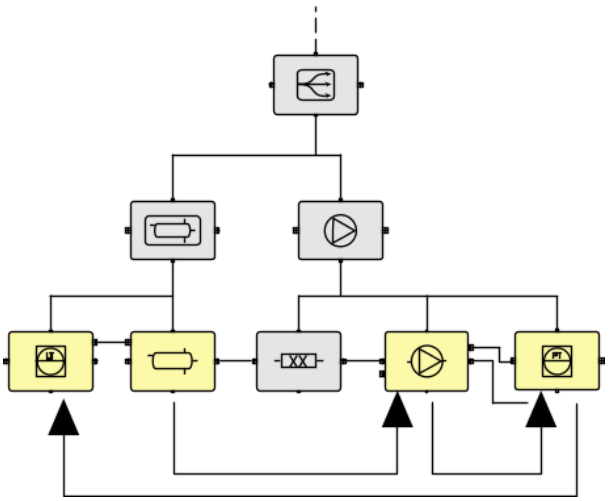


Figure 5.8: A follow-thread modelling approach. (Grey colouring indicates systems that do not have primary focus or that do not yet exist.)

Illustrated in Figure 5.8, the follow-thread approach is well suited for multidisciplinary modelling. The threads can be explained as:

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

The disciplines involved in this thread are process, process control, and instrumentation. What results from this approach is therefore a thread of defined dependencies between systems, and to finish such a model will require a bottom-up integration modelling effort.

5.5 How to Create and Connect Aspect Elements

The building blocks of an IMF model are the various types of aspect elements: blocks and terminals with aspects. The aspect elements are fetched from a library, or more precisely they are created or instantiated from IMF types that reside in a IMF type library. When an aspect element is instantiated from an IMF type, it has several attributes already defined, reflecting what it represents. For example, an aspect element Pumping should, e.g., have an attribute named Volumetric Flowrate and other attributes that characterises the pumping activity.

To compose a model requires creating aspect elements and connecting them. This is done by placing them into a breakdown hierarchy and topology, and by defining relations to aspect elements in other aspects. Depending on the modelling approach chosen, all of these different types of connections may not need to be defined initially.

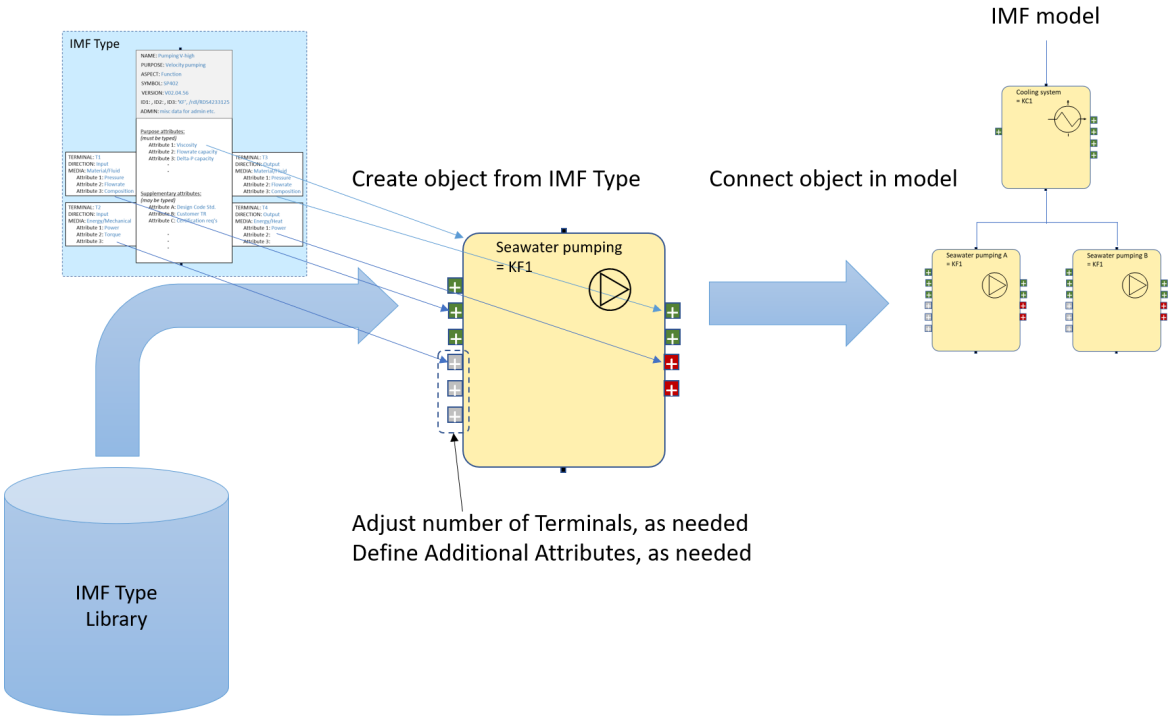


Figure 5.9: Using an IMF type from a library.

5.5.1 Selecting an IMF Type

The IMF type library will hold a large volume of IMF types, and they represent different levels of specialisation. 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 specialised pumping. When selecting the appropriate IMF type, it is therefore not sufficient to

only select the desired purpose (in the this example: Pumping), but also the desired level of specialisation must be chosen.

When selecting the IMF type, also 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 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 specialisation 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 Aspect Element into the Model

When an aspect element has been created it can be placed into the model by means of connecting it to other aspect elements. There are three main kinds of such placements/connections:

- Into the hierarchy of existing aspect elements, effectively creating hierarchical relations.
- Into the topology of existing aspect elements, effectively creating topology relations.
- Into the relation between aspect elements of different aspects, effectively creating inter-aspect relations.

The order in which the connections are made depend on the chosen modelling approach, but ultimately all must be made to complete a model.

5.5.4 Set Attribute Values of Aspect Elements

When a new aspect element is created from an IMF type it will have one or several attributes of a given quality. This is provided as part of the IMF type definition. Examples of qualities are 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 specialisation 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 aspect element, 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.

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.

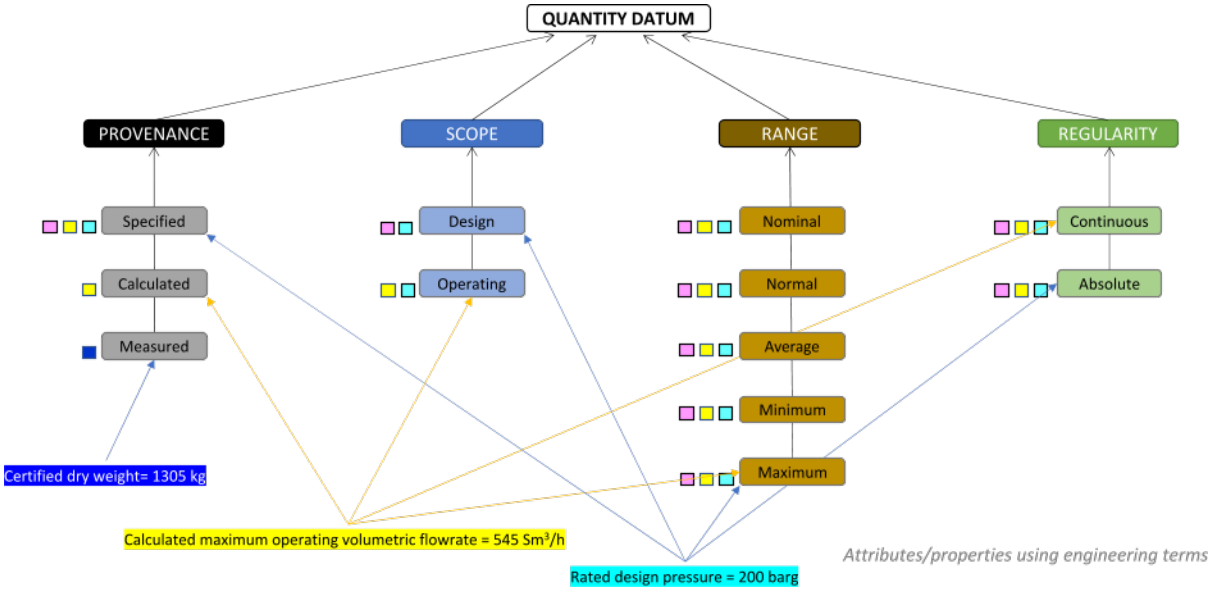


Figure 5.10: Typing of attribute values to provide context.

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 attribute 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 characterises 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 is created it will have one or several terminals. This is typically provided as part of the IMF type definition. A terminal can have a specified flow direction, e.g., Input or Output, 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 two input terminals with the media Energy/Electric.

5.5.7 Setting Attribute Values on the Terminals

Examples of qualities are Force, Voltage, Power, etc. It is part of the modelling process to increment as needed the number of attributes of the same quality for a terminal as needed, e.g., increment to two 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

Blocks will have terminals, and they are the means for modelling how streams flow. The 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 terminal and input terminal 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 Aspect Elements

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 aspect elements need to be edited and possibly also reconnected elsewhere in the IMF model. This allows for 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 (possibly far-reaching) effects of a change could be. A change to one aspect element could mean that other aspect elements 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 unwanted consequences.

5.5.11 Conditions for Shifting the Modelling Aspect

It is 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 recognised 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 a system is modelled in the different aspects. This is done using 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 logical first step is the relation which specifies that activity requirements modelled in the Function Aspect 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 logical second step 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; this is modelled

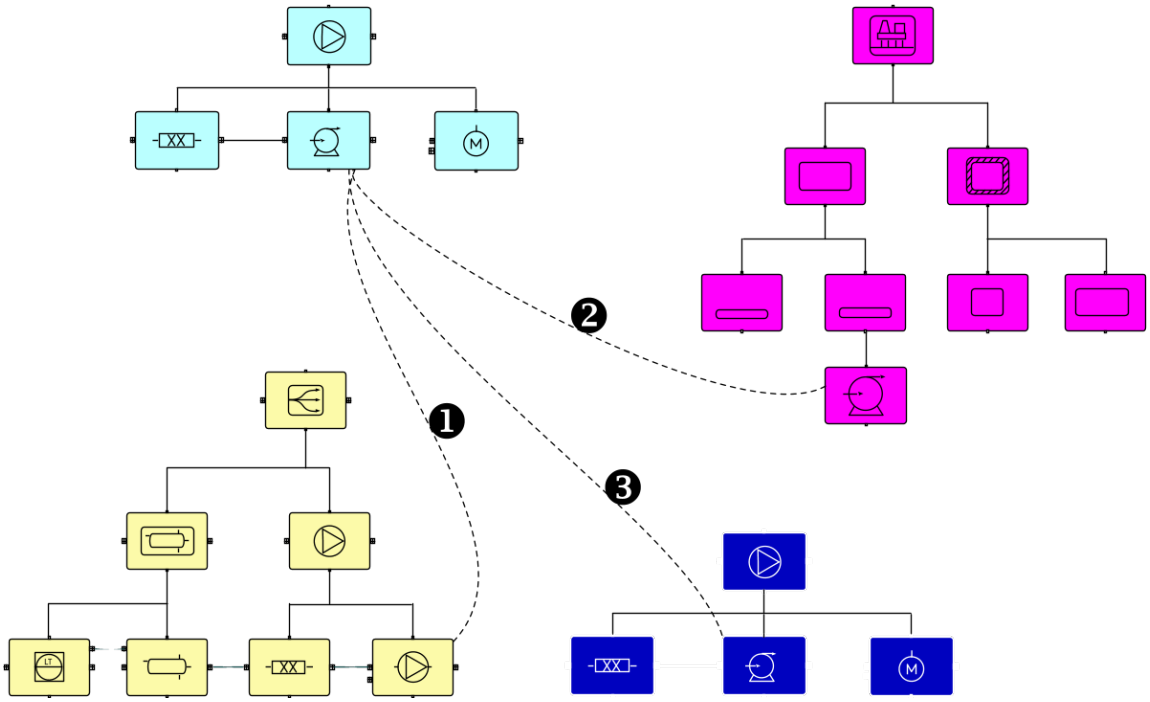


Figure 5.11: Inter-aspect relations.

in the Location Aspect. In this example the specified pump has a defined size and is relatively positioned in a defined location.

(b) That the solution specification modelled in the Product Aspect fulfils the location-based requirements modelled in the Location Aspect. In this example the specified area where the pump is located is specified as exposed to weather and is met by the pump being specified as weatherproof.

Note that for the Location Aspect 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.

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

Not all aspect elements will have relations to other aspects. A function block which is modelling a large system will almost never have a relation to a single solution specified as a product block. And an area or room or other such location modelled as a location block will almost never have a relation to a single solution specified as a product block.

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 representing the requirements, and the product block 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 representing the solution, and the location block representing the spatial properties of the solution, *and* by connecting the location block into the location block 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 representing the actual component, equipment, or assembly, and the product block 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 this is yet to be defined.

5.7 IMF Model Integration

IMF allows for 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 described for a later version of IMF. The integration can be between models in different aspects (1) or between different multi-aspect models (2).

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 sections, whereas integrating 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.

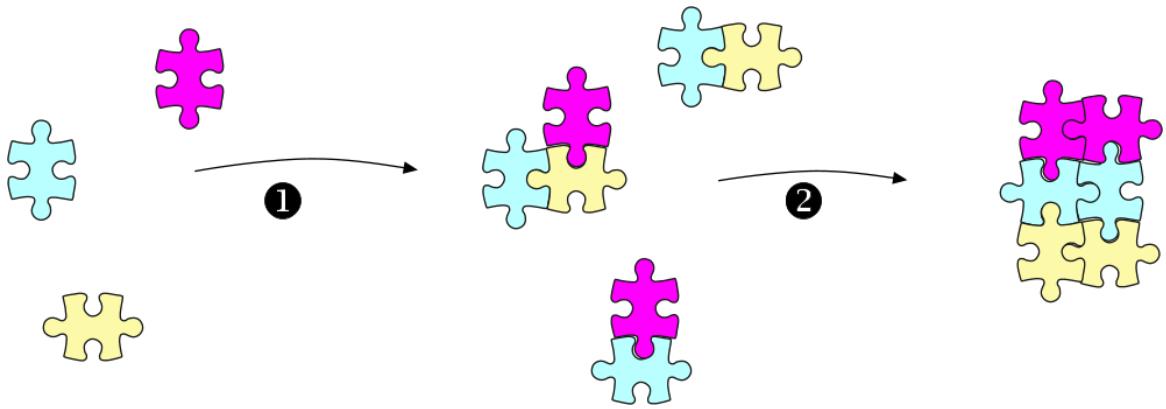


Figure 5.12: The concept of model integration.

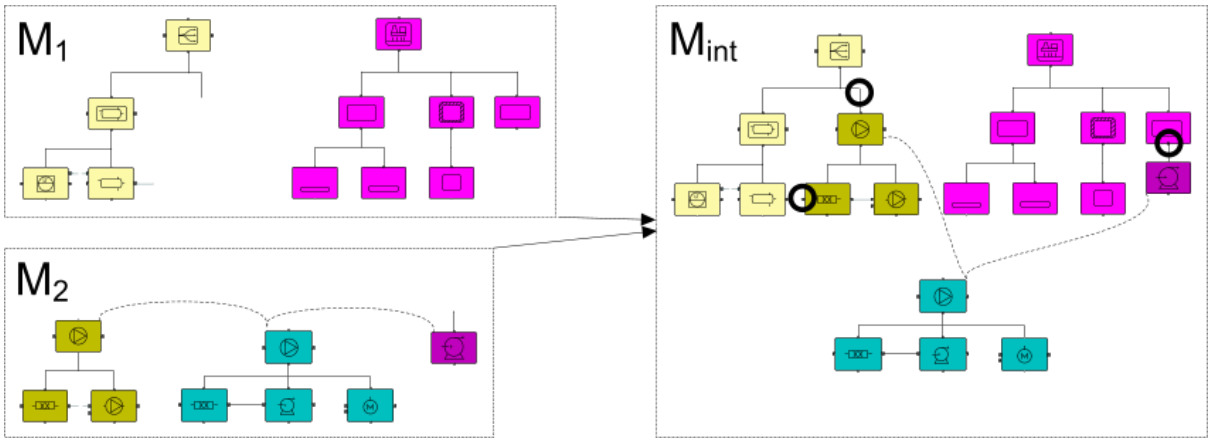


Figure 5.13: Model integration.

When 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 or 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 placed in the Function Aspect. The model is illustrated in Figure 5.14.

The function aspect hierarchy defines the system breakdown structure, and the streams define how the systems are connected together. Each of the function blocks define the requirements to what they represent, e.g., the Pumping function block defines the requirements to the pumping

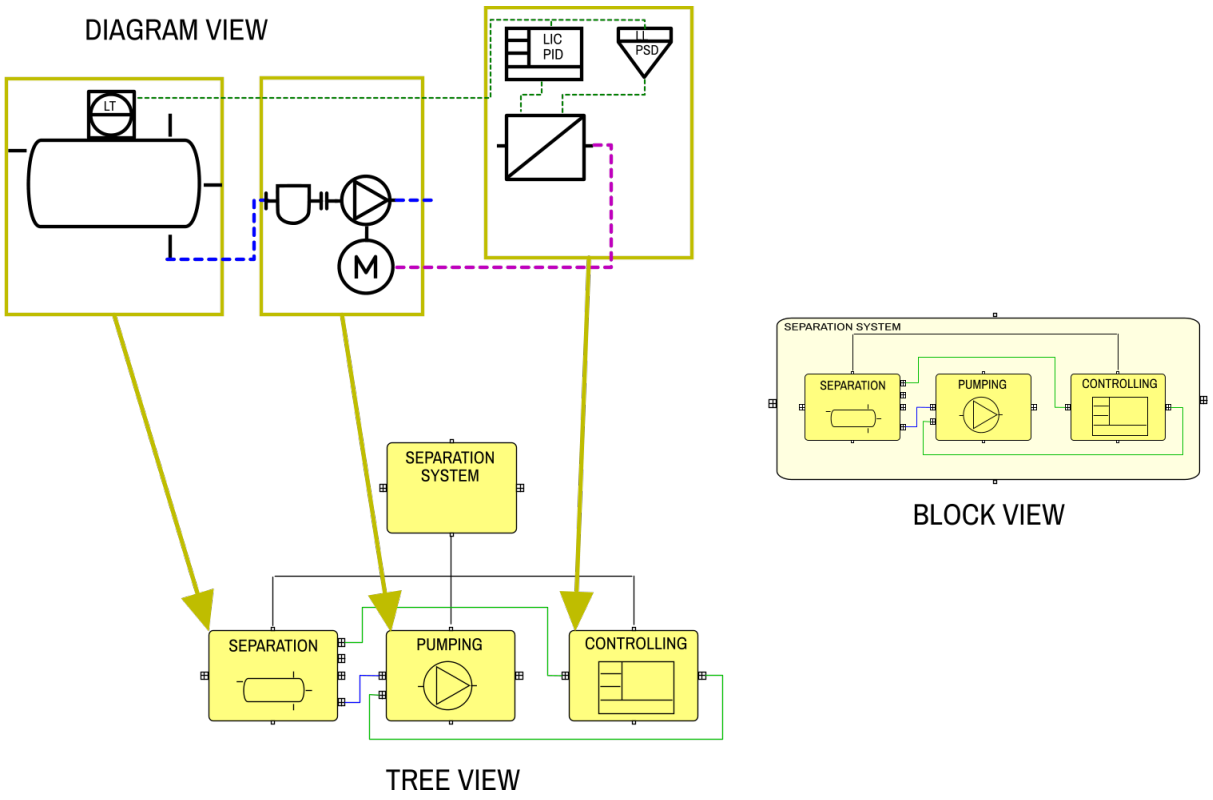


Figure 5.14: Separation system modelled in Function Aspect.

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, e.g., by setting the attribute 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. At this level the stated requirements could be limited to the required capacity to perform separation of crude oil of a specified quality.

To help interpret the figure, the same processing plant is shown in three different views: Diagram View is for similarity with the formats known from drawings such as process flow diagrams and system control diagrams. Tree View is a view of the model which emphasises the breakdown structure, and Block View is a view of the model which emphasises the topology (streams). If there is a need for further detailing the requirement, such as for the pumping performance, further breakdown of the pumping system could be modelled, e.g., by breaking down Pumping into Filtering, Pumping, and Driving, each represented by a function block.

5.8.2 A Multi-Discipline Requirements-Thread Model

This example is a process facility, and the emphasise 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 placed in the Function Aspect.

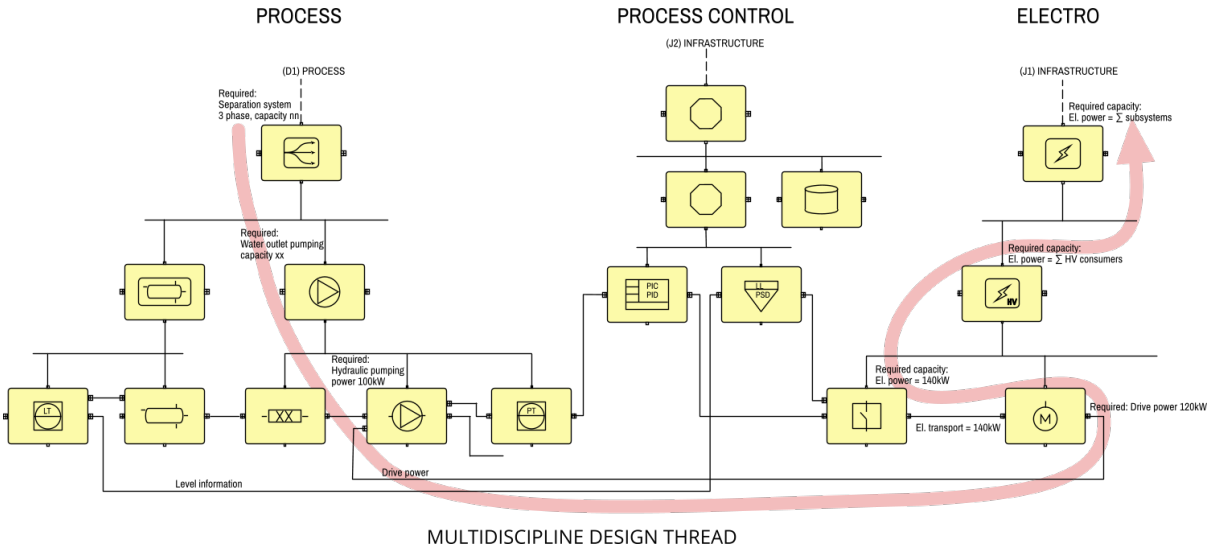


Figure 5.15: Modelling directed by the thread of requirements.

This particular requirement thread ties high level process requirements to high level electro requirements. Other requirement threads (not shown) similarly 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 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.

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, which provides a specification of that part, including input and outputs specified as terminals. 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. Motor Rotor has an output of type Energy/Mechanical/Rotating which here is connected to 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.

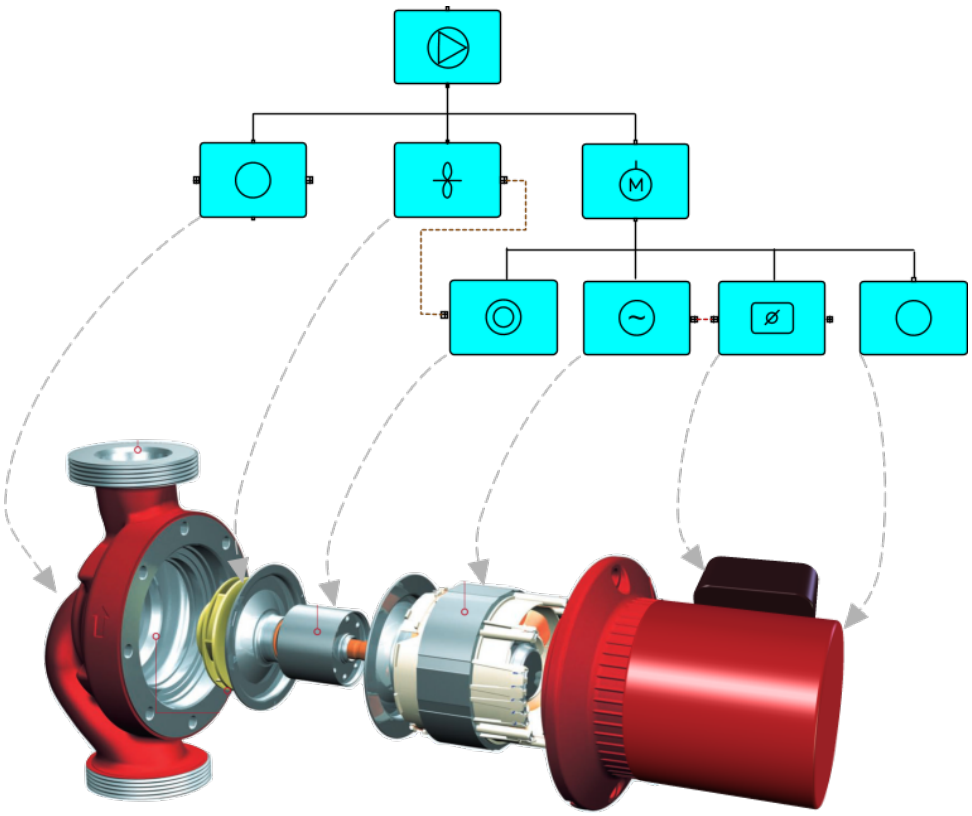


Figure 5.16: Pump product model.

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 breakdown 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.

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

Figure 5.18 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

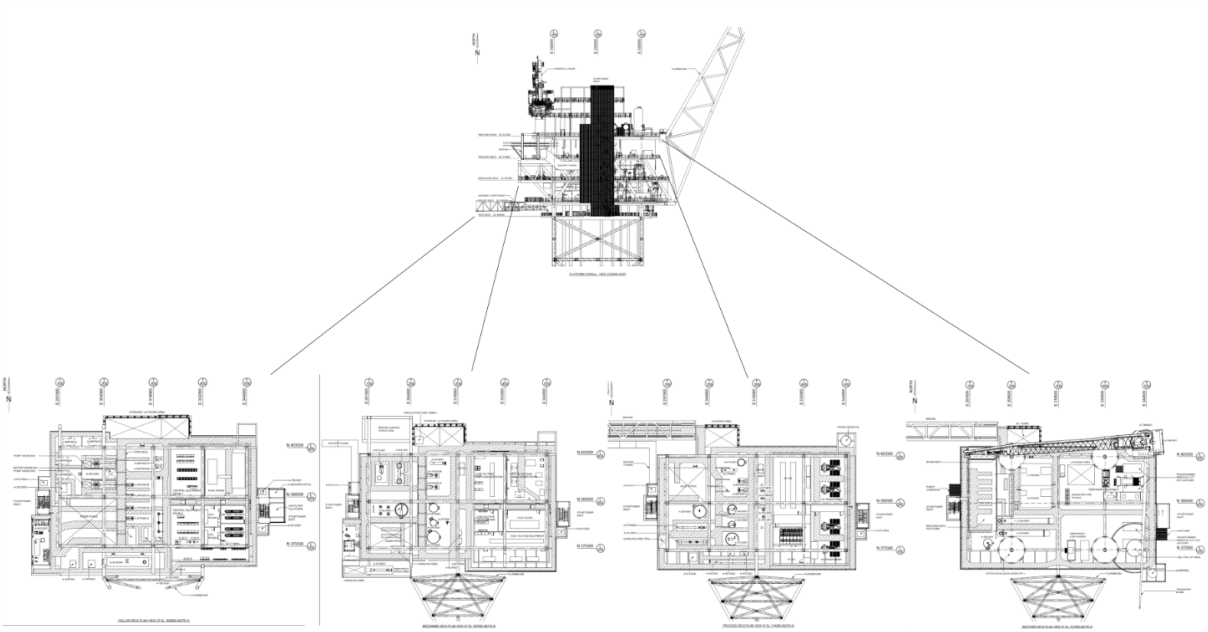


Figure 5.17: Plot plans for a platform.

example 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 preexisting solutions, or *design patterns*. Whenever it is an option to re-use preexisting designs, there is a need to do this as efficiently as possible. To accommodate this, *IMF complex types* offer a means to encode such typical design patterns in the form of a template—at the level of granularity and detail desired.

5.9.1 Design Patterns in Engineering

A design pattern 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 formalised 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 breakdown 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 breakdown structure may need to be included, but the individual

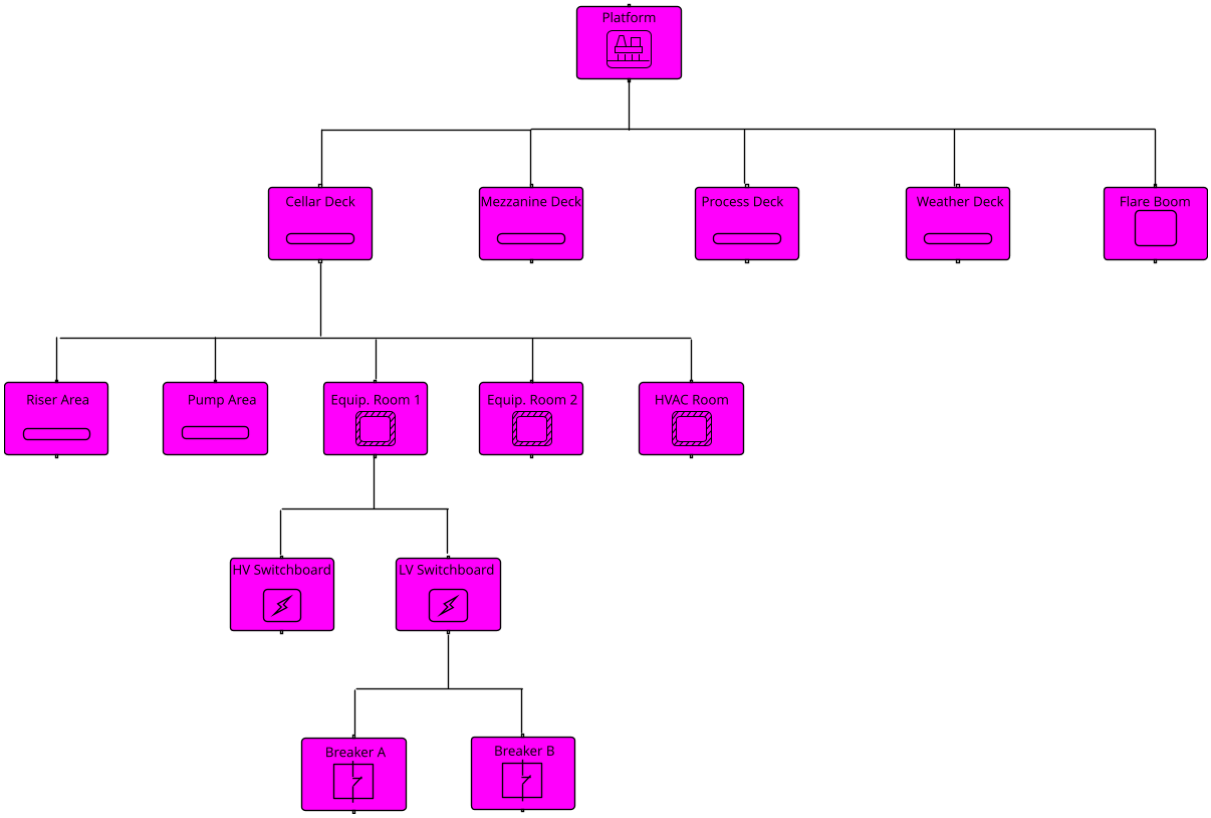


Figure 5.18: Model in Location Aspect.

elements in the breakdown structure may only require the attribute Purpose to be specified. Optionally type and attribute information can be included for the elements in the breakdown 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 in that they allow more than one aspect to be specified, and they allow specifying a breakdown 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 their main purpose, attributes and terminals, and furthermore specify the breakdown into Electric Motor, Shaft& Coupling, and Pump Unit—with their Purpose attributes set to 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.

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 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, breakdown 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, so 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 breakdown structure before details of the individual elements in the breakdown 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 breakdown hierarchy. This can be progressively enriched at later stages by deciding which IMF type each aspect element shall be, followed by deciding on terminals, attributes, and topology information. The breakdown 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 IMF as a framework and language is assumed. Understanding of reference data libraries is beneficial, but is 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 engineer's 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 applications, 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 specialised. 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 characterises '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 are mandatory and hence are part in defining the type, whereas other attributes are optional and offer 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 and terminals are created. Basic IMF types are *not* blueprints for

complete models, meaning they do not comprise blueprints for, e.g., a breakdown structure, rather they are restricted to templates for one block with terminals (in one aspect). IMF *complex types*, introduced later, allow for more complex patterns.

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 standardisation as well as minimising 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.

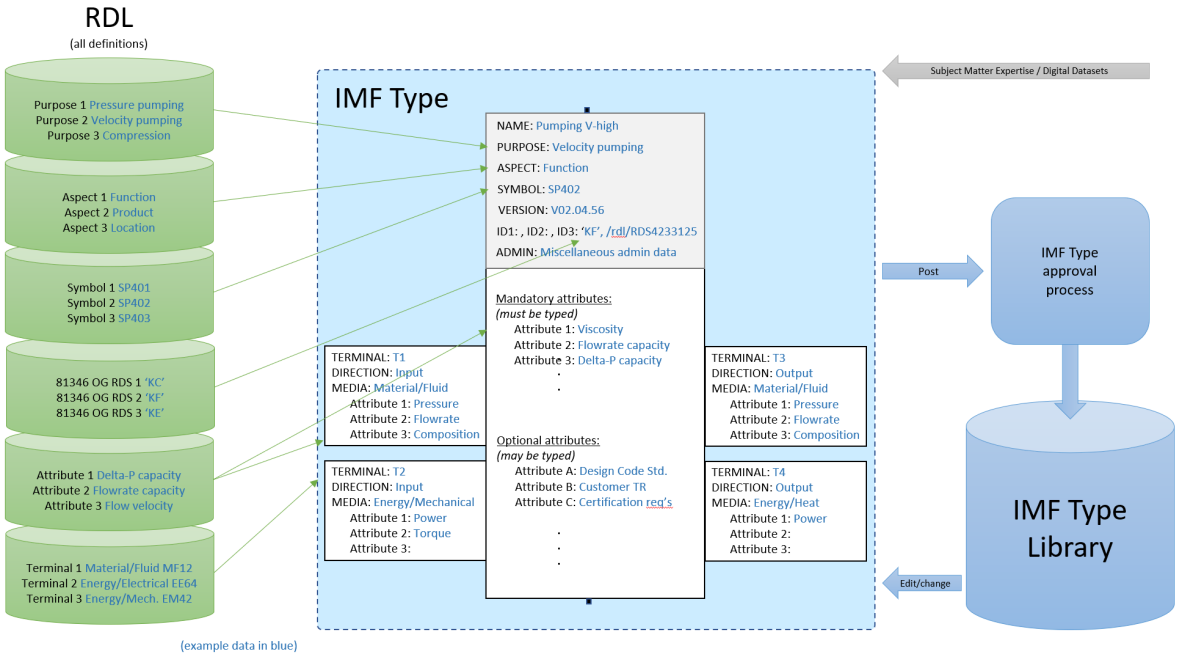


Figure 6.1: Creating and using a block with terminals from an IMF type

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 standardisation as well as minimising the duplication of work. IMF types are intended to reflect the need of SMEs for building blocks with which they can describe elements of a facility asset. Therefore, the primary input for defining an IMF type is taken from the SME. The means for compiling such domain knowledge could e.g., be datasheets (digital or conventional).

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 elements representing each aspect are needed, and consequently, IMF types representing different 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 new 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 mandatory attributes characterise the IMF type. Data contents for such attributes 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 Pumping to be specified at a minimum level.
2. Identify the mandatory attributes of other similar IMF types and make a comparison. There must be a difference in the set of mandatory attributes if the newly created IMF type should be created at all. As an example, if some other IMF type, also with the purpose Pumping, also has 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 additional mandatory attributes.

Optional Attributes Optional attributes are *not* essential in characterising 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 attribute to hold a reference to a pump curve document.

Additional Attributes Additional attributes are 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 (with a value of, 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 library 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 library, such that needs can be identified and incorporated. This should be supported by a tool and process where SMEs can propose RDL extensions as part of a defined work process.

6.3.2 Proactive versus Reactive Workflow

Maintaining a comprehensive IMF type library requires expert time and resources. Two approaches to this work are valid: proactive and reactive.

Proactive creation implies creating IMF types up front, with no specific facility assets in mind, but aiming at creating a useful set of IMF types irrespective of particular facility assets. To scope out what is useful requires understanding of which disciplines come 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 will have to be created in order to enable the modelling to progress.

6.3.3 Determining the Aspect of the IMF Type

When defining an IMF type with a given 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.

- **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.
- **Product:** In this aspect it is specified how to solve and fulfil a need by means of a component, equipment, or assembly.
- **Location:** In this aspect the spatial attributes are defined, including dimensions, relative location, and conditions and requirements within the space.
- **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 value for 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:

- **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.

- **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.

- **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 breakdown structure that is not identical to the physical breakdown, 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.

- **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 attribute 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. The qualities of attributes are selected from a reference data library.

6.3.6 Assigning the IMF Type to a Class

An IMF type definition may be a specialisation of a more general IMF type, which again is a specialisation of a still more general IMF type, and so on, i.e., there is a de-facto 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 similar to a subclass of the more general IMF type. A hierarchy of types will emerge as more and more specialised IMF types are created from the more general ones by copying and extending. This hierarchy will 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 specialisation (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 the input to and output from a block. The main properties for a terminal are direction (Input, Output, None/bidirectional) 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, a medium object specifically for Energy is referred to as an 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 fulfil this need are provided by mandatory attributes such as Volumetric Flow, 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.

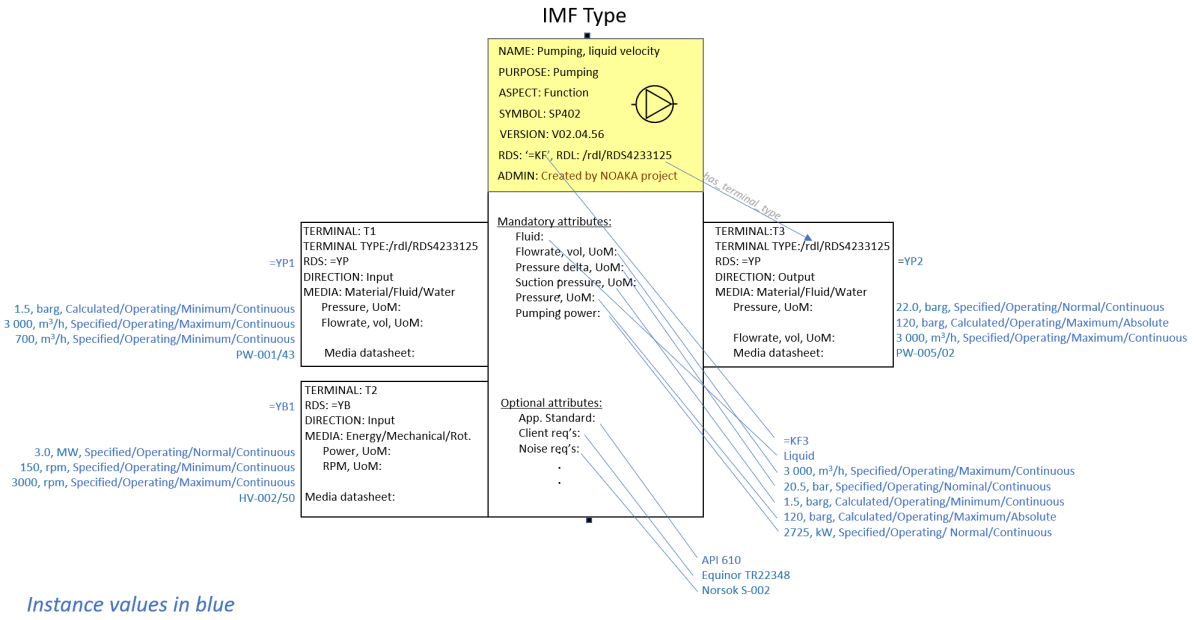


Figure 6.2: IMF type example of a Function Aspect.

6.3.9 IMF Type in the Product Aspect

In this aspect it is specified how to solve and fulfil 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.

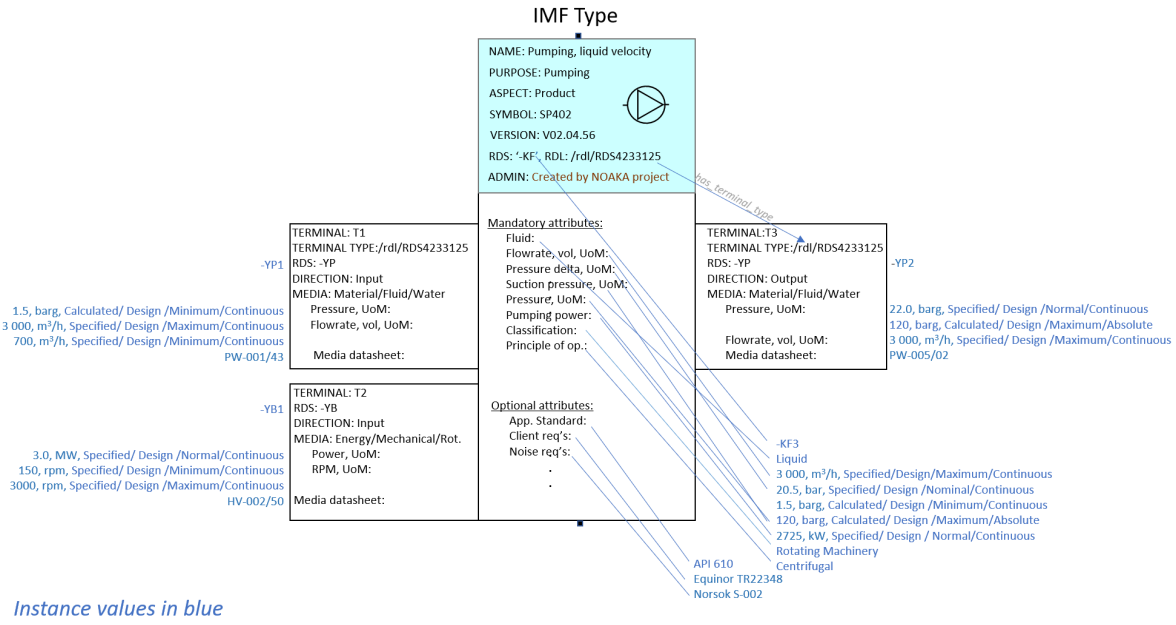


Figure 6.3: IMF type example of a Product Aspect.

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.

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 terminals record/document

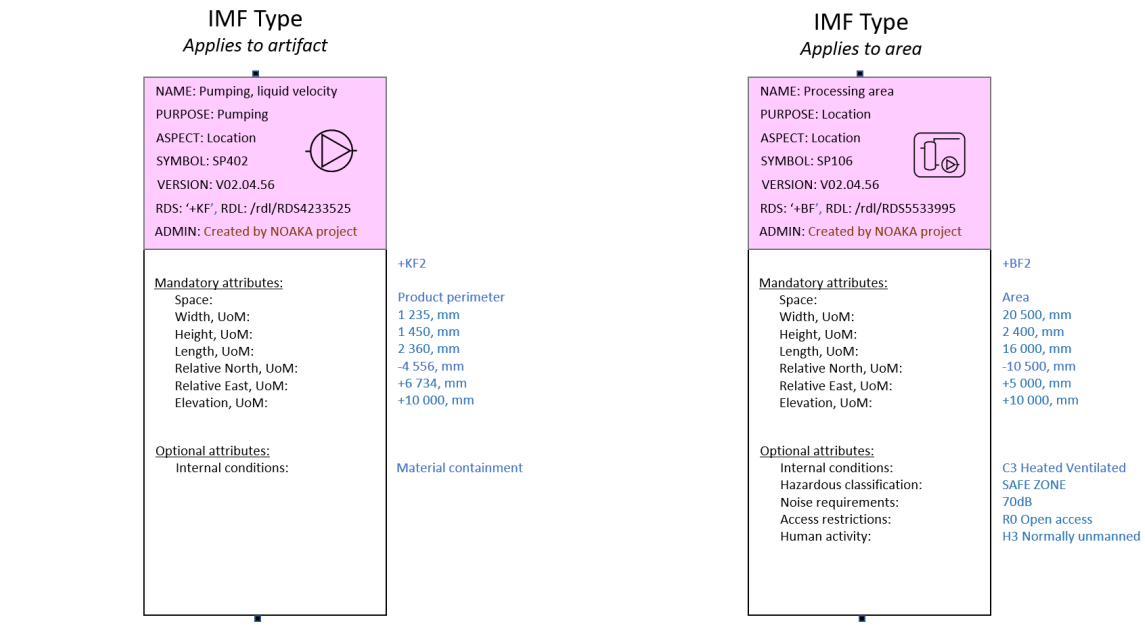


Figure 6.4: IMF type example of a Location Aspect.

the physical connection of inputs to or outputs from the delivered or installed product.

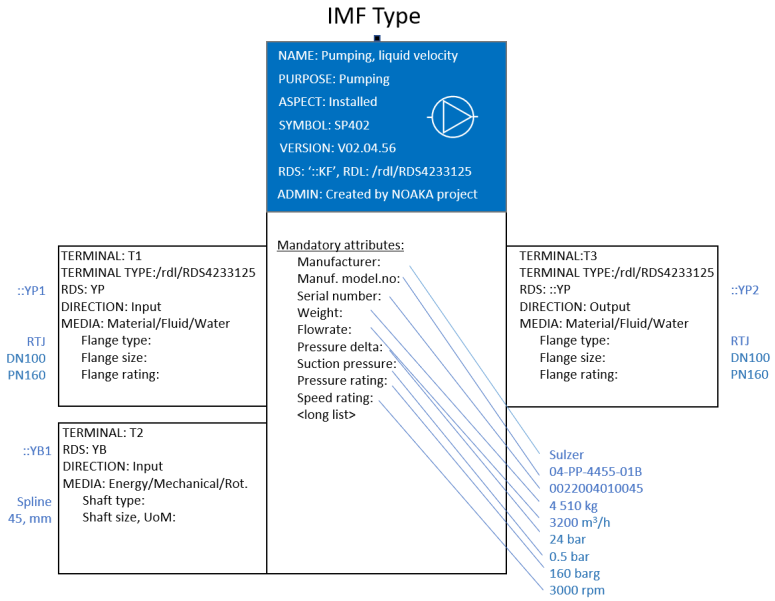
6.4 Using Reference Data Libraries

IMF models contain objects populated with attributes and relations according to the IMF types. Developing IMF types requires a proper source of standardised definitions of the different concepts an IMF model needs to utilise. Such sources are often referred to as reference data libraries. Developing 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 the ontology language OWL.

The PCA RDL contains a comprehensive definition of classes, properties and objects 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 and 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 utilise 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 loss of information.



Instance values in blue

Figure 6.5: IMF type example of an Installed Aspect.

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, as long as the extension are not in conflict with the type definition. A *closed* IMF type, on the other hand, only allows instantiating objects to set or contain the attributes and terminals that are explicitly permitted by the IMF type. The result is 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, an 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 breakdown 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.

6.7 [EXPERIMENTAL] Sourcing and inheriting attribute values

The default is that an attribute of an IMF type, whether it is a mandatory value or an optional value, is without value - because values are assigned to the instance during modelling. This is a very basic functionality which could be enhanced with the following mechanisms:

- IMF Type provides a default attribute value:
  - The default value is immutable (it contributes to defining the type)
  - The default value is a typical example value, and can be changed
- IMF Type provides constrains to what valid values can be assigned during modelling:
  - Equal to value of parent attribute of same attribute type
  - Larger than value of parent attribute of same attribute type
  - Less than value of parent attribute of same attribute type
  - In range of value range of parent attribute of same attribute type
- IMF Type provides a mechanism for linking the attribute value to any other attribute value such that it inherits this other value. The specific linking is part of the modelling work. The link may established to any other attribute of the same attribute type within the same IMF model.

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] *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.

[3] *'81346 - O&G - Reference Designation System for Oil and Gas*. Standard.

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

[5] *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.