



Conceptual model for safety requirements specification and management in nuclear power plants

Teemu Tommila | Jarmo Alanen



Conceptual model for safety requirements specification and management in nuclear power plants

Teemu Tommila & Jarmo Alanen

VTT



ISBN 978-951-38-8365-2 (URL: <http://www.vttresearch.com/impact/publications>)

VTT Technology 238

ISSN-L 2242-1211

ISSN 2242-122X (Online)

<http://urn.fi/URN:ISBN:978-951-38-8365-2>

Copyright © VTT 2015

JULKAISIJA – UTGIVARE – PUBLISHER

Teknologian tutkimuskeskus VTT Oy

PL 1000 (Tekniikantie 4 A, Espoo)

02044 VTT

Puh. 020 722 111, faksi 020 722 7001

Teknologiska forskningscentralen VTT Ab

PB 1000 (Teknikvägen 4 A, Esbo)

FI-02044 VTT

Tfn +358 20 722 111, telefax +358 20 722 7001

VTT Technical Research Centre of Finland Ltd

P.O. Box 1000 (Tekniikantie 4 A, Espoo)

FI-02044 VTT, Finland

Tel. +358 20 722 111, fax +358 20 722 7001

Preface

The Finnish Research Programme on Nuclear Power Plant Safety (SAFIR2014) was one in a series of national activities to develop and maintain research capability, safety assessment and nuclear safety expertise of Finnish nuclear power plants. Within the SAFIR2014 programme, the SAREMAN project (Safety requirements specification and management in nuclear power plants) developed modelling concepts and practices for requirements engineering with the focus on instrumentation and control systems. The report at hand is the final release and summary of the work carried out at VTT on nuclear power plant vocabulary during 2011–2014. The work was started by analysing and clarifying fundamental concepts of power plant systems and their life-cycles in co-operation with Aalto University. In order to bring the general modelling principles closer to design practices and tools, related information models developed by VTT for risk assessment of machine automation were adapted and integrated to the report in 2014. The result now consists of two main parts. The first one aiming at a shared understanding and vocabulary on a general level was primarily written by Teemu Tommila. The second part, mostly formulated by Jarmo Alanen, provides a limited but more practical data model for tool developers.

The project was funded by the Nuclear Waste Management Fund (Valtion ydinjätehuolto-rahasto, VYR), VTT and Aalto University. Some financial support was also received from TVO, Fortum, Fennovoima and Posiva. During the four project years, the results were reviewed and the work guided by the members of the Reference Group 2 (Automation and control room) of the SAFIR2014 programme. The authors of this report are grateful for all the valuable comments and ideas.

More information about SAFIR2014 can be found on <http://safir2014.vtt.fi>. National research on nuclear power plant safety continues in the SAFIR2018 programme for the years 2015 – 2018, see <http://safir2018.vtt.fi>.

Contents

Preface	3
1. Introduction	6
PART I – ON THE PRINCIPLES OF SYSTEMS MODELLING	8
2. Basics of Requirements Engineering	9
2.1 Requirements, what are they?	9
2.2 What is Requirements Engineering?	12
3. On the principles of systems modelling	14
3.1 What is a system?.....	15
3.2 What is a system model?.....	18
3.3 Physical structure of a system	21
3.3.1 System elements	21
3.3.2 Connectivity of systems and system elements.....	23
3.3.3 Spaces and locations.....	24
3.4 System behaviour	25
3.4.1 Activities.....	25
3.4.2 System functions	26
3.4.3 Connectivity of system functions	29
3.4.4 States and modes.....	30
3.4.5 Events.....	32
3.4.6 Scenarios.....	32
4. Life-cycle modelling	34
5. Meaning and representation of requirements	39
5.1 Understanding requirements as statements	40
5.2 Associating requirements with the system model.....	43
5.3 How to express requirement statements	45
6. Traceability relationships	48
6.1 Recording modifications.....	51
6.2 Recording designer’s reasoning	52
7. Modelling nuclear power plant systems	54
7.1 Observations on the current terminology.....	54
7.2 Modelling NPP system structure and behaviour.....	55
7.3 On safety aspects in NPP system modelling.....	59
7.4 Modelling the I&C system life-cycle	61

8. Summary	67
PART II – IMPLEMENTABLE ARTEFACTS MODELS	68
9. Engineering artefacts models.....	70
9.1 Introduction	70
9.2 Project data repository core model	71
9.3 System package.....	76
9.3.1 System artefacts model	77
9.3.2 Physical architecture package.....	77
9.3.3 Behaviour package	81
9.3.4 Properties package.....	84
9.3.5 Product type package.....	85
9.4 System context package.....	86
9.4.1 System context artefacts model	86
9.4.2 Operational environment model	87
9.5 Requirements and V&V package.....	87
9.6 Specialty engineering package.....	89
9.6.1 Risk assessment package.....	90
9.7 Projects package.....	93
10. Traceability information model.....	95
10.1 Traceability demonstration.....	97
11. Summary and conclusions	113
References	115
Appendix A: Acronyms and abbreviations.....	121
Appendix B: Glossary.....	123
Appendix C: Artefact types	135

1. Introduction

Errors during the early stages of design have been widely recognised as a major source of problems in many engineering disciplines. The history of industrial automation suggests that most control system failures may have their root cause in an inadequate specification. As much as 40% of faults contributing to the incidents can be attributed to the requirements specification (Chambers et al. 1999, HSE 2003). Several other studies, summarised by Fanmuy et al. (2011), have clearly underlined the importance of requirements as an area of *Systems Engineering* (SE). Requirements engineering is a key success factor for development of complex products. Requirements engineering has been promoted for more than 20 years by standardisation agencies and various organisations such as the International Council on Systems Engineering (INCOSE). Despite the increasing maturity, requirements engineering often remains poorly understood and implemented. Mere production of huge sets of detailed requirements does not ensure good project performance. The study on industrial requirements engineering practices by Fanmuy et al. (2011) revealed that one of the common problems is that requirements are often written in natural language, which often leads to ambiguity and too complex sentences. Obtaining consistency and completeness of requirements remains difficult by only human-visual review since thousands of requirements need to be managed in most of cases. **Figure 1** lists typical defects in requirements. As seen from the figure, even the distinction between requirements and their solutions is not clear to many designers.

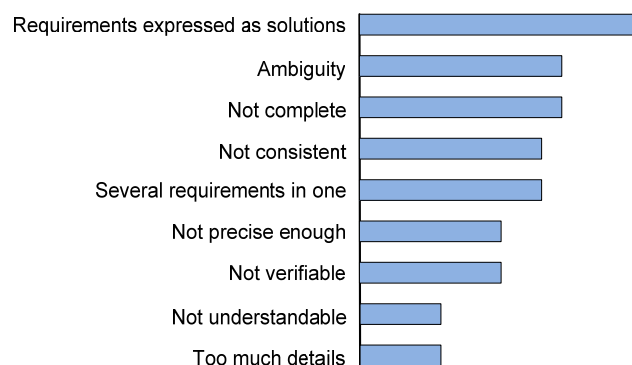


Figure 1. Most common defects in requirements (adapted from Fanmuy et al. 2011).

Potential explanations of the problems include lack of information, continuous changes and the abstract and multi-disciplinary character of the specification phase. Even when significant uncertainties are present, engineers tend to think in terms of technical solutions rather than what is actually needed by the users. The terminology, working methods and tools for collecting and describing requirements and keeping them up-to-date are often vague. The development path from high-level requirements to abstract system functions and finally to their technical imple-

mentation may not be systematic and well-documented. The lack of design traceability makes it difficult to ensure that the requirements are fulfilled in the final implementation. Also in process control, the early stages and collection of input information are considered important and challenging. However, requirements engineering is not seen as a dedicated design area. Explicit requirements can be seen, e.g., in contractual documents, but the specifications written by engineers often focus on technical issues. Safety critical applications are controlled by regulatory authorities. This emphasises the importance of requirements engineering. Unfortunately, safety standards do not give much practical advice (Tommila et al. 2011).

The primary aim of this publication is to promote systematic Systems Engineering (SE) and Requirements Engineering (RE) practices in the design, construction and maintenance of nuclear power plants (NPP). Our focus is on the design stage and, in particular, on monitoring and control of safety-critical processes and systems of a NPP. In addition to Instrumentation and Control (I&C) systems, we will consider requirements originating from the equipment under control and from human operators. Our research approach is based on the following claims:

- Requirements are statements concerning characteristics of entities encountered in the application. Therefore, requirements cannot be discussed in isolation from other engineering activities and system models. Even the boundary between requirements and design solutions is not always clear.
- By its nature, RE is a multi-disciplinary activity involving several stakeholders. Consequently, a technical system, e.g. an I&C system, should be seen in their context as part of a larger, enclosing system including both technical and human elements.
- Due to the large number of requirements and their dependencies in complex systems, and the need to revise and maintain them throughout the system life-cycle, computer tools will be needed in the future. Furthermore, those tools should be integrated with other engineering environments.
- Well-defined terminology is needed for successful communication and collaboration between various stakeholders and engineering disciplines. For computer tools and electronic exchange of design information a vocabulary is not enough but also formal and standardised data models (ontologies) are required.

In particular, the necessity of clear terminology as a foundation of design practices was the reason behind this report. Termed a “conceptual model” this report suggests a set of definitions of modelling concepts, their properties and relationships suitable for the NPP domain. Some aspects of the conceptual model may seem rather theoretical, but the rationale comes from the future need of computer tools. Consequently, our conceptual model does not concern the philosophical questions of what exists in the real world. Rather, it tries to define model elements (information items in the design database) that can be used to describe the current and future worlds. In order to support current practices, the glossary in appendix B gives common-sense definitions of key terms. They are intended to foster mutual understanding among industrial professionals. Furthermore, they are a basis for design guidance and tool development in the future.

This document is divided into two main parts. Part I discusses informally the basic principles of design and systems modelling covering topics like requirements engineering in general, the physical structure and behaviour of systems, system-life cycle, the meaning and representation of “requirements” and traceability of design information. Part II represents a more practical information model that provides a basis for tool development in the industry.

PART I – ON THE PRINCIPLES OF SYSTEMS MODELLING

The fastest path to a high-quality requirements and systems engineering follows improvements in working practices and training of designers. We claim here a that a shared and logical way of thinking about systems and design in general should be taken as the first goal and that successful introduction of computer tools is possible only on top of it. The purpose of this Part I of the publication is to give a short introduction to Requirements Engineering and to describe a few fundamental issues of modelling man-made systems and their life-cycles. A more structured information model is suggested in Part II.

Chapter 2 gives a short introduction to the basics of requirements engineering. The role of Chapters 3 to 6 is to outline our fundamental and generic principles of system modelling, design processes, requirements and traceability. The ideas are then refined in Chapter 7 for describing nuclear power plant systems and their life-cycle.

2. Basics of Requirements Engineering

As characterised by INCOSE (2007), Systems Engineering (SE) is an interdisciplinary approach and means to enable the realisation of successful systems. SE is an iterative process that looks both at the social and technical aspects of entire man-made systems. While focusing on engineering design, Systems Engineering includes also management processes and overlaps with the profession of Project Management (PM). With its technical processes, SE covers, for example, requirements definition and analysis and verification and validation of design solutions (ISO/IEC/IEEE 15288 2015, INCOSE 2007). So, Requirements Engineering is an essential part of Systems Engineering.

2.1 Requirements, what are they?

According to many standards and textbooks, a requirement is a condition or capability that must be met or possessed by a system or system element to satisfy a contract, standard, specification, or other formally imposed documents. A requirement is a statement of a stakeholder, e.g. a customer or a user, concerning a product or its development process. A requirement is usually realised by another stakeholder, e.g. by a supplier or a designer. Consequently, requirements are basically concerned with the communication between stakeholders in an engineering project. This communication chain continues from the customer to programmers and installation technicians. The outputs of one step can be considered as requirements for the next one. However, part of this material is not actually requirements but problems, facts and design decisions already made. This makes it difficult to define the term requirement exactly – the distinction between requirements and design solutions is not always clear.

A requirement may concern, for example, an entire socio-technical system, such as an industrial plant, a process control system or a single component of it. Therefore, requirements should not be treated in isolation from their targets but attached to an “integrated system model”. In other words, requirements can be seen as one aspect of a system. In the very beginning, only an idea of a new system may be available, and high-level requirements are attached to it. When details are added to the system model, more refined requirements can be attached to them. Refined requirements are derived from higher-level requirements, e.g., by decomposing them or allocating them to known system elements. Additional information sources, such as standards and domain experience, are essential here. However, this flow-down of requirements is not only top-down but existing implementation or selected solutions can result in iterations between different levels of requirements. As a result, the model of a socio-technical system includes a hierarchical network of requirements attached to system elements at various levels of decomposition (**Figure 2**). The traceability relationships between requirements, as well as all other kinds of model elements, allow one to explain the rationales behind the decisions made, to track the chain of previous version and to identify model elements affected if a requirement must be

modified. Without this traceability it is practically impossible to give a trustworthy argumentation for that requirements have been met, e.g. that a control system is safe.

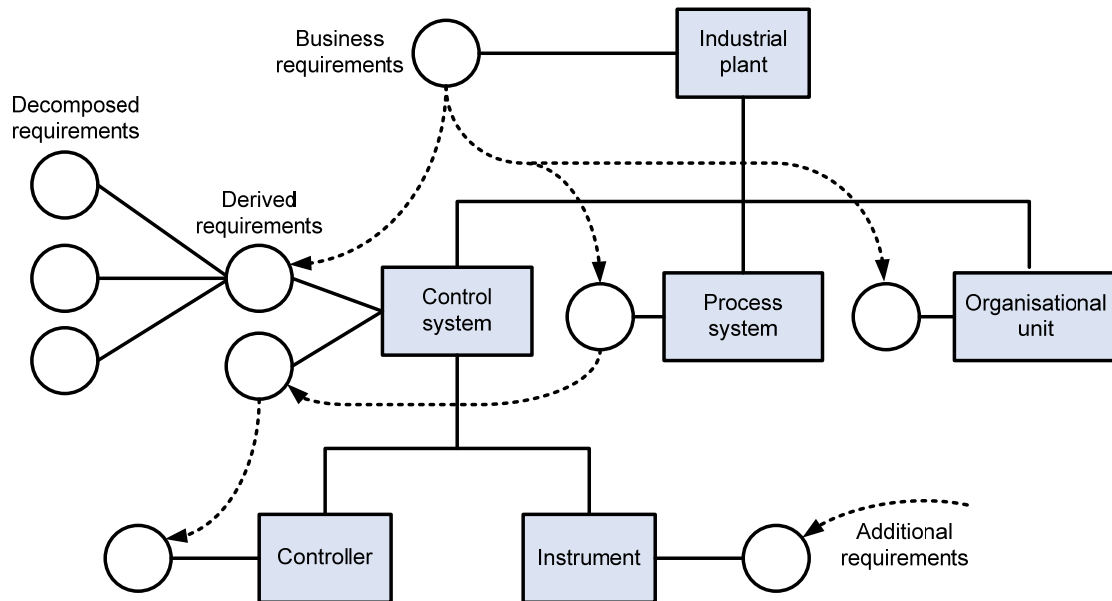


Figure 2. Requirements attached to various elements in an industrial plant hierarchy (Tommla et al. 2011).

Requirements can be classified according to several criteria. One of the most common is the distinction between functional and non-functional requirements. A *functional requirement* specifies what a system is supposed to do, such as a service that a system must be able to provide, without considering its implementation. A *non-functional requirement* describes, e.g., system structure or tells how well a system performs, i.e. its expected properties, such as accuracy, response times, usability, maintainability and reliability. Furthermore, different requirement types can be identified according to whose interests they represent. *Stakeholder requirements* look at the system in the context of its operational environment and specify what the stakeholders want to do without concerning any technical details. For example, business requirements, user requirements and many requirements from regulatory authorities are subtypes of stakeholder requirements. *System requirements* correspond to stakeholder requirements that are allocated to the system being designed. They are usually specified by engineers and re-formulated in technical terms. These requirements concerning the system as a whole are then further allocated to software requirements, hardware requirements and requirements associated with operation and maintenance of the system. A system is usually part of a larger system and has to interact with external entities like manufacturing equipment, other computer systems and human users. So, the external interface forms an important type of requirements that, depending on their content, may concern a system's behaviour, implementation or performance.

It should be noted that requirements can be attached not only to the future system but also to its development process. Life-cycle requirements are often called process requirements and seen, for example, in project and quality plans. They can set demands for the competence and qualifications of the designers or limit the way the development work is carried out, e.g. by requiring certain tools, documentation methods or software development standards to be used.

Regardless of its type, a requirement should maintain the stakeholder's viewpoint and leave sufficient freedom to those taking care of the next steps. Over-specified and premature requirements, e.g. insisting a specific hardware platform, are often unnecessary design *constraints*. Sometimes they are needed, for example when the solutions are specified in laws and regulations or decided by other disciplines. Otherwise however, constraints may hinder the designer from delivering a good solution to the user's needs.

Table 1. Features of good requirements (adapted from ISO/IEC/IEEE 29148 2011, Garcia & Fleisher 2010).

Correct:	The requirement corresponds to the essential needs stated by stakeholders and specifies a characteristic of the system meaningful and observable to them. If it is removed or deleted, a deficiency will exist. The requirement avoids placing unnecessary constraints on the design.
Feasible:	The requirement can be implemented with available technology and within the constraints of the project.
Consistent:	The requirement does not contradict with any other requirement or accepted design decisions.
Singular:	The requirement addresses one and only one thing.
Complete:	The requirement is fully stated with no missing or implied information.
Understandable:	The requirement is stated with well-formed expressions without technical jargon, acronyms or graphics unfamiliar to the intended audience.
Unambiguous:	The requirement has a clear logical structure and it refers consistently to the elements in the system model.
Allocated:	The requirement uses active voice, such as 'the system shall be able to select'.
Mandatory:	The requirement represents a characteristic the absence of which will result in an unacceptable solution.
Traceable:	All relationships of the requirement are identified such that the requirement is linked to its source and rationale, as well as to the implementation.
Prioritised	The requirement contains information on its importance with respect to the goals of the stakeholders, for example system economy and safety.
Verifiable:	The implementation of the requirement can be determined through inspection, analysis, demonstration or test.
Current:	The requirement has not been made obsolete by the passage of time.

A good requirements specification is complete, unambiguous, internally consistent, well-structured, understandable and traceable. As commonly suggested in the literature, good requirement statements should have characteristics shown in **Table 1**. It is easy to see that most of those features are not limited to requirements but apply to other kinds of design artefacts as well. To fulfil their communicative function, requirements should be understandable for all rele-

vant stakeholders, depending, however, on the level of detail and intended audience. When expressed in natural language, descriptions should be written by using simple structures and concise domain-specific terminology. In addition, various more structured techniques, such as use-cases, class diagrams, data flow diagrams, or even formal methods can be used to clarify what is wanted.

2.2 What is Requirements Engineering?

As characterised by ISO/IEC/IEEE 29148 (2011), Requirements Engineering (RE) is an interdisciplinary activity that mediates between the domains of the customer and supplier to establish and maintain the requirements to be met by a system, software or service of interest. This seems to include the idea that requirements engineering is limited to the early stages of a system's life-cycle. However, requirements do not end to user and system requirements but may go to details that may emerge during all design steps. Therefore, the life-cycle includes a chain of several "customers" having needs and "suppliers" providing solutions to those needs. Moreover, requirements may change and must be maintained also during system operation. RE is an iterative process that covers a system's whole life-time and is closely intertwined with other engineering activities. So, defining exactly what occurs in RE is difficult.

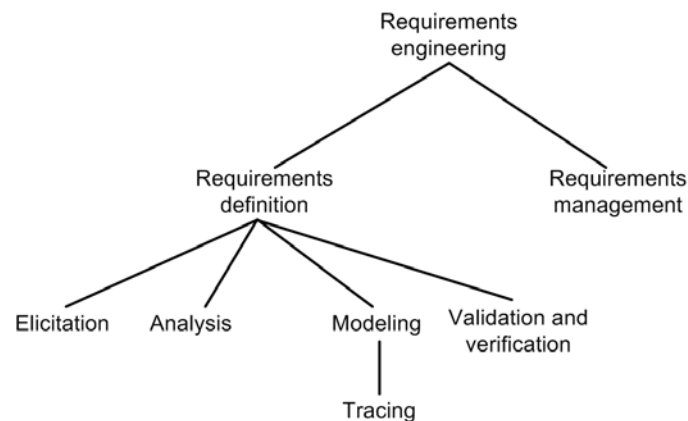


Figure 3. Main activities of requirements engineering.

We see RE as one "discipline" and thread of activities within the overall design process. RE defines requirements and attaches them to a shared model of the system being developed. RE is divided here into two main activities, *requirements definition* and *requirements management* (**Figure 3**). Requirements definition includes discovering, analysing, documenting (modelling) and validating and verifying the requirements. Analysis includes negotiation and prioritisation processes that lead to a set of agreed requirements. Requirements definition also includes maintaining the traceability of requirements. Requirements management, in turn, can be seen as part of *configuration management*, e.g. identification, traceability assessment and change control applied to requirements. In summary, RE is an engineering process that produces a consistent set of requirements that is then used in many other activities such as detailed design and test planning (**Figure 4**). Quality management and project management, on the other hand, are organisational processes that guide and monitor the work of the project organisation. The quality management organisation, in particular, should be independent of the design staff.

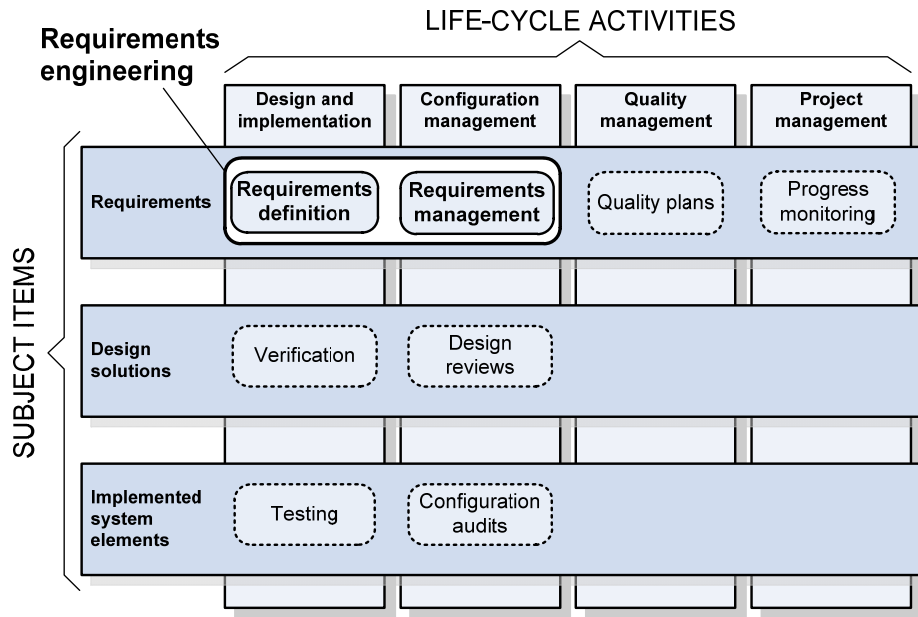


Figure 4. Requirements engineering produces and maintains the requirements but they are used in many other life-cycle activities (modified from Raatikainen et al. 2011).

In general, design should proceed in a top-down fashion, although iteration necessary, today even desired, for example in agile software development. According to the requirement levels outlined in the previous section, modelling should start with the integrated socio-technical system like a process plant. This overall model, sometimes called *Concept of Operations* (ConOps) (IEEE Std 1362 1998, INCOSE 2007, ISO/IEC/IEEE 29148 2011) describes the way the main elements collaborate in order to achieve their common goals (further information can be found in Tommila et al. 2013). The elements of a ConOps provide a structure to which the stakeholder requirements can be attached. It also provides an anchoring point for system requirements. Thereafter the model grows with subsystems, software, hardware etc. with attached requirements. When more details are added requirements are allocated, in a more technical form, to lower-level system elements. In this allocation and flow-down process, requirements traceability has to be taken into account.

Defining requirements begins with eliciting stakeholder intentions (referred to as needs, goals, or objectives). Initial stakeholder intentions do not serve as stakeholder requirements, since they often lack clear definition, consistency and feasibility (ISO/IEC/IEEE 29148 2011). With requirements analysis intentions evolve into a more clear statement before arriving as valid stakeholder requirements. Requirements modelling, requirements validation and verification and requirements management are continuous activities within RE. Requirements need to be documented to enable communication with stakeholders and for future maintenance of the system. Describing the relations of the requirements to each other is also a part of requirements documentation. Requirements validation tries to confirm that the requirements are correct, i.e. that they correspond to real user needs or to some other point of reference known as “good”. Requirements verification confirms that requirements are complete and consistent and that they have been correctly derived from higher-level requirements, standards, regulations and other source information. Requirements management, especially checking traceability, should be performed in parallel with requirements definition, not as a separate task afterwards.

3. On the principles of systems modelling

The purpose of this chapter is to introduce, in an informal way, our basic understanding of industrial plants and their I&C systems. The first two sections below discuss the essence of a system and its model. Next, the structural and functional aspects of systems are considered in Sections 3.3 and 3.4. A more structured modelling approach will be given in Part II.

As emphasised by ISO/IEC/IEEE 42010 (2011), system architecture can be seen from several viewpoints. In other words, all the information concerning a system, i.e. the *system model*, should be well organised. In our case, a system model includes various “abstraction levels” like

- Objectives: the big picture covering overall goals and constraints
- Behaviour: operational scenarios and functions of the system, including information items (and other work pieces) processed by the system
- Structure: composition of physical elements of the system
- Layout: placement of the system elements in the operational environment

Secondly, we should consider not only the system itself but also its life-cycle and environment. In requirements definition and conceptual design, external system interfaces and internal architecture are the most important level of description. This leads to the following basic “viewpoints”

- System
 - System in context: how the system interacts with external entities in its environment
 - Architecture: what are main elements and functions of the system and interactions between them
- Life-cycle
 - Engineering: how is the system going to be developed (the project)
 - O&M: how is the system going to be operated and maintained (part of the design solution)

These “viewpoints” and “abstraction levels” are independent dimensions that define a “system description matrix” as shown in **Table 2** below. The message here is that project organisation can also be seen as a system and that we need a pair of interconnected models: the system model and a project model. Moreover, the ways how the system will be used and maintained is part of the design solution, not just something that can be added afterwards.

Table 2. Overall organisation of system and life-cycle models.

	System		Life-cycle	
	System in context (interfaces)	Architecture (internal)	Engineering process	Operations and maintenance
Objectives	Overall goals System purpose and main functions Elements of the enclosing system Geographical locations	System-level goals and requirements Overview of system behaviour, structure and layout	Overall goals and constraints of the development project	Overall goals and constraints of system operation
Behaviour	Data definitions Data exchange, interactions with external entities	Data definitions Functional architecture and individual functions	Engineering activities and tasks Communication and collaboration	Activities and services
Structure (implementation)	Protocols HW and SW interfaces	HW and SW architecture	Project organisation Tools & methods	Operating and maintenance organisation Tools & methods
Layout	Locations	Locations, structures	Locations	Locations

3.1 What is a system?

In common terms, *system* is a set of interacting elements forming an integrated whole. A system, as seen by a human observer, has a boundary, and the system usually interacts with its environment. A system can be abstract (e.g. in mathematics), or concrete, observable in the real world (e.g. natural, social and technical systems). We are here most interested in man-made, intentionally constructed systems having typically both technical and human elements. In other words, we are speaking about *socio-technical systems*. An important consequence of being man-made is that a system has a purpose and an interface to the external world. The system is designed to accomplish certain tasks and to achieve certain goals in a specified operating *context*. This system context usually includes relevant physical, social and cultural aspects of the human organisation, other technical systems and the environment. The context is also affected by the characteristics of the spatial *operational environment* provided by areas, buildings and structures where system elements are located. They must all be considered in the design of the new system.

We can use here the definition given in ISO/IEC/IEEE 15288 (2015) and say that a *system* is a combination of interacting *system elements* organised to achieve one or more stated purposes. System elements can be devices, structures, software, data, humans, procedures (e.g. operator instructions), materials, and naturally occurring entities (e.g. water, organisms, minerals), or any combination. Different from ISO/IEC/IEEE 15288 we exclude “processes” from the list of the system element types. This is because we understand systems as physical (though not

necessarily materialised) entities that exist in the real world¹. Events and happenings occur and can be observed in the real world but are just behaviours generated by the physical system elements. In particular, humans can be considered both as users and as elements of a system. **Figure 5** shows the system being under development, the “system-of-interest”, within its wider *system context*. It consists of system elements, processes work items and interacts with various external entities.

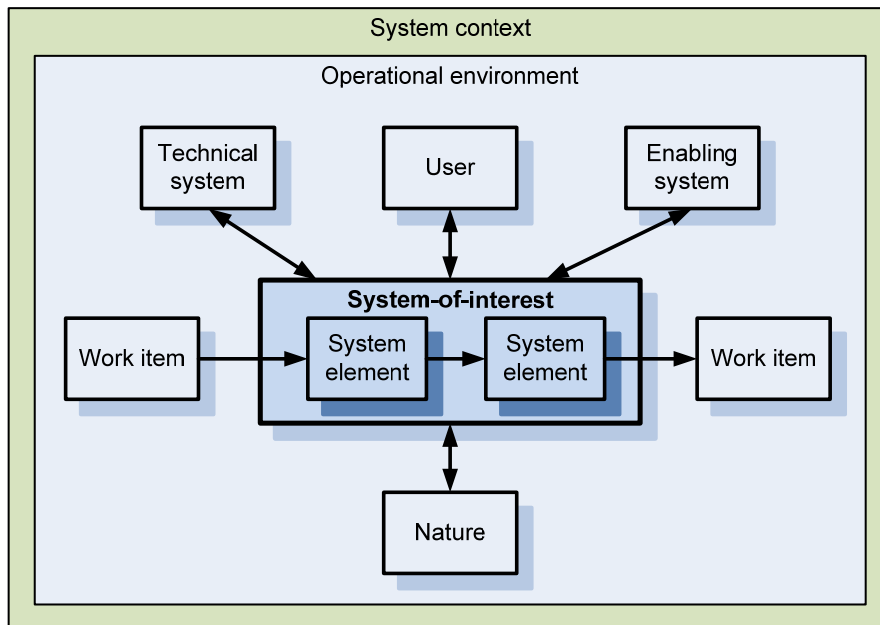


Figure 5. Examples of basic entities related to a system-of-interest being designed. A system consists of system elements and interacts with external elements of a larger enclosing system, all located in an operational environment. The wider system context covers also the business environment, cultural issues, etc.

A clear boundary between the environment and the *system-of-interest* considered by the designer in the system model is important, both in principle and in terms of responsibilities of the design organisation. However, the boundary is a matter of agreement and depends on the viewpoint of the observer. One person's system-of-interest can be viewed as a system element in another person's system-of-interest (ISO/IEC/IEEE 15288 2015). So, different systems-of-interest can in principle overlap. In many areas, e.g. in nuclear power, a clean tree-like decomposition hierarchy is, however, preferred. As an example, a system hierarchy adapted from ISO/IEC/IEEE 15288 is illustrated with an informal diagram in **Figure 6**. The system decomposition is modelled by two system element types: the composite element *System* (system-of-

¹ In formal ontologies this kind of entities are often called *endurants* or *continuants*. They can be perceived to exist as complete entities at any point of time. Examples include material objects but also abstract objects, such as an organisation or a piece of computer software. The “happenings” generated by endurants are known as *perdurants* or *occurrents*, e.g. accidents having a non-zero duration. At a given point in time, we can only see a part of a perdurant.

interest itself being its special case) and the atomic element *component* that is not decomposed further but treated as a black box with a specified interface. Note that, in addition to political and cultural issues, the system context includes external systems that can also be described in terms of (external) system elements.

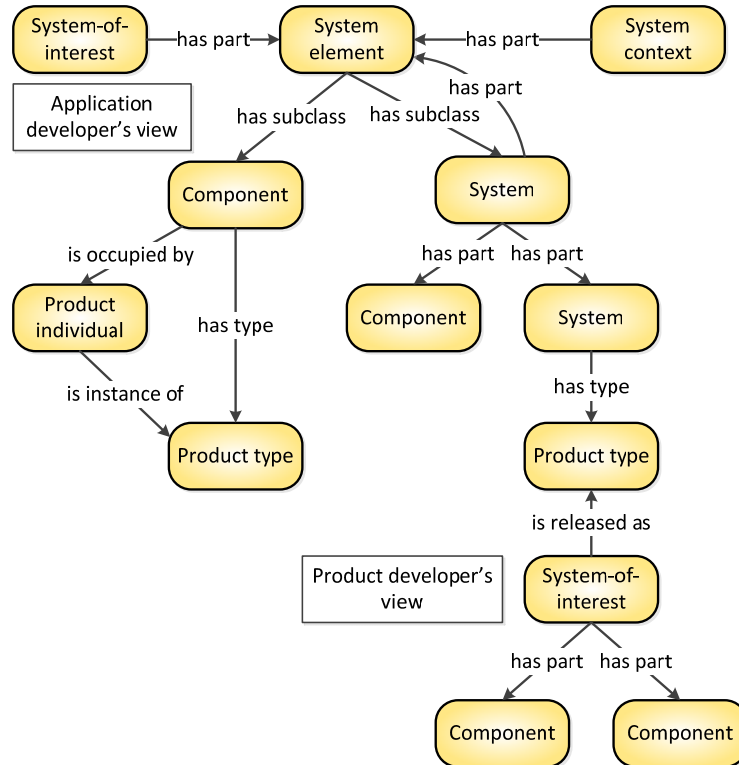


Figure 6. Hierarchical decomposition of a system-of-interest into subsystems and atomic components that can be implemented by using suitable types of products.

Reusable subsystems and commercial components can be published as products. In industrial applications, the primary topic of this publication, components and even larger parts of a system are often acquired as commercial off-the-shelf (COTS) products. For example, standard reactor designs in the nuclear industry can be seen as products. Also the development of entire subsystems is often allocated to a subcontractor in investment projects. As seen from the viewpoint of its developer, the system element is a system-of-interest (**Figure 6**). Its specifications, or relevant parts of the data, can then be integrated to the overall design repository of the main contractor, for example as a document, a *product type* in a type library or as a section of the design database. System element is a place holder referring to a product type and occupied by a *product individual* when the system is actually built and installed in its operational environment. We focus here on design data but tracking individuals is also important, for example, in maintenance and configuration management during system operation. Note also that subsystems that are developed in-house (at the system-of-interest owner) can also be thought of as product types. This facilitates consistent model integration regardless of the supplier of the subsystem and reuse of the design, for example in case of redundant system elements.

By definition, system elements interact with each other and with external entities. This takes typically place by exchanging material, energy, force or information. Some flows are continuous, such as hot water flowing in a district heating network. Discrete flows consist of *work items*, e.g.

of data packages in a computer network, moved between system elements. Anyway, a mechanism is needed to make the interaction happen. For example, in a process system a pipeline is needed to move liquids and gases between process components. In control systems, copper wires carry basic measurement signals, and network components transfer messages between controllers. The most common way to describe system connectivity is to define ports or terminals for system elements and to “wire” them together with connections. However, also other interaction patterns exist. Especially in computer systems, message broadcasting and services with request and reply messages are widely used. An interaction can also be indirect, even unintended, and mediated by a third entity, for example a shared memory location used by two software components. So, interactions are related to system behaviour but made possible by physical system elements. Therefore, interfaces and connectivity need to be considered on both levels of abstraction as illustrated in **Figure 7**.

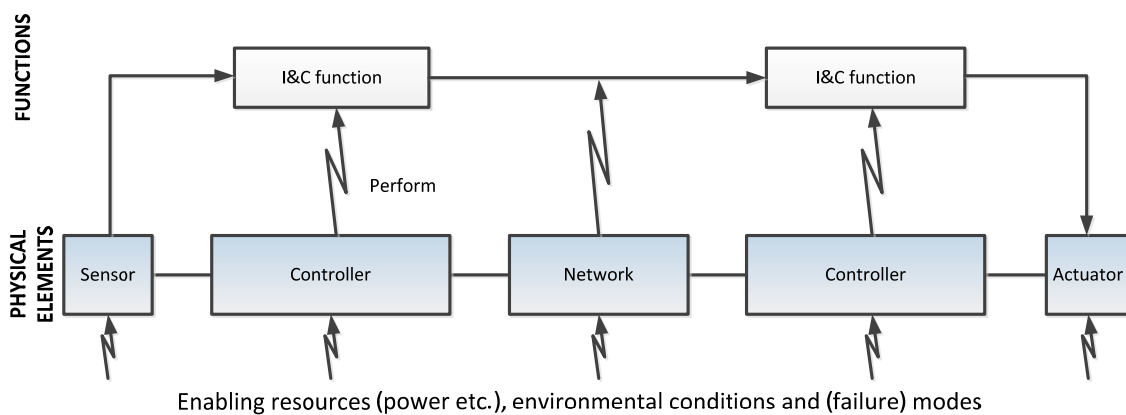


Figure 7. Connectivity is relevant both for control system functions and the physical equipment performing them.

3.2 What is a system model?

The main job of engineers is to provide the drawings, documents and instructions according to which a system can be purchased, built, installed, operated and maintained. Engineers need to develop a “model” of the system. For example, ISO/IEC/IEEE 42010 (2011) defines a model as follows: “M is a model of S if M can be used to answer questions about S.” In principle, a model is anything that can describe a system, for example textual descriptions, graphics, mathematical formula, simulation models, mock-ups, visual models or virtual prototypes. In this sense, all kinds of typical engineering work products (artefacts) that are created to specify or describe a system are models.

Traditionally, design data is originally created as drawings and written documents. Today, design information should preferably be in a well-structured, electronic form that allows it to be developed, stored, processed and transferred with computers. In general, design is more and more driven by models. For example, in Model-Driven Software Development (MDSD), only models are created and maintained, and the implementation, e.g. application software, is generated automatically from the models (see Stahl & Völter 2005). As defined by INCOSE, “Model-Based Systems Engineering (MBSE) is the formalised application of modelling to support system requirements, design, analysis, verification and validation beginning in the conceptual design phase and continuing throughout development and later life cycle phases” (Friedental et al. 2007). This does not mean that documents are not used, but it means that MBSE is not based

on documents. As a consequence, the role of documents changes: In MBSE, documents are a means to present information instead of being containers of information. This fact encourages using automatic or semi-automatic document generation, which is not possible in traditional SE. The requirements, physical architecture and behaviour of the system elements and its environment, as well as the relations between them can be modelled (see **Figure 8**). INCOSE lists several benefits provided by MBSE (Friedental et al. 2007):

- Clear and unambiguous representation of the concepts
- Improved communications among stakeholders and ability to manage complexity by enabling a system model to be viewed from multiple perspectives
- Enhanced knowledge capture, learning and reuse of the information
- Precise system model can be evaluated for consistency, correctness, and completeness
- Automatic or semi-automatic document generation

These are vital aspects also in NPP systems engineering. MBSE is not unfamiliar to current NPP developments. For example, the systems modelling language SysML has been one of the interest areas in nuclear sector (e.g. Pihlanko 2013). However, practices and tools to work with more formal models would still be needed to exploit the full benefits of MBSE.

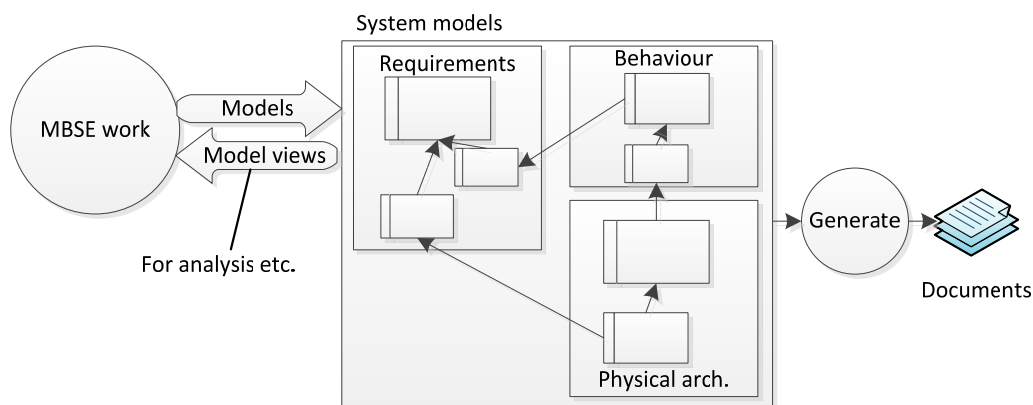


Figure 8. In Model-Based Systems Engineering designers’ work products are organised in a systematic way.

Traditional document-based design and full-fledged Model-Based Systems Engineering (MBSE) can be seen as two extremes of the scale. In this publication, we prefer the intermediate “data-base-oriented” approach where the design repository can include both traditional documents and a structured *model* stored, for example, in a relational database or XML files. Here we expect that the model part is based on a sound conceptual model of the domain so that it can be processed by computer tools, for example for making various checks and generating human-readable documents.

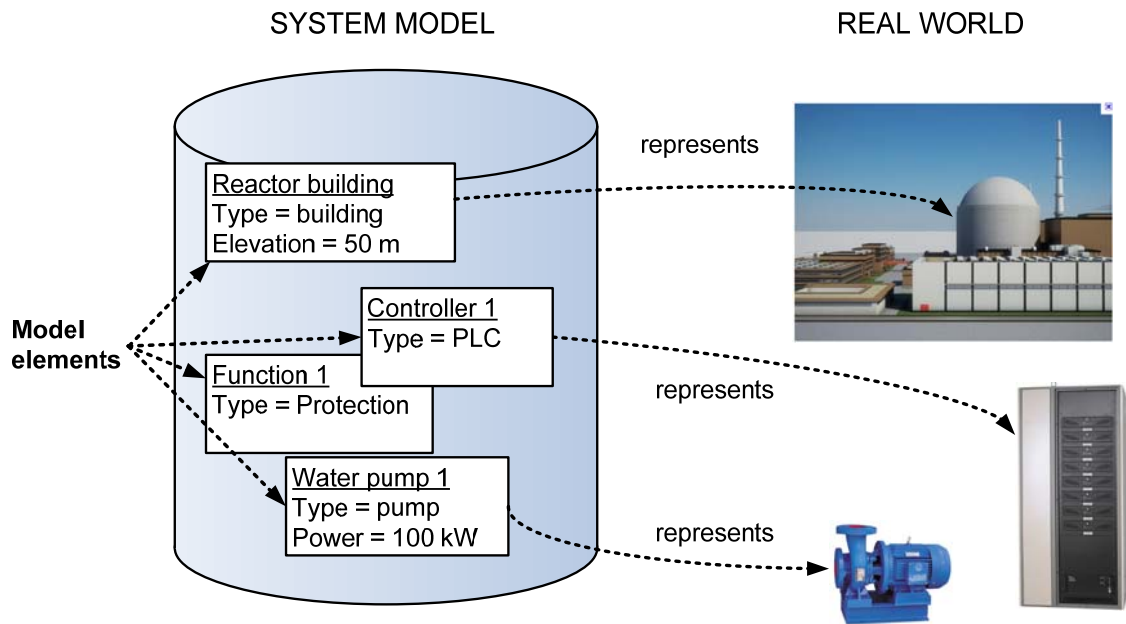


Figure 9. Elements of the system model describe relevant aspects of real-world entities.

A distinction should be made between the elements of the *model* and elements of the real *system* itself. In the following we concentrate on *model elements*, that is, to the pieces of design information that can be used to describe the real-world entities (**Figure 9**). In general, we say that a system model consist of model elements that represent the relevant aspects of the real system, see **Figure 10**. Since traditional documents are needed to complement structured system models we use also the term *engineering artefact* to cover all the information, i.e. documents and model elements, produced by the design organisation.

To cover all necessary aspects of the system, we need several types of model elements to describe the goals, structure, properties and behaviour of the system. Some model elements, such as functions, are abstractions that don't have direct counterparts in the real world. In addition to technical content, model elements include information related to the system life-cycle (metadata), for example versions, modification dates, authors and acceptance status. In particular, a model element should have an identifier that allows a reference to be made to it. The granularity of model elements varies. Smallest model elements may include, for example, just one property value or a requirement, while some others contain lots of information, possibly in the form of many lower-level model elements. In order to make the management of design information easier, small model elements are grouped into *configuration items* that are subject to the actions of *configuration management*. Moreover *information packages* (term adopted from NIST 2006) are collections of model elements transformed into an appropriate format for the exchange of (electronic) design data among various project participants. Traditional documents can be understood as information packages that have been transformed into a human-readable form.

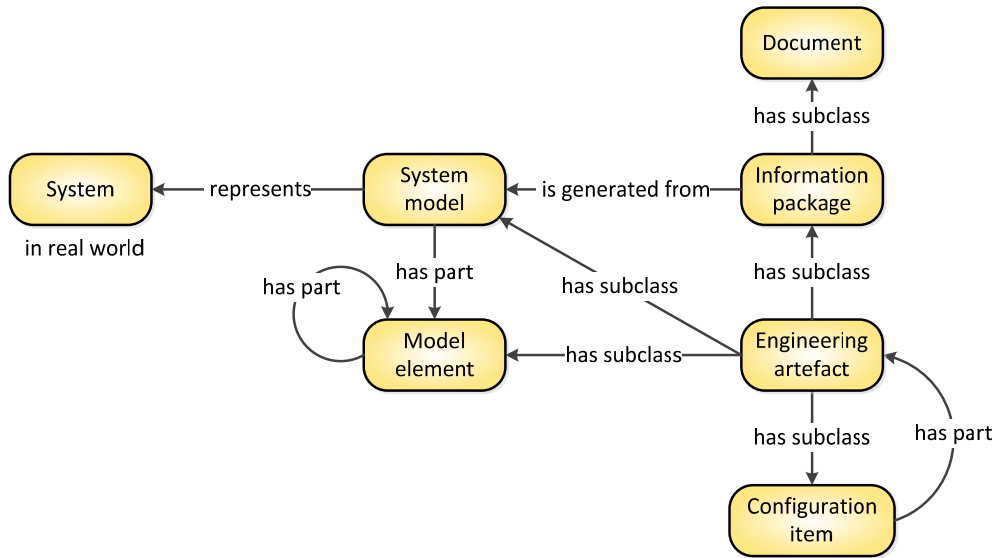


Figure 10. Elements of the system model represent the system in existing in the real world.

3.3 Physical structure of a system

The structure of a system can be understood as the arrangement of its elements. To describe it engineers should specify:

- elements of the system
- how the elements are interconnected
- where the elements are located.

These issues as discussed below. A practical description of system structure should also include, for example, the purpose and form of each element.

3.3.1 System elements

Our *systems* are not abstract but consist of concrete (physical) elements that exist or will exist in the real world. They can be materialised objects like people and computer hardware or information, such as configuration data or a piece of software running on a computer. We see systems as resources having capabilities that can be used to perform activities. Other noteworthy characteristics of these real-world systems include:

- A system comprises a set of subordinate system elements. Very often the system-of-interest itself is part of a larger enclosing system. There are hierarchical and other relationships between system elements.
- The perception and definition of a particular system, its boundaries, architecture and system elements depend on an observer's interests and responsibilities (ISO/IEC TR 24748-1 2010). This has several consequences:
 - there are more than one way to decompose a complex system
 - a system element can, in the general case, be part of more than one systems
 - an entity at any level of decomposition can be viewed as a system

- Elements of a system may be changed over time without the system losing its identity. Therefore, the elements of a system can be understood as place holders for actual component individuals that, in many cases, are instantiations of a commercial product or device type and have a manufacturer's serial number. For example, Figure 11 shows a functional pump object P101. It is distinct from the individual “pump 1” that was first installed as P101 and later replaced by a spare part item “pump 2”. Including the time dimension seems useful not only for design and system modelling but also for configuration management and traceability during system operation.
- Throughout the life cycle of a system-of-interest, services are required from external systems, e.g. a training or maintenance system. Termed *enabling systems*, they facilitate progression of the system-of-interest through its life-cycle (ISO/IEC/IEEE 15288 2015).
- Human organisations can even be conceived as systems of their own. In today's networked manufacturing, these organisations can exceed the boundaries of a company (Oedewald 2011) and a process plant. In addition, the project organisation, or usually a network of organisations, conducting the investment project is a system, as well.
- A further question is, whether the work items, e.g. data or material, processed by the system are part of the system itself. Just like system boundaries depending on the viewpoint taken, the answer could be yes or no. In general however, we think that work items flow through a system without being its components. In spite of this, it is necessary to describe work items and their properties in the engineering process.

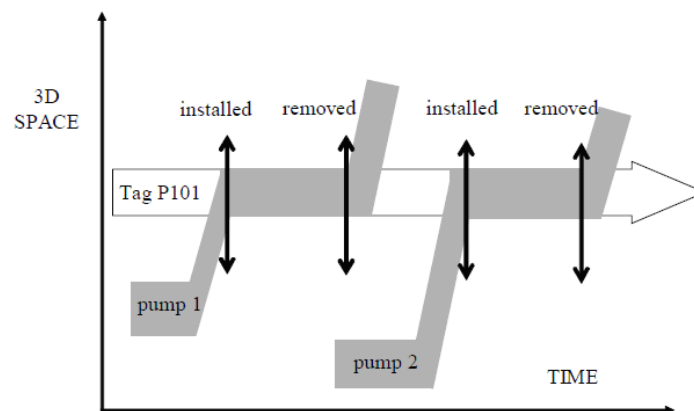


Figure 11. The functional pump object P101 in the three dimensional space can be occupied by several product individuals during the life-time of a process plant (ISO 15926-2 2003, West 2011).

Figure 12 shows a simple diagram to illustrate the key ideas. Firstly, a system is intended to carry out one or more activities, either alone or together with other systems. This is the *purpose* of the system as defined by the designer. Secondly, a system consists of system elements that can be of two types. Either a system element is a composite object, i.e. a “subsystem” of its own right. Or, a system element is a component, the internal structure of which is not interesting from the observer's viewpoint. So, components represent the leaf nodes of the decomposition tree. Components can be classified into a few subtypes like people, devices, software and information. Devices, e.g. pumps and instruments, are typically mechanical parts but may include

embedded software, as well. Structures are often defined as passive elements. However, they can also perform a function and today even include active parts and software. So, the classification is not exclusive. As said above, system boundaries depend on the viewpoint. The kind of components included in a system provides a basis for the classification of systems as a whole. *Organisational units* consist primarily of people, *process systems* of process devices, *structures* of structural elements, and so on. *Socio-technical systems* combine all kinds of components.

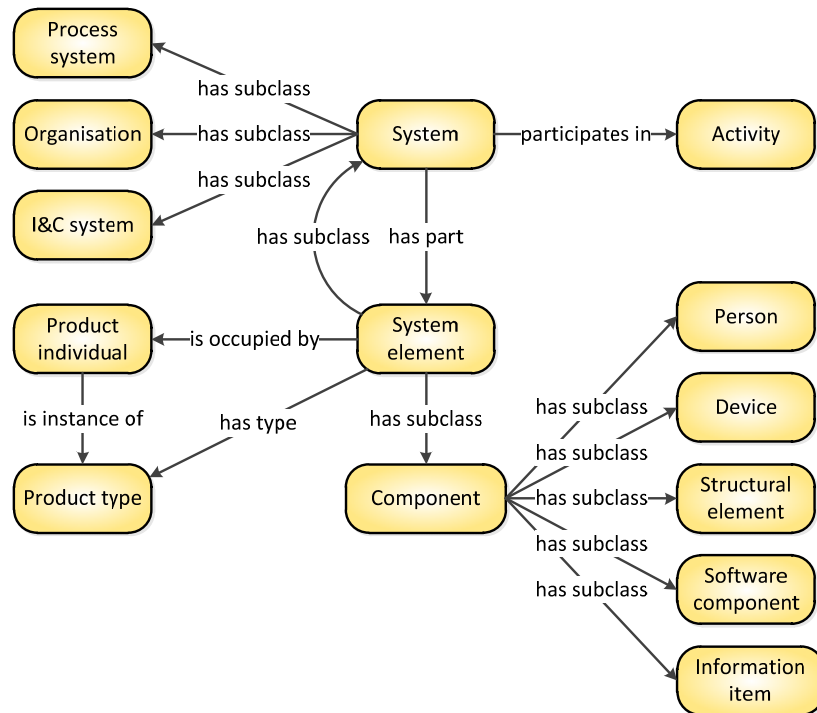


Figure 12. System purpose and decomposition with examples of system and component types.

3.3.2 Connectivity of systems and system elements

To perform their intended functions, physical system elements must interact by exchanging material, energy and information. A common approach, used for example in SysML (OMG 2010), to model these interactions is to define *ports* or terminals that accept and produce various types of work items. The main motivation for specifying ports on system elements is to allow the design of reusable blocks with clearly defined interfaces.

In this publication, we make a clear distinction between the physical structure and the behaviour of a system. The physical *system elements* produce the intended behaviours according to the rules defined by *system functions*. This also means that a distinction must be made between the connections between system elements and the connections between system functions. The physical components of the system implement the flows between system functions. Consequently, components making up a physical connection can be understood also as a system element. Similarly, a flow linking two system functions can be described as a function of its own in a more detailed view. So, in the model of the physical system structure system elements have ports like piping nozzles and screw terminals that are linked by joint components, such as pipes and cables. Notice, however, that system elements can interact, e.g. through heat transfer pro-

cesses, without having physical ports and connections designed for that purpose. The interactions are then considered in the functional model. Connectivity of system functions is discussed further in Section 3.4.3.

3.3.3 Spaces and locations

Systems and people need a suitable *operational environment*, i.e. geographical regions, site areas and buildings, to work efficiently. The properties of the operational environments are part of the input information, constraints, to system design. On the other hand, the decisions made in system design can put demands on the operational environment. Hence, it makes sense to take some ideas from the construction industry. In particular, the building sector’s Industry Foundation Classes (IFC) represent an open specification for Building Information Modeling (BIM) data that is exchanged and shared among the various participants in a building construction or facility management project (IFC 2011).

Figure 13 illustrates the main generic concepts. Following the terminology of IFC, *building elements* are the primary parts of a building. They are physically existent and tangible things, such as walls, beams or doors. In a nuclear power plant, the containment is a good example. A *space* represents an area (2D) or volume (3D) intended to provide certain functions. A space can be decomposed into parts. It can have various attributes like required design temperature, humidity, ventilation, and air conditioning. Spaces are typically bounded by structural building elements like walls. A space boundary can also be virtually defined with respect to structural elements. For example, explosive atmospheres or possibility of nuclear radiation may require a classified space to be declared around hazardous components. Systems and system elements can be placed in a space wherein they are therefore “contained”. The *location* of a system element is understood as the relationship between the system element and the space. It is in other words a (possibly dynamic) property of the system element. In addition to building elements and spaces, industrial systems require various support structures, such as cable trays and steel frames for process equipment, to which they can be attached. Sometimes, for example in case of computer cabinet, these NPP *structures* can also be considered as components of NPP systems.

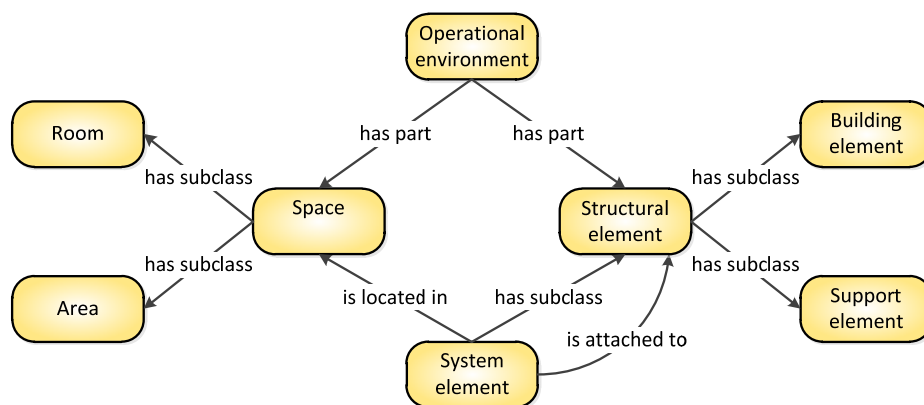


Figure 13. Elements of the operational environment.

3.4 System behaviour

We stated above that *systems* are designed for a purpose, usually to do something. Therefore, behaviour is an important aspect of a system in addition to its physical structure. To describe what happens we need (at least) the following concepts:

- Activities: What needs to be done, e.g. production of electricity
- System functions: What a system can do, e.g. measuring reactor pressure
- Operational states: Foreseen states of the system, e.g. cold shut-down
- Operating modes: Optional ways performing system functions, e.g. manual or automatic
- Events: Momentary changes in state
- Scenarios: Examples of sequences of states, events and actions

3.4.1 Activities

In general, an activity is something happening or changing. In our case, activities are used to describe the intended behaviour of goal-oriented agents, such as human organisations or technical systems. Activity refers to something that is desired and should be accomplished in some way. So, activity is closely related to the term “process”. For example, a block diagram (ISO 10628 1997) developed early in process design describes what a process plant is supposed to do with the raw materials. At this stage, the details of the process equipment may not yet be known.

We can apply the old function modelling standard IDEF0 (1993) to describe activities. An activity is modelled as a box as shown in **Figure 14**. Arrows entering the left side are inputs that are consumed by the activity to produce outputs on the right side. Arrows on the top are controls that the conditions, such as an enabling signal, required for the activity to produce correct outputs. Mechanism arrows connected to the bottom represent the means, i.e. resources and techniques, for the execution of the activity.

Each activity can be decomposed in a hierarchical fashion to lower-level activities and flows of material, energy and information between them. Well-defined activities with a limited scope and duration can be called *tasks* that are often associated with human users but can also be performed by technical systems. Tasks can in turn consist of atomic *actions*, e.g. turning on a light switch. Actions can be considered as *events* (see Section 3.4.5). An activity, e.g. periodic test of an instrument, can be performed several times. Therefore, it makes sense to distinguish activity types (classes, e.g. “starting up the plant”) and activity instances, that are executions with specific start and end times (e.g. “start-up on the 11th of November”).

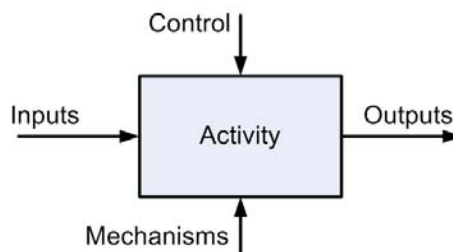


Figure 14. IDEF0 activity box.

The inputs and outputs of an activity are usually material, energy or information. More generally, activities produce, maintain or sometimes prevent certain states-of-affairs, e.g. “fuel loaded into

reactor”. The control arrow can, for example, be understood as a goal or activation command. The interesting feature of the mechanisms is that it links the activity to the resources, e.g. process equipment or people that are used to perform it. We say that systems have capabilities that, depending on their structure, behaviour and properties, can be used for various purposes.

3.4.2 System functions

The term “function” has many meanings in various disciplines. In engineering, function is an essential concept, both for designing systems to have desired properties and for diagnosing why a system is not functioning as expected. However, the term has been a source of endless discussion and attempts at definition (Chandrasekaran & Josephson 2000). In spite of that, we try here to find an interpretation useful in the context of industrial automation.

In engineering design, it is useful to apply this term to the general input/output relationship of a system whose purpose is to perform a task (Pahl & Beitz 1996, p. 31). Function is a property of a technical system and describes its ability to fulfil a purpose (Hubka & Eder 1988, p. 72). Quite obviously, a system can serve several purposes, and therefore also have several functions.

Engineered systems, such as cars, houses and computers, are made to perform a function, and it is that function that defines them. A screwdriver remains a screwdriver even if newer used to drive screws but only to open tins of paint (West 2011, p. 152). So, the intentions of a user in a given situation may be different from the usages that the designer had in her/his mind. IEC 81346-1 (2007, p. 14) states that the components of the technical system are the static prerequisite for the dynamic activities of the process. A “function” signifies the task of an object, e.g. an “object intended for heating a liquid”, without taking into account its implementation.

So, the concept of function starts with the concept of behaviour, but (different from natural systems) it is distinguished by the fact that some agent regards it as desirable (Chandrasekaran & Josephson 2000). Some sources associate functions with the users’ goals, some others seem to think the systems and functions are the same thing. As stated by Chandrasekaran & Josephson (2000), functions can be described from a device-centric or an environment-centric viewpoint. The former describes the desired behaviour in terms of the system while the latter emphasises the intended effects on external entities.

Note: The word “purpose” can also be given two interpretations, one from the user’s and another from the designer’s perspective. For example, the user might say that for him the purpose of the screwdriver is to open a tin of paint. The designer in turn would probably say that its purpose is to drive screws.

In order to provide a sound basis for system modelling we need to give the term function a more specific meaning. To do this we define a *system function* as a capability of a system to do something useful. With this definition, the *purpose* of a system refers to the activities or goals it is designed for by the developer or used for by the end-user. A function refers here to a *model element* describing the capability of a system and possibly its internal logic (e.g. a rule or an algorithm) (**Figure 15**). Depending on the situation, a system function, such as a temperature control loop or a start-up sequence, is often capable of generating many required behaviours of the system. In the system model, a system function can be represented as a requirement (“shall have temperature control loop”) or a specification (function block diagram) that is later implemented as a piece of application software (software component) in the control system.

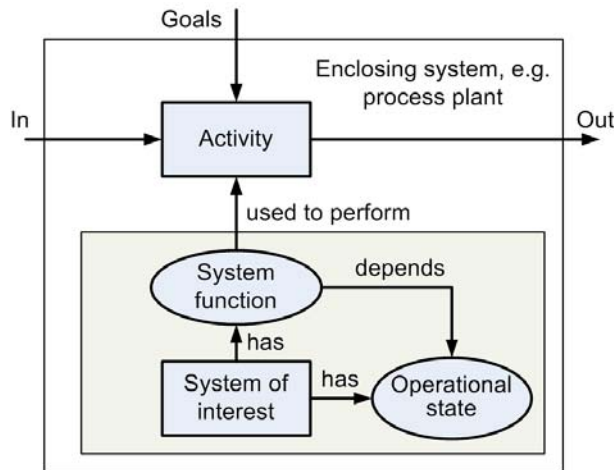


Figure 15. Systems have system functions that can be used to carry out activities.

In many cases, systems are designed to actively intervene with the external world by exchanging material, energy or information with it. This is, however, not the whole truth. For example, Lind (2005) lists on the basis of the work by Georg Henrik von Wright a limited set of primitive (active) interventions that a system can do. Firstly, system can produce a new state-of-affairs by changing the state of the world. Secondly, it can maintain the current state-of-affairs. By considering the negations of the state-of-affairs we have two more interventions, to destroy and to prevent a state-of-affairs. Furthermore, taking into account the negations of active interventions, i.e. omissions, leads to four additional ways of letting things happen. Not reacting to an alarm can be considered as an act of the control room operator. In particular, maintaining and preventing states-of-affairs match perfectly with what various *structures* in an industrial plant are expected to do. For example, beams and columns of a building are designed to support other building elements in a fixed position. This leads us to the conclusion that passive structures can have functions. To take an example from the nuclear domain, the containment in a nuclear power plant should be able to prevent the release of (any kind of) materials to the environment. “Keeping materials inside” is a *function* of the containment. According to what was said before, the *purpose* of the confinement is to contribute to the safety function called “confinement of radioactive material”.

A system function of a large system can be decomposed into “subfunctions” that are then allocated to its subsystems (**Figure 16**). There are several options. Firstly, system-level function can be allocated to exactly one subsystem. In this case, the model element “function” could be associated both with the system and the subsystem. Or two model elements might be used, for example the system-level function interpreted as a requirement and the more detailed one as a specification. Secondly, a system-level function can be allocated to several subsystems. In this case, the subsystems must interact in some way. For example, subsystems performing different parts for the function must exchange information. Or, the action performed by one system element can make use of a function provided by another system element, for example by calling a subroutine or activating a function. Also redundant subsystems performing the same function must be coordinated by a switching or voting function. This means that functions at the system level and functions at the subsystem level are usually not the same thing and must be modelled with separate model elements.

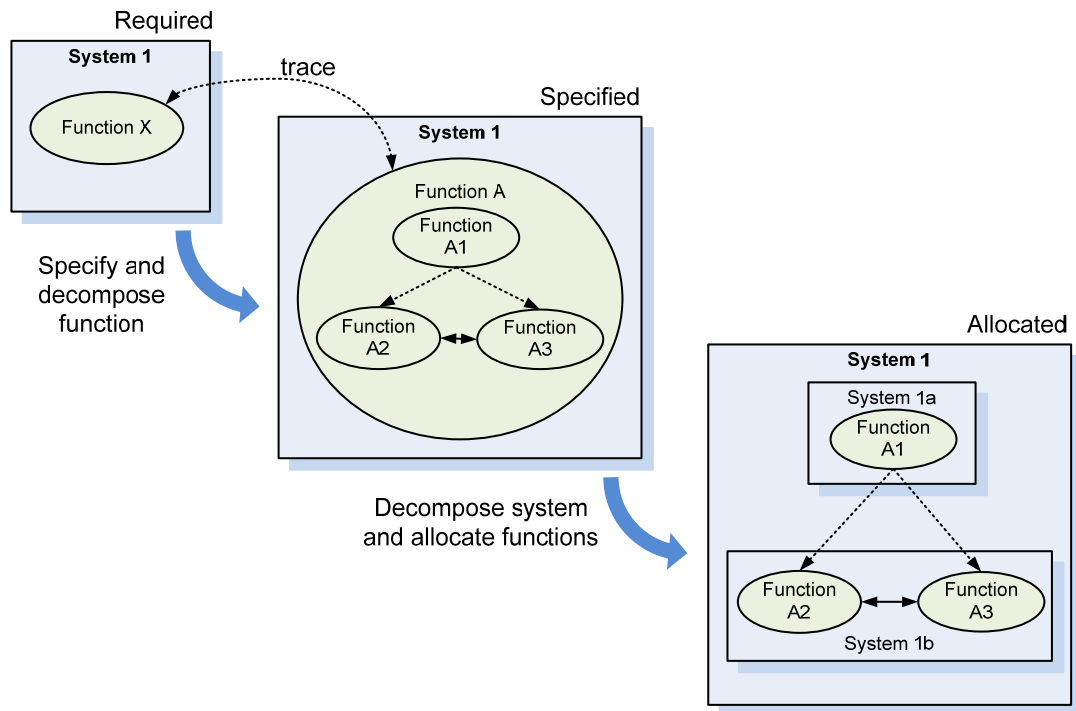


Figure 16. System-level functions are accomplished as various combinations of the functions allocated to subsystems and their components.

As was indicated in **Figure 15**, activities often represent requirements associated with a larger, enclosing system. For example, activities tell that the power plant must be designed to generate electric power. System functions, on the other hand, often describe the “services” that a power plant system, e.g. an I&C system, provide to its “customers”. To do this, lower-level internal subfunctions need to be defined.

So, functions are here understood as features of systems while activities are not necessarily linked to any resources. It may not be known at design time what specific systems will be used to perform an activity. However, the characteristics and requirements of an activity determine the kind of resources that can be used. Similarly, the designer of the system and its functions may not know exactly the activities the system will be used for. This situation is typical, for example, in batch chemical processing, where the recipe corresponds to the production activities that are carried out by the procedural control functions associated with the process units. It is, however, not always clear whether functionality should be modelled as an activity or as a function. A safety function of a nuclear power plant, for instance, could be seen as an activity or as a plant-level function. These main safety functions are then achieved with combinations of the safety functions or various plant systems.

As explained above, the capabilities of a system to exhibit certain kinds of behaviours can be defined in terms of *system functions*. In an I&C system this typically means control and measurement loops. In the first place, functions are specified in a way that is independent of the implementation. However, the available technology can have an impact on the semantics of the specification. For example in programmable control systems, functions are often implemented as software components that are executed cyclically. This is very different from the continuous model in older analogue solutions.

This publication focuses on basic process control systems where control loops, interlocks and protective functions are the most important ones. The behaviour of a control application is mod-

elled following the “function block paradigm” widely applied in the development of basic process control systems (**Figure 17**). Function blocks are often understood as elements of an application-oriented programming language (e.g. IEC 61131-3). Therefore, we use here the term *I&C function* (IEC 61513 2011) when referring to an implementation-independent transformation of incoming data flows (signals) to outputs. So, I&C functions represent an application-oriented view to an I&C system. Most I&C functions are thought to be performed cyclically. They represent continuous control functions, such as controlling pressure of a reaction. However, a function may describe sequential activities and state machines, as well.

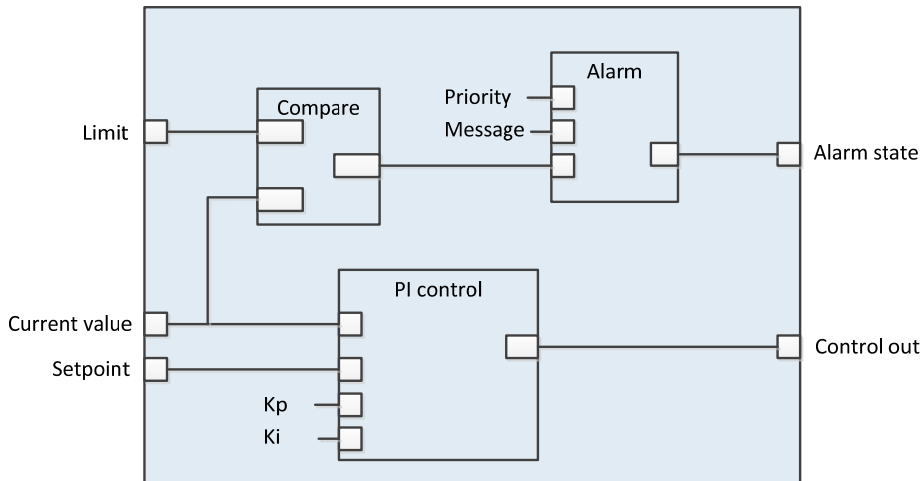


Figure 17. A network of I&C functions as a Function Block Diagram (FBD).

3.4.3 Connectivity of system functions

The designer uses system functions to describe, in a structured way, how systems exchange and process material, energy and information. Each function accepts certain inputs that it transforms them into outputs for use by other functions. Like functions themselves, the flows between functions can be understood as continuous (steam in a pipe) or discrete (work pieces on a transfer line). As said earlier in Section 3.1, a physical resource, e.g. communication network, is often needed to make the transfer take place. So, the flows can also be considered as functions. There are several mechanisms to implement the flows and to connect the functions to each other. Especially in models of computer-based systems, flows are thought to carry discrete *information items* that represent signal values, commands, alarms, service requests, etc. As illustrated in **Figure 18**, the digital technology enables several interaction topologies that are visible already in the platform-independent functional specification. With the focus on basic I&C functions and function block paradigm we limit the discussion here to wiring ports that can consume or produce certain kinds of information items. In the typical case, the semantics is that data is transferred “continuously” over link. This is true for traditional hard-wired solutions, but in digital systems data exchange actually takes place cyclically or triggered by a change in the value. The designer may need to be aware of technical issues like allowed maximum delays, cycle times and reliability of data communication and synchronisation of functions allocated to different controllers. Moreover, also other types of connectivity are found in digital I&C systems.

In particular, alarm handling is an example of distributing event notifications to interested subscribers. In higher-level information systems, interactions can be modelled in terms of request and reply messages, i.e. services provided and required by system functions. In principle, also messaging and services can be modelled with ports of different types.

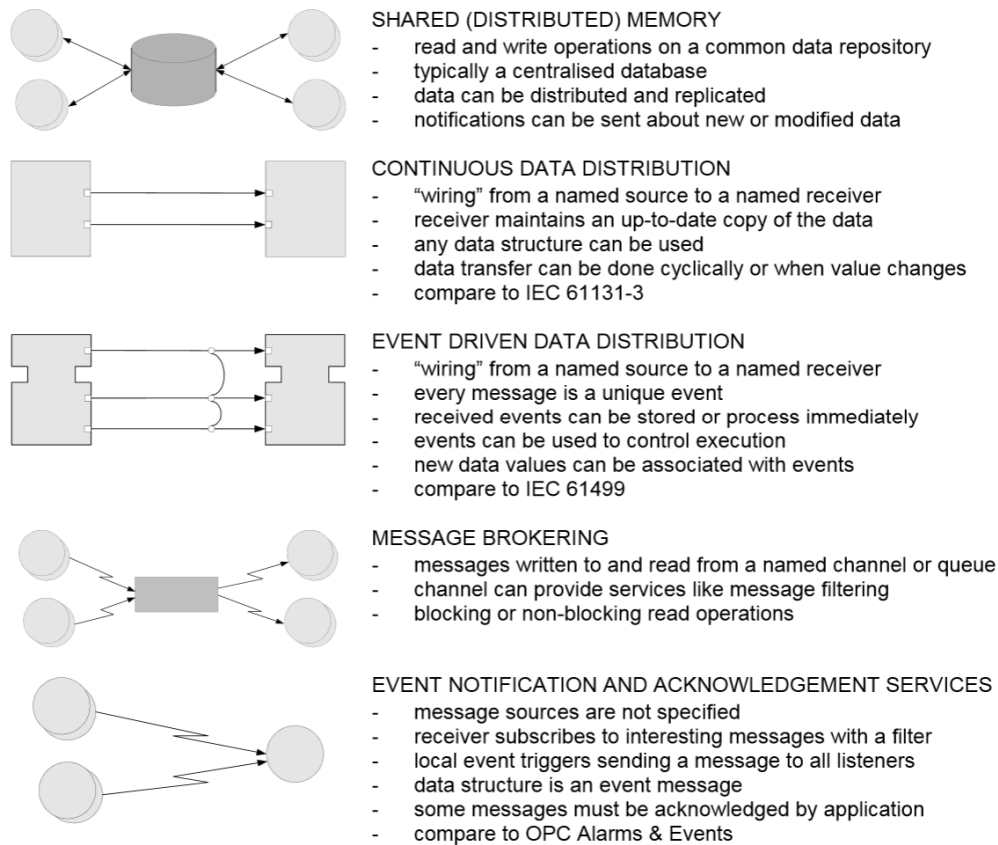


Figure 18. Examples of interaction mechanisms in digital systems (Tommila et al. 2005).

3.4.4 States and modes

A further observation to be made is that systems can be in various states. In system theory, a system has a very large or infinite state space defined by their state variables. This is not very practical for design work. To model relevant alternatives we use the term *operational state* which can be understood as regions in the system's state space. They are seen as agreements made by designers and plant personnel. Often, the operational state of a system is rather declared or forced by the control system than inferred from current and past values of process variables. Some operational states are stationary with the design intention to maintain stable process conditions. The purpose of transient states, e.g. plant shut-down, is to move the system from one operational state to another. Therefore, they often have a sequential character. The availability and performance level of system functions depend on the current operational state. A power plant, for example, can have operational states like “in maintenance”, “starting-up” and “power operation” (**Figure 19**). Operational states are connected by instantaneous *events* and *state transitions*, thus allowing us to describe the behaviour as a state diagram.

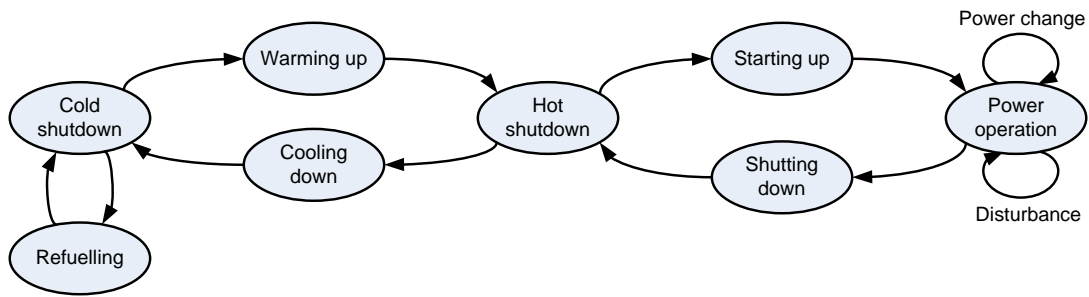


Figure 19. Operational states of a nuclear power plant.

Operational states can be associated with systems on different levels of decomposition, e.g. with whole power plant or with each of its subsystems. Thus, each operational state can be defined in terms of the states of lower-level subsystems. In addition, an operational state may have substates that are associated with the same equipment entity but valid during different periods of time. This results in hierarchical state diagrams. For example, UML statecharts introduce hierarchically nested states.

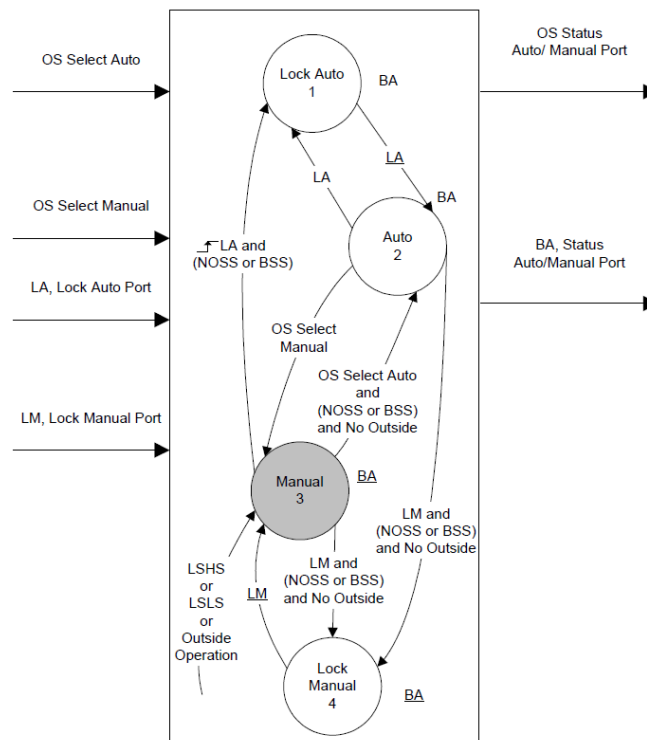


Figure 20. States and state transitions of the auto-manual mode (NORSOK I-005 2005).

Operational states determine what a system is capable or supposed to do in certain situations. The concept of *operating mode* defines how the system performs its functions. In process control, functions can be typically performed automatically or manually. Modes and transitions can be defined as another state machine that controls how the actual functions, e.g. PID function blocks, are performed (**Figure 20**). Operational states and modes are a valuable tool for organ-

ising the behavioural descriptions of a system and to identify risks and requirements related to various situations. To some extent, states and modes are application-dependent, but also more general taxonomies can be defined. For example, the ISA SP88 committee and Organization for Machine Automation and Control (<http://www.omac.org>) have together defined states and modes for automated machines (OMAC 2006, ISA 2013).

3.4.5 Events

Events are understood as instantaneous changes in the state of affairs, or as occurrences that are short enough in duration to be considered as atomic from the modeller's viewpoint. For example, the temperature reaching a specified limit or a push button being pressed are examples of events. This idea is adopted here also even if events are often treated in the industry as episodes of special interest, usually unfortunate ones, such as accidents. For example, IAEA glossary (IAEA 2007) defines event as "any occurrence unintended by the operator, including operating error, equipment failure or other mishap, and deliberate action on the part of others, the consequences or potential consequences of which are not negligible from the point of view of protection or safety". The International Nuclear and Radiological Event Scale (INES) managed by IAEA rates events at seven levels: Levels 1–3 are "incidents" and Levels 4–7 "accidents".

Events are related to *actions* discussed above in Section 3.4.1. Both are atomic with respect to their duration, but actions are performed by active system elements (actors). For example, a description of an operator's task might include the statement "operator starts reactor cooling". Events can also express observations made from the external world and be described in passive voice, e.g. "reactor temperature becomes less than 100 C". So, actions can be seen as subclasses of events. Actions are also intended to have an effect on some external objects. As in the examples above, external objects may then react after a while by producing an event. If foreseen events and actions are defined as model elements they can be used in the description of more complex system behaviours, for example in sequential I&C functions and state machines of operational states and operating modes as in **Figure 20** above.

3.4.6 Scenarios

Scenarios are stories that describe (both desired and unwanted) happenings and situations expected to occur during the operation and maintenance of the system. Like test-cases and simulations, scenarios are examples of possible occurrences encountered during the system's life-time. As such, they don't necessarily cover all situations, for example complex combinations of events. However, scenarios are useful for the synthesis, analysis and communication of needs, requirements and solutions. Identifying possible exceptions and hazards lead to new requirements and help to make systems safer.

Many notations can be used to describe scenarios. The basic form is a short story. More structure can be added to a story with dedicated sections and tabular form. For example in software and systems engineering use cases are a means for specifying units of useful functionality that a system provides to its users (OMG 2010). A use case captures a contract between the stakeholders of a system about its behaviour (Cockburn 2001). A use case is a list of steps, possibly with variations to describe alternative paths, typically defining interactions be-

tween an actor and a system, to achieve a goal. The actor can be a human or an external system. As such, a use case can define a whole family of specific scenarios. Also some normal design artefacts, such as sequential procedures and timing diagrams, can be understood as scenarios as they are examples of potential, usually intended, occurrences. In any case, scenarios should use well-formed natural language and agreed terminology of the domain and particular application. Scenarios illustrate and verify how the various actors of the application, i.e. people, systems and their functions collaborate in various situations in order to perform the activities they are designed for.

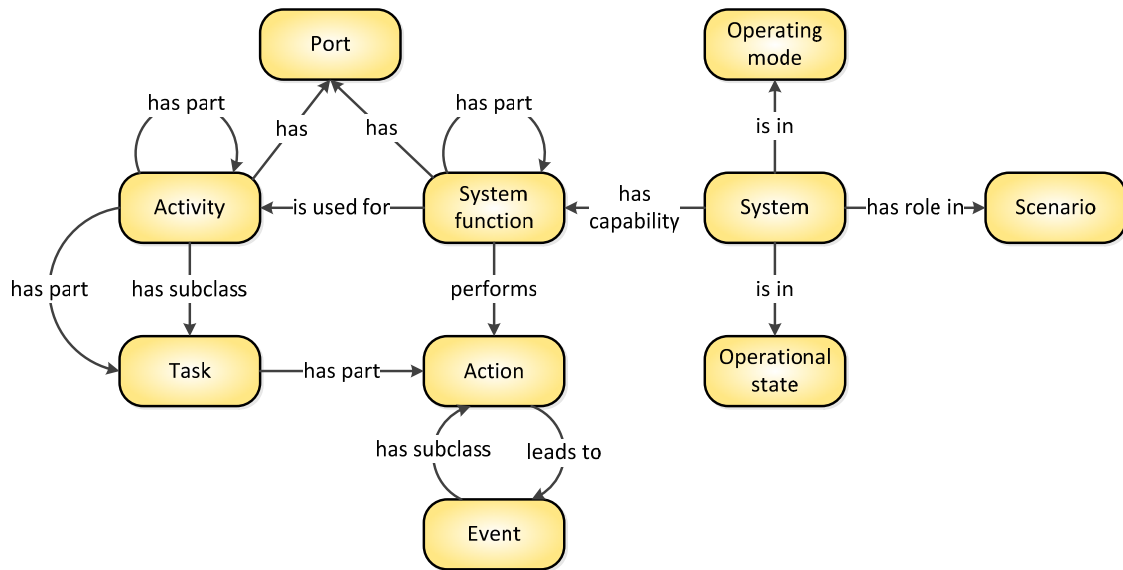


Figure 21. Modelling concepts for describing system behaviour.

To summarise Section 3.4, **Figure 21** illustrates the main model elements that can be used for describing the dynamic behaviour of a system. *Activities* and *functions* basically define the rules according to which a system behaves. Another way to describe system is to give representative examples. This approach, often called “specification by example”, is used, for example, in agile software development that tries to avoid formal and extensive documentation. Examples assist agile teams in ensuring that there is a shared understanding of what a system shall do. Examples can also be used for planning acceptance testing. We use the general term *scenario* for a *model element* that describes a hypothetical but realistic episode in the operation of one or more system elements.

4. Life-cycle modelling

Developing, operating and maintaining a *system* involves many *activities* carried out by various organisations (**Figure 22**). The project organisation can be understood as a kind system with a structure and behaviour event though the terminology may be different. Usually a life-cycle model is understood as the sequence of distinct phases in the development and operation of a system. There are many well-known options ranging from waterfall and spiral models to various agile methods. According to ISO/IEC/IEEE 15288 (2015), the detail in the life-cycle model is expressed in terms of these processes, their outcomes, relationships and sequence. We suggest here an extended interpretation saying that the *life-cycle model* of a system describes

- activities to be carried out
- goals, inputs, outputs (e.g. design artefacts, information items) and flows between activities
- phases, baselines and milestones
- roles of participating organisations
- practices, methods and tools to be applied

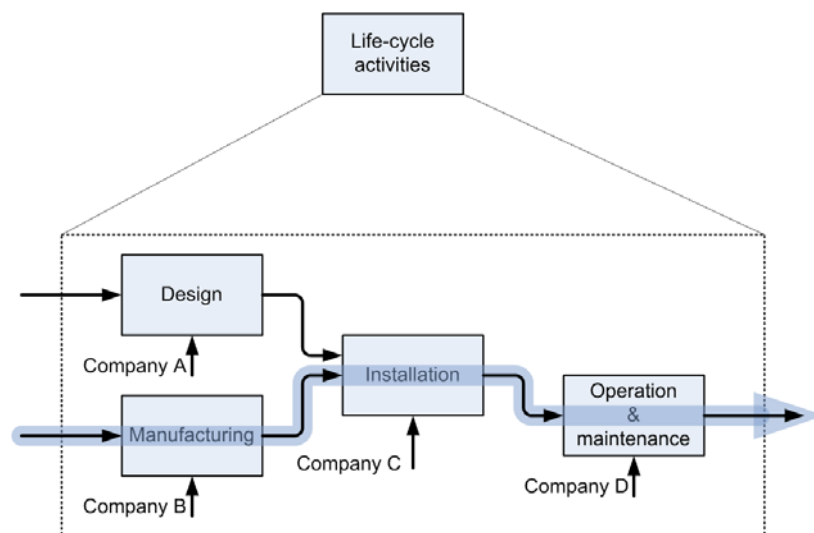


Figure 22. An example of life-cycle activities using IDEF0 notation.

A life-cycle model can be generic within an application domain or a product family, or it can be specific to a development *project*. As illustrated in **Table 3**, the top-level activities around an industrial plant are design, (Engineering, Procurement and Construction, EPC), component manufacturing, installation and Operations & Maintenance (O&M). We are here more interested on design activities and can use the term *project model* as a pair of the system model. When combined with application-specific data the generic reference model of the system life-cycle

becomes a project model. The detailed practices of system operation and maintenance are part of the design solution and must therefore be specified during the design stage in the system model.

Engineering activities can be decomposed into lower-level activities and tasks in many ways. Preferably this should be done on the basis of the goals and expected outcomes. We say that engineering activities create, maintain or manage certain sections of the system model. It may also be reasonable to consider who is going to perform the activities. ISO/IEC/IEEE 15288 (2015) suggests one decomposition for systems and software engineering. **Table 3** shows a simplified version freely adapted to our context.

Table 3. Examples of life-cycle activities (adapted from ISO/IEC/IEEE 15288 2015).

Enabling activities of the organisation	Project-related activities			O&M activities
	Procurement	Management	Engineering	
Human resource management	Acquisition	Project planning	Mission analysis, requirements definition	Operation
Infrastructure management	Supply	Project assessment and control	Architectural design (functions, hardware, software)	Maintenance
Knowledge management		Risk management	Detailed design	Disposal
Quality management		Configuration management	Implementation	
		Information management	Integration	
			O&M planning	
			Commissioning	
			Verification and validation	

The network of activities and tasks is quite large. There is often a need to see relevant information together to provide visibility to an engineering interest or thread that cuts across several processes. For example, the business process of a component manufacturer might be more focused on the flow of its products through the manufacturing, installation and maintenance activities. Also a safety engineer would probably have a special viewpoint to the life-cycle activities. For this purpose, ISO/IEC/IEEE 15288 (annex D) formulates a “process view” that includes guidance and recommendations explaining how the outcomes can be achieved by employing the existing activities and tasks. The “delivery process” illustrated with a thick line in Figure 26 is an example of this idea. So, the concept of process view can give a more specific meaning to the common term “process”.

Note: In our context, the term process is primarily associated with energy production activities, i.e. something happening in a power plant (see ISO 10628 1997). However, in every-day language the word “process” often refers to process equipment, i.e. to process systems. Therefore, we prefer speaking about activities. When the word process is used, a distinc-

tion should be made between engineering processes, business processes, manufacturing processes, etc.

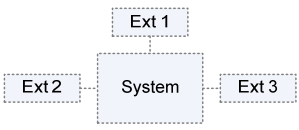
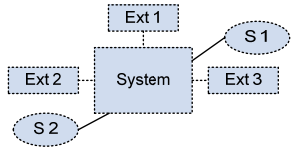
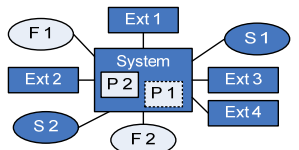
When same activities are repeated on the same level of the system, the design approach is referred to as iterative. When the same activities are applied to successive levels of system elements within the system structure, the approach is referred to as recursive. Since different stakeholders often view the system from differing levels of detail, it is necessary to define requirements at more detailed levels than just the overall system-of-interest. This is accomplished by allocating the system requirements to the system elements. The activity of allocating requirements to system elements proceeds in parallel with the definition of the system architecture. Some requirements cannot be derived until some portion of the design evolves. There may be multiple iterations between the requirements analysis and architectural design to resolve trade-offs between the requirements and architecture (ISO/IEC/IEEE 29148 2011).

So, system descriptions are refined and corrected in the course of the project, and subsystems may need their own design steps. The iteration and recursion can be generalised by saying that (see **Table 4**):

- first there is an overall concept (a black-box) of a future system or situation, i.e. something existing the real world
- facts, problems, threats, needs and requirements are attached to it in the form of *statements* about its behaviour, properties, etc.
- on the basis of this and other domain knowledge, the functions and structure of the system are refined by adding more details
- the higher-level requirements are allocated to the newly added functions and system elements
- at each stage, the system model is evaluated and corrected if necessary

This approach allows designers to start their work by outlining the concept of a “real system”, typically having a broad scope, such as the whole industrial plant instead of the control system being designed. More abstract information, such as users’ needs, is then attached to that concept.

Table 4. Evolution of the system model.

Step	System model	Description
Concept definition		The first idea of a future system and external entities (Ext) is outlined.
Concept analysis		Statements (S) expressing facts, problems, requirements etc. are added.
Refinement		Details are added, for example functions (F) and parts (P). Parts can again be concepts of lower-level subsystems.

Defining life-cycle activities is not enough. Efficient project management requires that progress is evaluated at certain points of time and decisions made about directing the course of actions. As illustrated by **Figure 23**, the life-cycle of a system divided by *milestones* into consecutive, non-overlapping periods of time called *life-cycle phases*². Activities can cross phase boundaries, and they be performed several times in iterative design. But it does not make sense to say that life-cycle “stages can be overlapping”, as often is done, even in international standards (e.g. ISO/IEC/IEEE 15288 and ISO/IEC TR 24748-1). Here we make a distinction between life-cycle phases and the substance of engineering activities. Therefore, we use the term *life-cycle phase* and require that they are strictly sequential. However, the life-cycle phases of subsystems and corresponding subprojects can be shifted in time. Each subproject and design area can have its own phases and milestones that must be synchronised at the major milestones of the whole project.

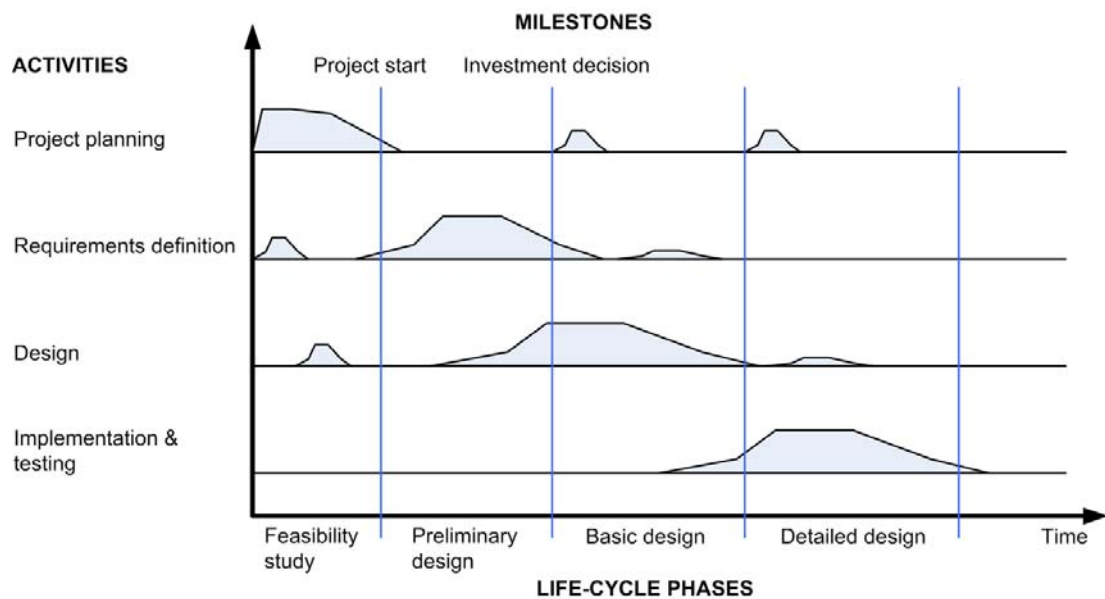


Figure 23. Distribution of engineering activities to life-cycle phases (modified from Booch et al. 1999).

The diagram in **Figure 24** summarises some key points. A project aims at a new or upgraded system. Designers in *project organisations* participate in engineering *activities* that create and modify various work items, here termed as “work areas” comprising sections in document repository, the system model or elements of the physical system. A work area might include, for example, the whole project, a subsystem model or manufacturing and assembly of the physical system. The work around each work area is divided into *life-cycle phases*, each starting at one *milestone* and ending at another. Milestones are decision points where *baselines* are frozen, active life-cycle phases declared as completed and next ones started. Therefore, at a given point of time, a project is in exactly one life-cycle phase. For example, we might say the “now that the investment decision was made, the I&C system project is in the basic design phase”. In other words, life-cycle phases of a specific system don’t overlap and a clear distinction is made between engineering activities and life-cycle phases.

² Some standards and guidelines use the word “stage” instead of “phase”.

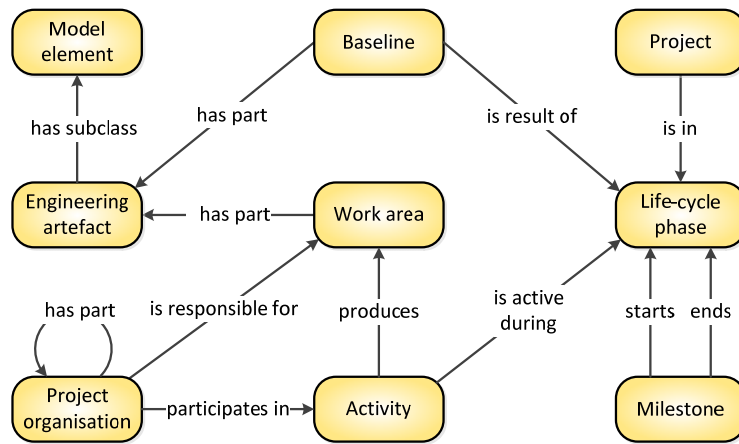


Figure 24. Main elements of the project model.

5. Meaning and representation of requirements

The focus of this report is on requirements that, in general, express features that a system shall have. The purpose of this chapter is to clarify how the term “requirement” should actually be understood and how requirements could be expressed in association with other elements in the system model.

Requirements start growing early during current situation and mission analysis and user requirements specification but are refined and maintained during the whole life-cycle in parallel with other engineering activities. As claimed in the previous chapter, there must be an idea of a future system before anything can be said about it. Changes and details can come up at any stage for design. As illustrated in **Figure 25**, the starting point of design is in the *business requirements* of the owner/operator, originating from experienced problems or identified technical or commercial opportunities. In regulated areas, also laws and standards are a source of these top-level requirements. *User requirements* specify what the users want to accomplish with the system-of-interest and other elements of the enclosing system. User requirements are closely related to the *Concept of Operations* (ConOps). The allocation of responsibilities leads to more technical *system requirements* which are then further allocated to subsystems, software and hardware. These requirements, often called “product requirements”, are concerned with the system of interest. Some others, so called “process requirements”, are associated with the life-cycle activities. To avoid confusion around the word “process”, we call them *life-cycle requirements* and divide them into two groups, *engineering process requirements* and *O&M requirements*. At all levels, standards and regulations provide additional input for the requirements definition. In addition, there is communication with the design and procurement activities that may provide feed-back leading to modifications made to the requirements.

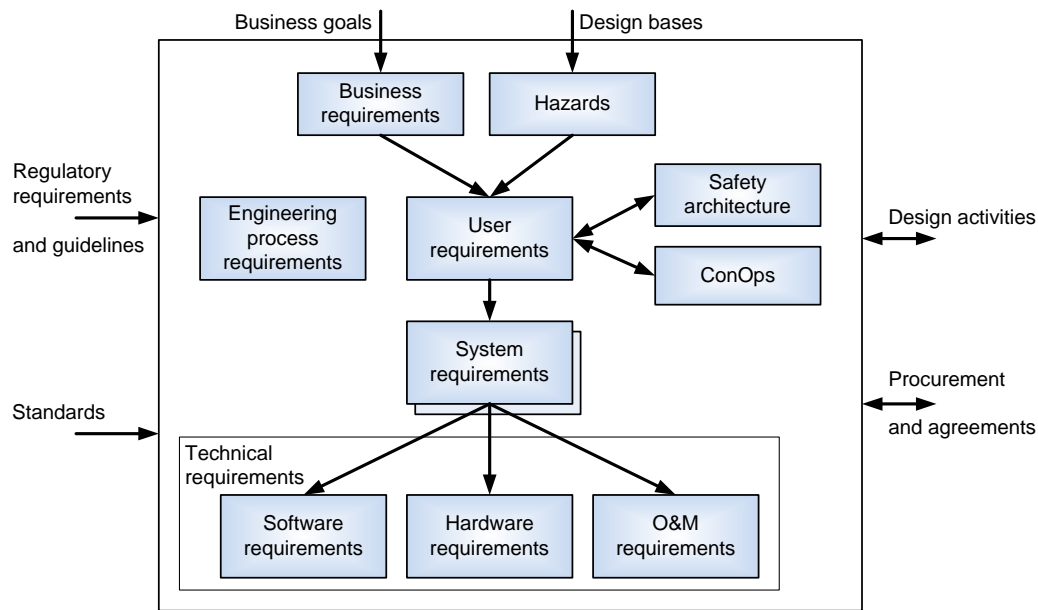


Figure 25. Flow-down of requirement types.

5.1 Understanding requirements as statements

As suggested above, a *system model* consists of *model elements* that represent:

- systems, system elements and components
- entity types (e.g. device types) and individuals
- capabilities and functions of systems
- operational states of systems
- operating modes of systems and their functions
- information items and materials
- activities, tasks and actions
- behaviours of one or more systems and system elements (use cases and scenarios)
- relationships and properties of various entities

Similarly we have a project model consisting of project organisations, artefact types, activities, life-cycle phases, milestones, etc.

Let's now think what the designer actually does, when she/he adds a model element to the system model. Our interpretation is that she says something about the current or future states-of-affairs related to the system-of-interest and its environment. In other words, the designer makes a *statement*, intended to be received by other stakeholders, sometimes even by the designer herself later in the course of the project. This highlights the idea that one purpose of a model is to foster communication among the parties involved in the life-cycle of a system. Model elements express different kinds of statements, for example:

- existence of a system element
- property of a system element
- relationship between two or more system elements
- dynamic behaviour of a system element

Moreover, the discussion between stakeholders can be classified by using the concept of the speech act theory (Searle & Vanderveken 1985). The “illocutionary force” of an utterance describes its intended meaning, for example requesting, commanding, promising, etc. Utterances that stakeholders make in communicating with the engineer are intended to advance stakeholders’ personal desires, intentions, beliefs, and attitudes. The engineer should distinguish its content from its illocutionary force in order to know to represent the content as a requirement or otherwise (Jureta et al. 2008). It is important to agree in advance on the specific keywords and terms that signal the presence of a requirement among other non-mandatory statements (ISO/IEC/IEEE 29148 2011, p. 9). A common approach is to use of words like “shall”, “should” and “may”. When applied to system modelling, we can say that statements have various “communicative intents” describing their illocutionary force, for example:

- fact: “The current **power plant** is 30 years old”
- problem: “The existing automation system is obsolete”
- plan: “The **upgrade project** of the automation will be started next year”
- goal: “usability should be carefully considered in the design of the control room”
- specification: “The new **control system** will include a separate **protection system**”
- requirement: “The availability of the **protection system** shall exceed 99.9%”
- requirement: “**All components** of the protection system shall have a certification”

We can see from the examples above that statements say something about other statements (or the real-world entities they represent). For example, the specified existence of the protection system above is a statement and so is the availability requirement associated with it. Note also that a statement can concern a whole set of entities as is the case in the last example.

In general, all model elements can be understood as statements that can be associated with one or more other statements. This situation is illustrated in **Figure 26** where the (specified or actual) physical system elements are shown as rectangles and other statements attached to them as ellipses.

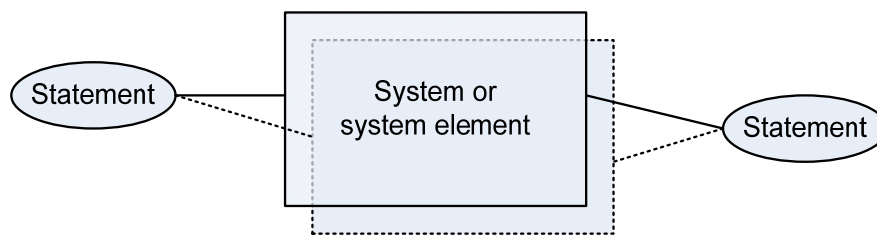


Figure 26. Requirements and other kinds of *statements* are attached to model elements that represent future system elements.

Table 5 gives our draft definitions of the primary statement intents. The most interesting of them are goal, requirement and specification. Their differences should be clarified even if a definite borderline between them can’t be drawn. Goals are soft in the sense that they give a direction but not an exact target that should be reached (**Figure 27**). Requirements are mandatory but avoid saying how they should be satisfied. Specification, in turn, differs from requirements in that it ties the hands of the designer more exactly.

Note: Some seemingly strict requirements may actually be goals since relaxing them a bit might lead to a better overall solution.

Table 5. Communicative intents of a statement.

<i>Fact:</i>	Known to be true now and/or in the future
<i>Problem:</i>	An undesired state of affairs
<i>Need:</i>	Desire of a stakeholder
<i>Assumption:</i>	Expected to be true
<i>Suggestion:</i>	A draft specification or plan
<i>Attitude:</i>	Value preference of a stakeholder, e.g. a need or a problem
<i>Goal:</i>	Defines a target to pursue, as far as possible <ul style="list-style-type: none"> • defines a policy to guide design activities • not necessarily (directly) measurable (soft-goal) • not necessary to reach the target (should)
<i>Requirement:</i>	Something that shall be fulfilled by a future state-of-affairs <ul style="list-style-type: none"> • defines what the stakeholder must be able to do and how well • sees the subject as a black-box, structure and details left open • must be verifiable • result of a negotiation process, agreed among stakeholders • no deviations are permitted (shall)
<i>Claim:</i>	Postulated system property, e.g. that it satisfies a requirement (used e.g. in safety cases)
<i>Specification:</i>	Defines the (functional and physical) elements and properties that the stakeholder wants to <u>see</u> in the solution. Details can, however, be added.
<i>Plan:</i>	Defines intended actions of an organisation (or a system)
<i>Example:</i>	Description of an individual, e.g. a possible solution or scenario
<i>Note:</i>	Provides additional information

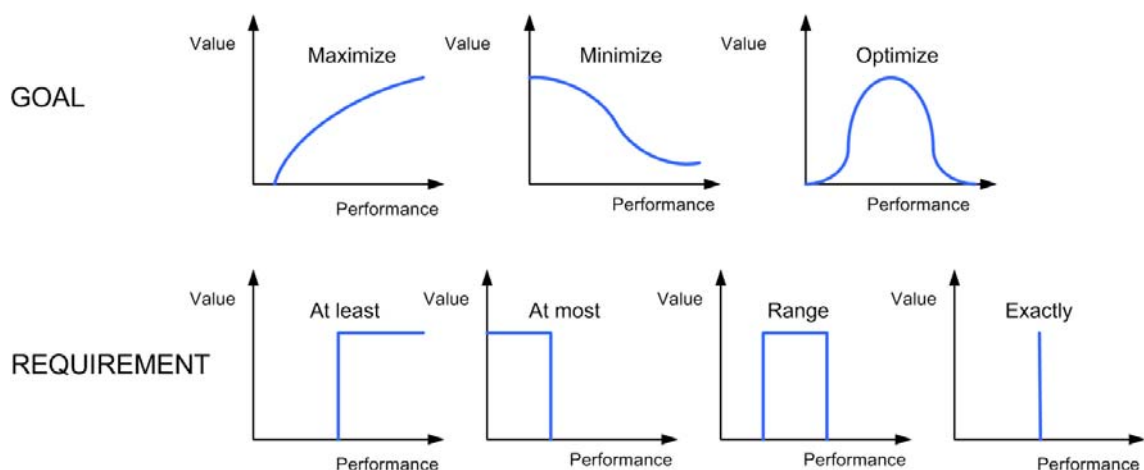


Figure 27. Typical value functions of goals and requirements (adapted from Hull et al. 2002).

Understanding elements of the system model as “statements” has a comfortable implication. Statements can be expressed with any modelling formalism appropriate for the domain and purpose, such as natural language, lists or diagrams. This holds also for requirements. Any statement, let it be a natural language sentence or a formal function block diagram, becomes a requirement when the designer says with its communicative *intent* that it must be read as a “requirement”. The advantage is the flexibility of selecting the notation used for requirement representation.

5.2 Associating requirements with the system model

Statements say something about the real world, and therefore also requirements have a target. As illustrated in **Figure 28**, different types of requirements are associated with elements on different levels of the plant hierarchy. Business requirements are concerned with the whole plant and its operations, while software requirements focus on the I&C system, for example. User requirements relate to the interplay of several plant elements from the end user’s point of view.

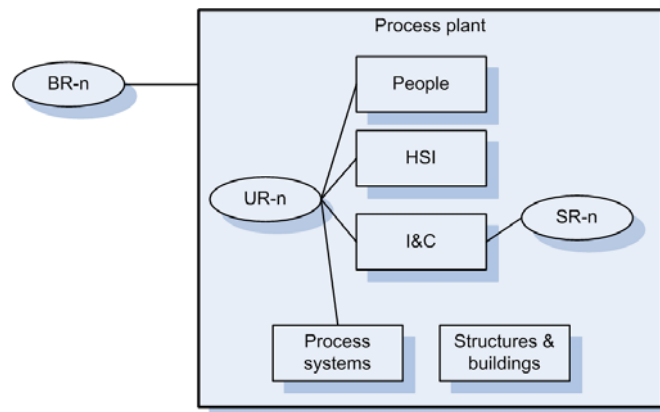


Figure 28. Business requirements (BR) are typically attached to the whole plant, user requirements (UR) describe the orchestration of the main plant elements and system requirements (SR) are attached to a specific plant elements, such as an I&C system.

Engineers tend to think in technical terms like devices and software. This is perhaps not the ideal way, but it might be practical in the design of technical systems. The hierarchical decomposition of the physical system, such as an industrial process plant, can provide the skeleton to which other model elements are attached as shown in **Figure 29**. For example, the whole plant has manufacturing activities and operational states. Requirements can be associated with any kind of model element, e.g. a system, activity, state, function and even with another requirement. If system details have not yet been specified, requirements can be associated, in an appropriate form, with relevant higher-level elements.

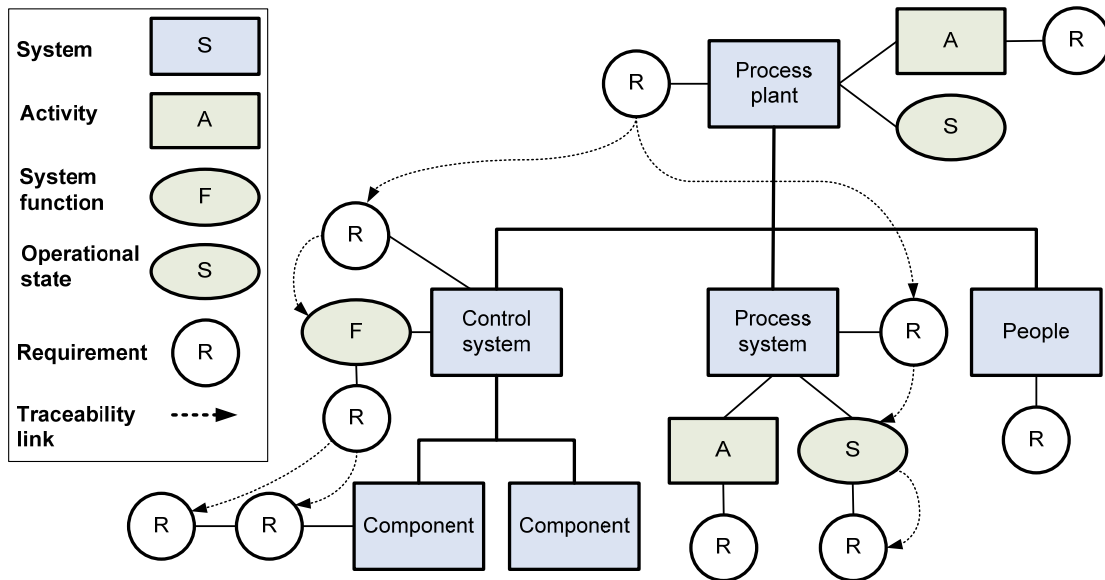


Figure 29. Activities, states, functions and requirements attached to process plant elements.

Finally, we should observe that the system model evolves and grows during its life-cycle. With time, a system element, for example an instrument of an I&C system, can have several descriptions like “as required” by the customer, “as specified” by the designer and “as installed” after a modification. This idea can be found, for example in the purchase of industrial-process measurement and control devices (Löffelmann et al. 2007, IEC 61987-10 2009). As illustrated in **Figure 30** for a temperature measurement as required by the process engineer and specified by a control engineer. A supplier may offer and deliver an instrument that exceeds the performance characteristics specified in the inquiry. During plant maintenance, it can be replaced by an even better device. All this knowledge should be stored in the system model for traceability and future modifications.

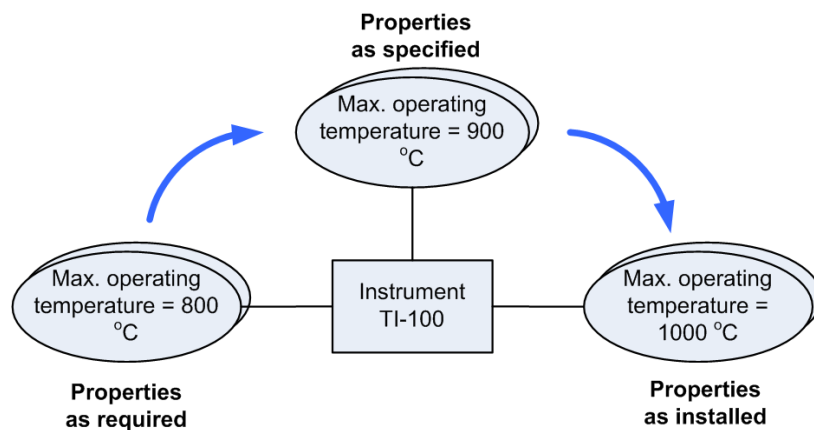


Figure 30. Tracing required, specified and as-built properties of a field instrument.

5.3 How to express requirement statements

All these model elements carry various stakeholders' (e.g. customer's and designer's) *statements* about existing or future real-world entities, and sometimes statements about the model element itself (as metadata). Statements have various *communicative intents*, such as fact, problem, goal, requirement and specification. All these are needed in requirements definition. The benefit of the explicit intent attribute is that the contents of the statement itself can be represented in any formalism suitable for the purpose. For example, an old wiring diagram in **Figure 31** can be labelled to be a fact about the current situation or a functional requirement of the new, upgraded control system.

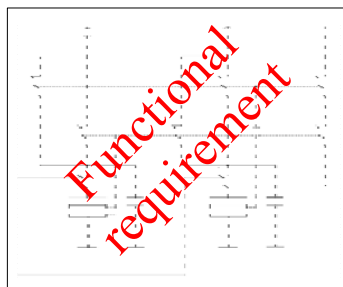


Figure 31. Virtually any description can be labelled with the communicative intent to represent a requirement instead of a design (adapted from Malm & Kivipuro 2004).

When understood as model elements, requirements, facts, assumptions, etc. can be stored in a relational database as records with various attributes attached to a natural language statement. **Table 6** gives an example where a functional requirement is associated with the model element called “I&C system” and linked to a separate graphical object “SFC-001”. Only some of the requirement attributes are shown.

Table 6. Some requirement attributes in tabular form.

<i>ID:</i> R1234	<i>Title:</i> Automatic start-up		
<i>Intent:</i> <i>Functional requirement</i>	<i>Status:</i> Draft	<i>Date:</i> 29.4.2011	<i>Author:</i> TET
<i>Statement:</i> The <i>I&C system</i> implements the plant start-up procedure. A typical plant start-up sequence is shown in the sequential function chart <i>SFC-001</i> .			

It is important to agree in advance on the specific keywords and terms that signal the presence of a requirement. A common approach is to stipulate the following (ISO/IEC/IEEE 29148 2011):

- Requirements are mandatory binding provisions and use ‘shall’.
- Statements of fact, futurity, or a declaration of purpose are non-mandatory, non-binding provisions and use ‘will’. ‘Will’ can also be used to establish context or limitations of use. However, ‘will’ can be construed as legally binding, so it is best to avoid using it for requirements.
- Preferences or goals are desired, non-mandatory, non-binding provisions and use ‘should’.
- Suggestions or allowances are non-mandatory, non-binding provisions and use ‘may’.

- Non-requirements, such as descriptive text, use verbs such as 'are', 'is', and 'was'. It is best to avoid using the term 'must', due to potential misinterpretation as a requirement.

This guidance is easily applied in the case of textual requirement statements. However, a statement can be expressed in other ways, e.g. using graphics, voice or a video clip, not necessarily containing words like "shall". Moreover, a statement initially considered as a mandatory requirement can be demoted to be a goal. Therefore, we suggest keeping the communicative *intent* of the statement as a separate attribute as illustrated in **Table 6**. For readability, textual requirements can use the guidance above, even if it is redundant information that must be updated when the intent of the statement changes.

Often requirements are expressed in well-formed natural language, as recommended in ISO/IEC/IEEE 29148 (2011, p. 9), EARS (Mavin et al. 2009) and Boilerplates (Hull et al. 2002). Also computer tools are available for authoring and analysing requirements written in a Controlled Natural Language that provides a limited vocabulary and set of sentence templates. A literature review and tool concept for model checking I&C functions can be found in Tommila & Pakonen (2014).

To close our discussion on the nature of requirements, **Figure 32** illustrates the main concepts for requirements modelling. In principle, all elements in a system model can be understood as *statements* about the current or future state of affairs. The contents of a statement can be, for example, text, graphics or a formal language. A statement says something about entities in the real world and, therefore, concerns other *model elements* that represent them in the model. The *communicative intent* of a statement explicates whether it should be interpreted as a goal, fact, assumption, etc. A *requirement* expresses a mandatory feature but avoids specifying the solution prematurely. The term *constraint* refers to a requirement that deliberately does so for some reason. Requirements (and other statements as well) can be classified using various criteria, such as originating stakeholder, target concerned or content. **Figure 32** gives only some common examples.

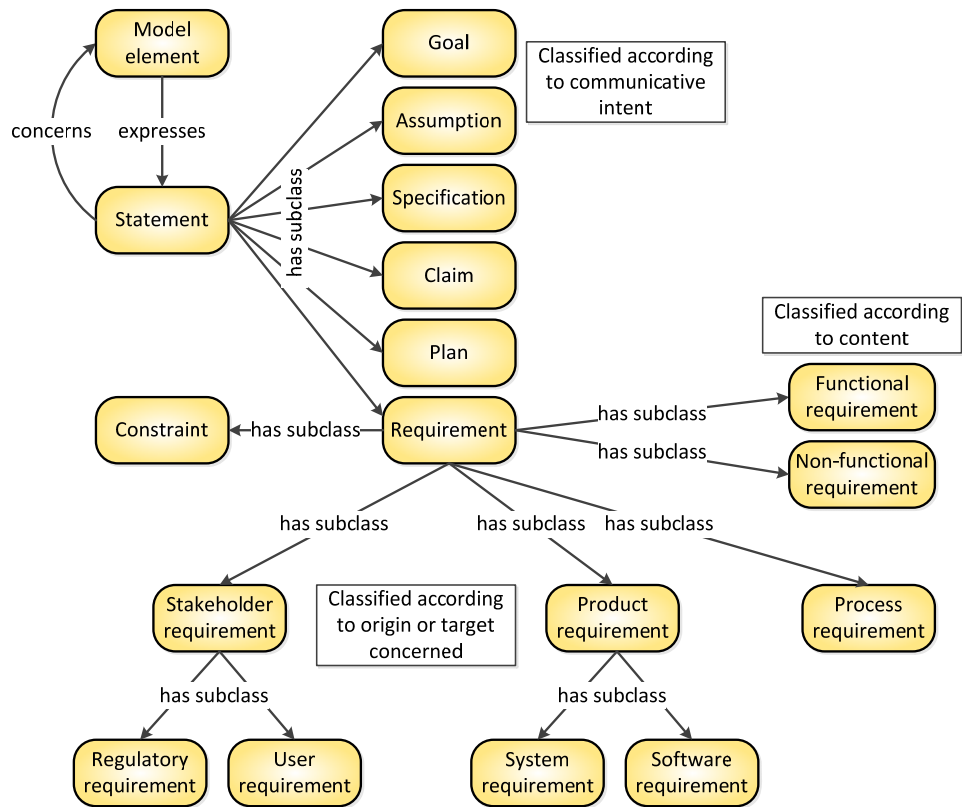


Figure 32. Main concepts related to requirements.

6. Traceability relationships

In complex and critical applications, it is necessary to manage dependencies between model elements. This is required for the verification and validation of the design solutions, as well as for assessing and managing the changes made during the design process. Traceability is valuable to the developers, as well as to plant owners and operators, for example by the way of better management of dependencies and utilisation of design knowledge. Today, traceability is also one of the basic principles of software safety standards and a key prerequisite for certification of software (Nejati et al. 2011). Therefore, this section gives an introduction to the topic and outlines a simple conceptual model to describe the traceability information. See the report by Tommila (2013) for a literature review on the subject.

The basic idea of traceability is old and widely accepted but its exact meaning and practical methods are still under development. So, advanced traceability is not often seen in the industry (Gotel et al. 2012). Instead, traceability still tends to be undertaken in rather ad hoc ways, with unpredictable results (Mäder et al. 2009a). Standards and guidelines describe the need for traceability in general terms, but explicit guidance on specific practices is scarce (Gotel et al. 2012). Though widely accepted as beneficial, the costs associated with traceability can be high. Traceability information must be updated with the addition of new links, removal of existing links, and changes in existing links. Without maintenance, traceability relations between elements get lost or represent false information. (Mäder & Gotel 2011.)

The literature gives many definitions for the term traceability (Tommila 2013). For the purposes of this report, especially model-based design, we come to the following interpretation of the term:

Traceability is a characteristic of design information about a system (the system model) that makes it possible to find out how (when, by whom...) and why model elements have been derived from external information sources (stakeholders and standards), from previous versions or from other model elements.

Note: This definition is not intended to cover the traceability of real-world individuals, such as nuclear material or particular devices currently installed into the system. To do that, additional information would be needed to record the status and evolution of the system configuration.

This definition means that traceability should cover both derivations paths and the progress of individual model elements in time (**Figure 33**). Moreover, design information should support both backward traceability (from requirements to their sources) and forward traceability (from requirements to design solutions). This traceability information should be visible at both ends of the links (from and to traceability, Kotonya & Sommerville 1998).

Note: Traceability concerns all design (and maintenance) information, not just requirements.

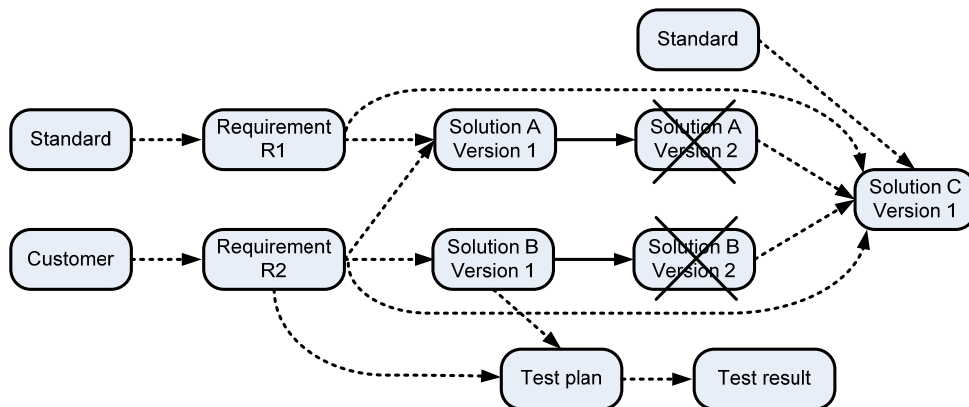


Figure 33. Examples of traceability links between model elements.

Traceability requires some information, the *traces* to be created and maintained in addition to the basic design data. First of all, previous versions of model elements must be archived with associated change history. Secondly, traceability relationships, possibly including lots of attribute data, between model elements and their versions need to be maintained. There are several types of traceability relationships in the system model, for example:

- A requirement is allocated to (concerns) a system element (as demanded by the system architect).
- A system element satisfies a requirement (as claimed by the software developer).
- A model element refines a requirement stated in general terms (e.g. a use case may be used to refine a text-based functional requirement, OMG 2010).
- A requirement is derived e.g. by inference or decomposition, from a higher-level requirement (an availability requirement might be decomposed into requirements concerning system reliability, accessibility for repair and capabilities of the maintenance organisation).
- A version of a model element replaces the previous version.
- A test plan and test result verifies whether a requirement is satisfied.
- A requirement depends on another requirement.

Requirements are themselves often linked to each other, especially because one (atomic) requirement should state only one thing. For example, the sentence “Reactor temperature shall be measured with an accuracy of 1 °C.” actually includes two requirement statements: R1, “Reactor temperature shall be measured”, and R2, “The accuracy of the reactor temperature measurement R1 shall be 1 °C”.

Traceability is often represented in matrix form. This has, however, a number of disadvantages when the amount of information grows. Hyper-linked documents make traceability information visible but are also hard to maintain. The user may also get lost when traversing the links in the design space. Traceability is easiest to achieve with some kind of database support, preferably storing traceability information separately from design data (Hull et al. 2002).

Maintaining traceability links represents a major step forward, but some additional information may be needed for advanced traceability. Hull et al. (2002, p. 143) introduce satisfaction arguments to explain explicitly how the requirements are being satisfied. Interestingly, this idea seems similar to the *safety case* approach (Kelly 1999) used to assess safety-critical systems in many application areas. More generally, the term *assurance case* is used to refer to a structured

assessment of system properties against a set of top-level claims, e.g. dependability, security or usability (see Tommila et al. 2014). As illustrated in **Figure 34**, design and assessment are linked and can apply similar reasoning principles, though the argumentation sometimes goes in opposite directions.

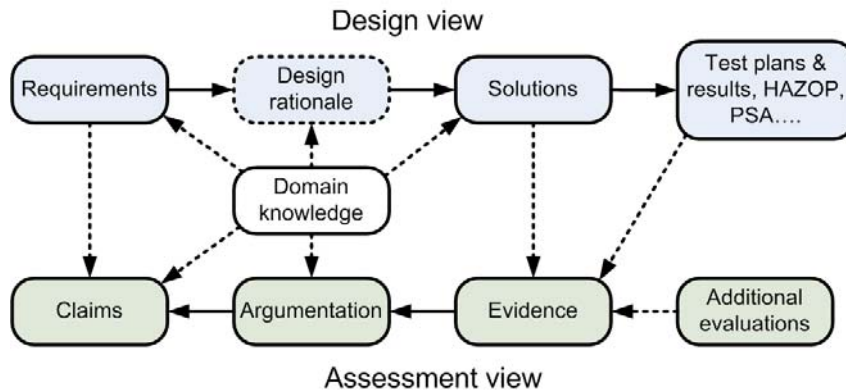


Figure 34. Linking reasoning in design (synthesis) and safety assessment (analysis).

Addressing traceability without requiring more structure and precision on the design artefacts is unlikely to succeed because traceability links cannot be adequately defined over poorly structured and imprecise artefacts (Nejati et al. 2011). In addition, a “traceability information model” must be defined to specify the contents and well-formedness criteria for the traceability links between model elements (Mäder et al. 2009a, Nejati et al. 2011). The idea of a traceability information model is close to the approach used in the SAREMAN project. In fact, the following complements our conceptual model with a simple traceability information model.

We interpreted *traceability* above as qualitative characteristic of the system and project models. A traceable model contains traceability information, *traces*, that allow interested parties to determine how the design solutions have emerged. The model can give answers to questions like

- What information is the artefact based on?
- Where has it been used as input information?
- Why is it a good solution?
- Who made it and when?
- How will it be verified?
- How has it seen modified (when, why, who)?
- What are its dependencies on and implications of changes to other artefacts?

The design data itself, combined with intelligent queries and views provided to the user, can give answers to some of these questions. However, additional information must be added that records the history and the reasoning of the designers. This means new attributes of model elements and additional relationships between them. The project organisation needs to decide what information to record and at what level of granularity. In safety systems, the criteria are usually set by regulations and the needs of safety demonstrations. In the following, we investigate what kind of information the traces might contain and how it could be modelled.

6.1 Recording modifications

Changes can be classified in several dimensions, such as model structure, meaning or purpose. For example, we can identify the following structural change types (modified from Sivertsen et al. 2005 and Mäder et al. 2009b):

- creating a new model element with no prior history
- deleting an existing element
- splitting an existing element, thereby creating a number of new elements
- combining existing elements into a new element
- replacing existing elements by a (completely) new element
- moving model elements in the “product breakdown structure”
- modifying an element in a way that is externally visible (e.g. interface or behaviour)
- modifying an element internally without changing its meaning (e.g. editorial corrections).

The list above focuses on the contents and structure of the model without considering the reasons and purpose of the change. We might say in general that a change of a model element can be triggered by 1) normal progress of design activities; 2) a deficiency observed in the current versions; or 3) an external change. Modification of some elements easily leads to the situation where the properties or structure of other elements must be modified. Depending on the types of model elements, links and their attributes a change can put linked model elements in a “suspect” state and lead to adding/improving, modifying or removing/reducing features of the system. So, we introduce the following (overlapping) semantic change types:

- refinement: additional details are added to system model during the normal design process
- enhancement: system design is improved to better fulfil the stated needs
- reduction: unnecessary features are removed
- correction: an observed design flaw is removed
- adaptation: design is modified due to a change in external circumstances or a modification made elsewhere in the model

A comprehensive taxonomy of structural and semantic changes is beyond our scope. The upper part of **Figure 35** illustrates simplified main concepts related to the change history. A model element is subject to an action called *change*, and a new version is a result of the change. A change object contains information that describes the relationship between the old and new versions. The attribute change type can be used to tell what happens (see the lists above). Note that old and new versions are stored in the design database and are, therefore, thought of here as different objects. A change action is usually (but not necessarily) part of a *change request*. Both the changes and the change request should have descriptions of the reasons behind the change and the modifications to be made.

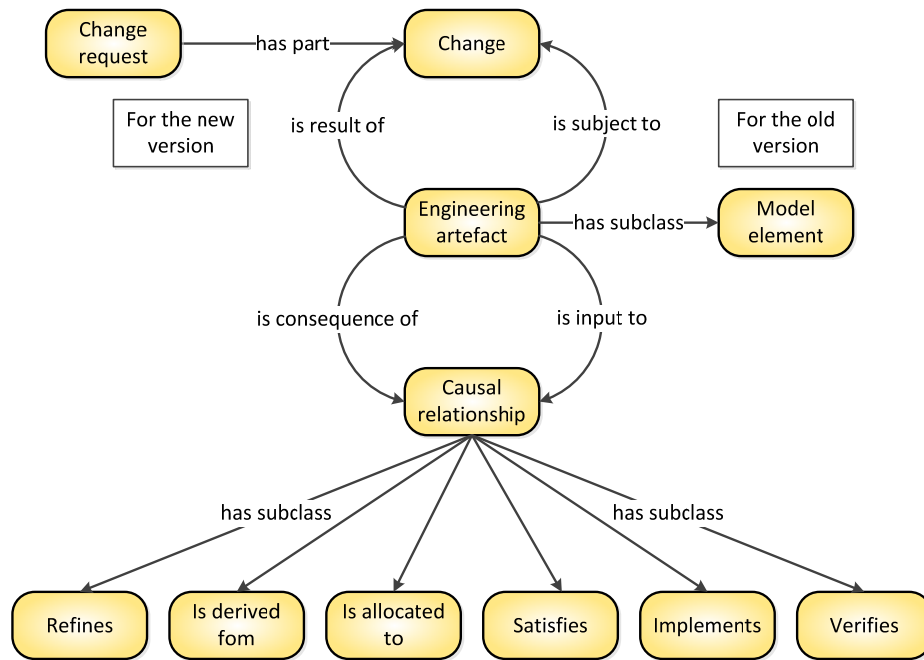


Figure 35. Concepts of change history and design reasoning.

6.2 Recording designer's reasoning

The change history describes what kind of modifications have been made and why. However, it doesn't necessarily explain why the resulting artefacts are what they are, i.e. how they have been derived from input data. Something must be added to record the reasoning of the designer. In this case, new model elements or their versions are derived from external information and/or other model elements that are not going to be modified. The traces need to tell, what information was used, how and why. The change history is about evolution in time, while the other traces are concerned with causality.

To record the reasons for the selected solutions, we need two kinds of information, the sources and the design rationale. The sources describe the relationship of the new model elements to the input data. Types of relationships are numerous, and will be briefly discussed below. Design rationale, in turn, contains the argumentation of the designer. It's a complex issue per se (e.g. Shum & Hammond 1994, Tang 2007), so we can't go deeper into the realm of design rationale here.

Many types of causal relationships in design data can be found in standards and literature. Each organisation must decide the link types and their semantics suitable for their needs and available resources. The set of possible link types and their interpretations below is a combination from several sources:

- **Refines:** States that a model element contains additional information to another model element. For example, a function block diagram may be used to refine a text-based functional requirement (adapted from OMG 2010). The inverse relation might be named as "is refined in".

- Is derived from: A statement of the designer to say that another model element has been used as input data and, consequently, influenced the resulting model element. The inverse relation is “is used for”.
- Is allocated to: A statement telling that a model element has a role (responsibility) in achieving the targets defined in another model element. Inverse “is responsible for”. For example, a system-level requirement can be allocated to a system function. A function can be allocated to a physical system element or a task allocated to a designer.
- Satisfies: A statement of the designer claiming that a model element is specified in a way that fulfils a requirement. Inverse “is satisfied by”. Note that the previous relationship “is allocated to” is a statement made by the requirements engineer, while this one is designer’s response to that demand.
- Implements: States that a designed device or piece of software is intended to fulfil a specification, e.g. to perform a function.
- Depends: Model elements are mutually dependent.
- Verifies: States that a planned design review or test case is intended to verify that a model element satisfies a requirement attached or allocated to it.
- Reports: States that the results of an activity instance or task execution realise what was intended by a plan. For example, a test report reports a test case.

Note that some relationships can be characterised as needs or plans, i.e. something that needs to be considered in next design steps. Some others are sort of historical data reporting what has been decided by the designer or done by the tester. Different from the *changes* above, causal relationships retain their identity over modifications. In other words, they are maintained as part of the design data, while changes are snapshots that are not modified once they are closed. Consequently, causal relationships can have a number of associated changes recording the modification that have been made to them. The relationships are usually many-to-many and may concern all types of model elements, not just requirements and their solutions. The discussion above is summarised in the lower part of **Figure 35** above.

We have learned from a small survey carried out in (Tommila 2013) that traceability is not a trivial issue. It is required by the standards and regulations concerning safety-critical systems. However, it may be complicated and expensive to implement. Even if the idea is old, no single definition or a way to implement traceability can be found. Commercial software tools for requirements management and engineering provide some but fairly limited support for traceability. However, the issue is widely accepted as a critical one, and much research is going on around it, especially in the area of model-driven development. In the long run, traceability information models are seen as one approach to manage the traces and to enhance their (automated) creation, maintenance and utilisation for change impact analysis and system assessment. Meanwhile, practical solutions should be sought for. To be accepted and effective, traceability policies should be adapted to the current practices and needs of the domain. However, traceability can’t be developed in isolation, but the underlying principles and structure of design artefacts, as well as the design and management processes should be improved simultaneously.

7. Modelling nuclear power plant systems

The previous chapters discussed systems modelling in very general terms. The aim was to clarify certain fundamental concepts and to suggest some new viewpoints that might be valuable to the nuclear domain. The purpose of this chapter is to map some of the concepts to the terminology commonly used in the NPP domain, perhaps with slightly new interpretations and definitions. The aim of this analysis is to end up with a limited set of practical terms (found in Appendix B).

7.1 Observations on the current terminology

Definitions in the literature and discussions with domain experts give rise to the following observations:

- The definition of “requirement” does not significantly differ from text books or standards in other domains. In other words, it is defined in general and somewhat ambiguous terms. The importance of regulatory issues is visible, especially in the glossary of IAEA (2007). Compound phrases, such as “regulatory requirement”, might be a way to indicate the viewpoint taken and to avoid ambiguity.
- In nuclear power plants, systems consist of “subsystems”, structures and components, but their essential character and relations are not clearly defined. The term “system” has a technical interpretation, which means that humans are seen rather as external users of a system than as its constituents. A system element (subsystem or component) can be part of only one higher-level system.
- The term “activity” seems not to have a specific use in NPP standards, except for “radio activity”. It is used in its general meaning, e.g. “activities of humans”. The term “function” is often applied instead of “activity”. However, saying that functions and activities are synonyms is misleading since in our view activities can be achieved by using several alternative system functions. Instead it is possible to speak in a more general terms about “functions required at plant level”, e.g. fundamental safety functions, that are then implemented as concrete functions of various plant systems.
- In the context of energy production, the term *process* refers to the chain of activities for transforming inputs to the outputs, not to the *process system* (process equipment) used for that. This is consistent to what was suggested in Chapter 3. In the engineering domain, processes are carried out by the project organisation.
- The words “task” and “action” are often used without a precise definition. Actions can be understood as being parts of tasks, but it’s unclear whether tasks are parts of activities or processes. Moreover, there seem to be different opinions about a “task” being allocated to humans only or either to humans or automated systems. We interpret that tasks and actions

can be performed both by humans and technical systems. Actions are “atomic” in the sense that they are not further decomposed.

- The term “function” has not been clearly defined in the NPP standards. In expressions like “safety function” it is understood as a “purpose or objective” that can be described without reference to the resources used to achieve it. However, the term “function” is sometimes used in its context, e.g. “plant functions”, or associated with a specific system, e.g. “I&C system whose functions are performed by microprocessors”. Some standards don’t even make a clear distinction between goals and functions.
- Operational states and operating modes seem to be understood in a way similar to many other areas of industrial automation. Nuclear reactors can have standard states like power operation, hot and cold shutdown, refuelling etc. With respect to safety, IAEA (2007) operational states that cover normal operation and anticipated operational occurrences and accident conditions including design basis accidents and severe accidents.
- In the context of the reporting and analysis of incidents, an “event” is any occurrence the consequences or potential consequences of which are not negligible from the point of view of protection or safety (IAEA 2007). This definition is very close to actual or potential accident and does not say whether an event has a zero duration. The related term “scenario” is a postulated or assumed set of conditions and/or events, which can also be understood as an event.

The examples above make us believe that the terms used in the nuclear domain should be clarified. The sections below include a discussion and some suggestions.

7.2 Modelling NPP system structure and behaviour

Figure 36 illustrates the main physical constituents of a nuclear power plant. We say that a nuclear power plant is a *sociotechnical system* that consists of various kinds of “NPP elements” (a helper class corresponding to system element), such as *organisational units* (people, e.g. *shift*), plant equipment here called *plant items* and *areas* (spaces, e.g. controlled area). Plant items include technical *systems* (devices, software, data, e.g. *process system* or *I&C system*), *structures* (buildings, walls, pipe supports) and atomic *components*. NPP elements are located within an *area*, which is defined in terms of structural elements like walls. All NPP elements can be hierarchically decomposed into lower-level elements. Different from the general system concept discussed in Section 3.1, an NPP element is part of exactly one larger element. The figure shows that all combinations are not allowed. For example, process equipment should obviously not be a part of an organisational unit.

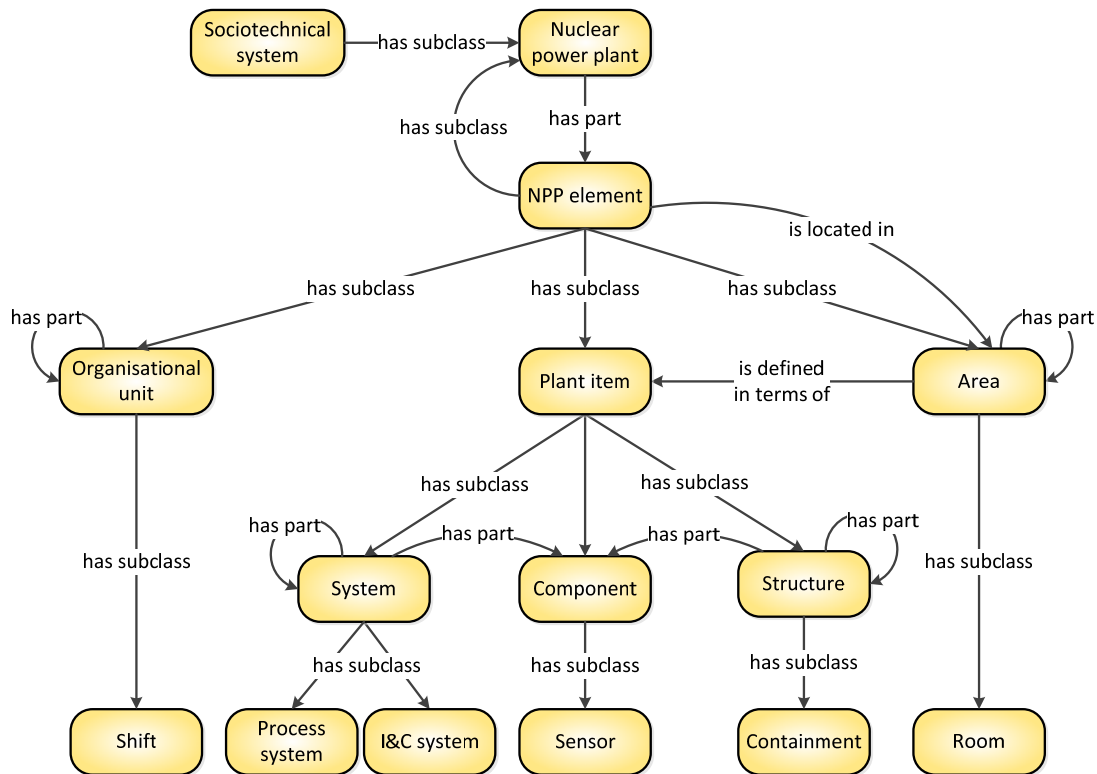


Figure 36. Main elements of a nuclear power plant with some concrete examples.

Figure 37 illustrates some functional concepts related to a nuclear power plant and its elements. These NPP elements are used to carry out *activities*, primarily the energy production *process*. “Activity” is goal-oriented doing intended to contribute to the achievement of the goals but leaving the means open. “Process” is a view to the network of activities highlighting the flow of work items and materials through it from a specific perspective. Activities can be decomposed into smaller “subactivities”, *tasks* and atomic *actions* with flows of material, energy and information between them. A term *task* refers here loosely to a part of an activity which can be clearly defined, for example in terms of sequential *actions* to be performed. For example, starting up a process section could be called a task. As parts of activity, tasks and actions are not necessarily allocated to any NPP elements at design time. During plant operation tasks are, however, performed by specific technical systems and humans. For example, the reactor operator can perform the action of “start process system” by pressing a push button in a display. This action triggers the function “startup sequence” provided by the I&C system. So, the action performed by one NPP element uses the services provided as a function by another NPP element.

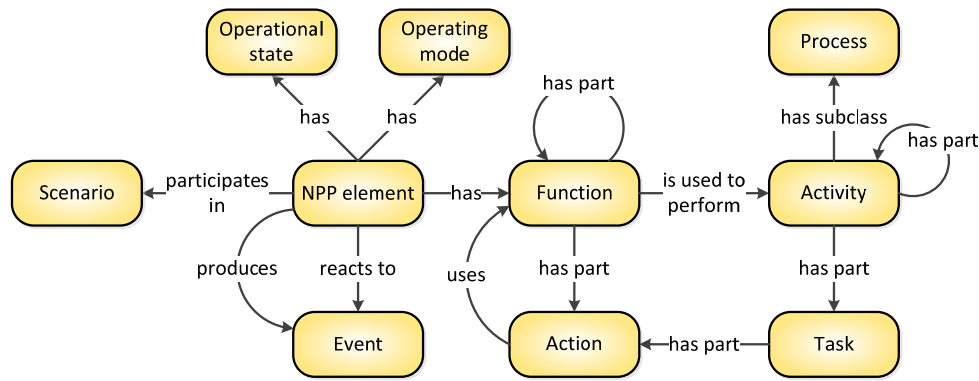


Figure 37. Functional concepts related to power plant systems.

Various NPP elements like technical systems and organisational units are designed to provide certain functions that are used to perform activities. Different from activities, functions don't "hang in open air" but are (required, designed or actual) capabilities of a system. For example, IEC 61513 (2011) defines an *I&C function* as a "function to control, operate and/or monitor a defined part of the process". The term is used by process engineers to structure the functional requirements for the I&C system. *Safety function* is an I&C function that is important from the point of view of safety (YVL B.1 2013). Some of these capabilities are intended to serve external "clients" while some others are internal functions not visible outside the system boundary. Even if associated with a system, the description of a function should not (definitely not during the initial design steps) specify how it will be implemented or allocated to the internal elements of the system. This reflects the general idea in engineering design that requirements, processes and functions should come before their technical solutions. An activity can be performed by several NPP elements and their functions, separately or in co-operation. A *function* can be understood as a capability of a specific system while an activity (or process) can be allocated to any system having appropriate capabilities. The distinction of activities and functions depends, however, on the needs of the modeller. In the case of a nuclear power plant, we have the option to use functions only. Plant-level functions, such as *fundamental safety functions*, can be accomplished by combining, in different ways, the functions provided by plant systems and components. So, energy production and removal of residual heat can be understood as functions of the whole plant.

All functions can be decomposed into "subfunctions" and flows of material, energy and information between them down to a level where atomic *actions* are encountered. Data-flow diagrams in software development and logic diagrams in control engineering are examples of possible representations. The subfunctions are allocated to the elements of the system. This allocation can take place in many ways, e.g. allocation of a function to one system element, to several redundant system elements or to several elements with a shared responsibility. The allocation can also be dependent on the situation resulting in complex and dynamic collaboration patterns between the human and technical system elements. This means that the subsystems have their own functions that together accomplish the higher-level function.

Functions are modelling elements of the *functional architecture* of a system and make up the core content of its *functional specification*. Functions are mostly derived from functional requirements and implemented as physical NPP elements like devices or software. Many functional requirements and even functions can be described in natural language using well-formed sentences and consistent, application-specific terminology. However, also, longer scenarios, tables and graphics are needed. In systems and software engineering, the graphical represen-

tations of the Unified Modeling Language (UML) are widely used. For example, activity and state diagrams can be used to describe power plant processes and operational states. Even features of the function block paradigm familiar to control engineers can be found in UML and its derivatives, such as the System Modeling Language (SySML). More domain-specific methods are available for control system design. For example, Function Block Diagrams (FBD, IEC 61131-3 2013) and Sequential Function Charts (SFC, IEC 60848 2002) are suitable for describing I&C functions in an implementation independent way. The Norwegian standard “System control diagram” (NORSOK I-005 2005) is another example of an approach to define exact but platform independent functions for the control system supplier. Also other suggestions to this direction can be found, but there are no widely accepted “automation specification languages” for the time being.

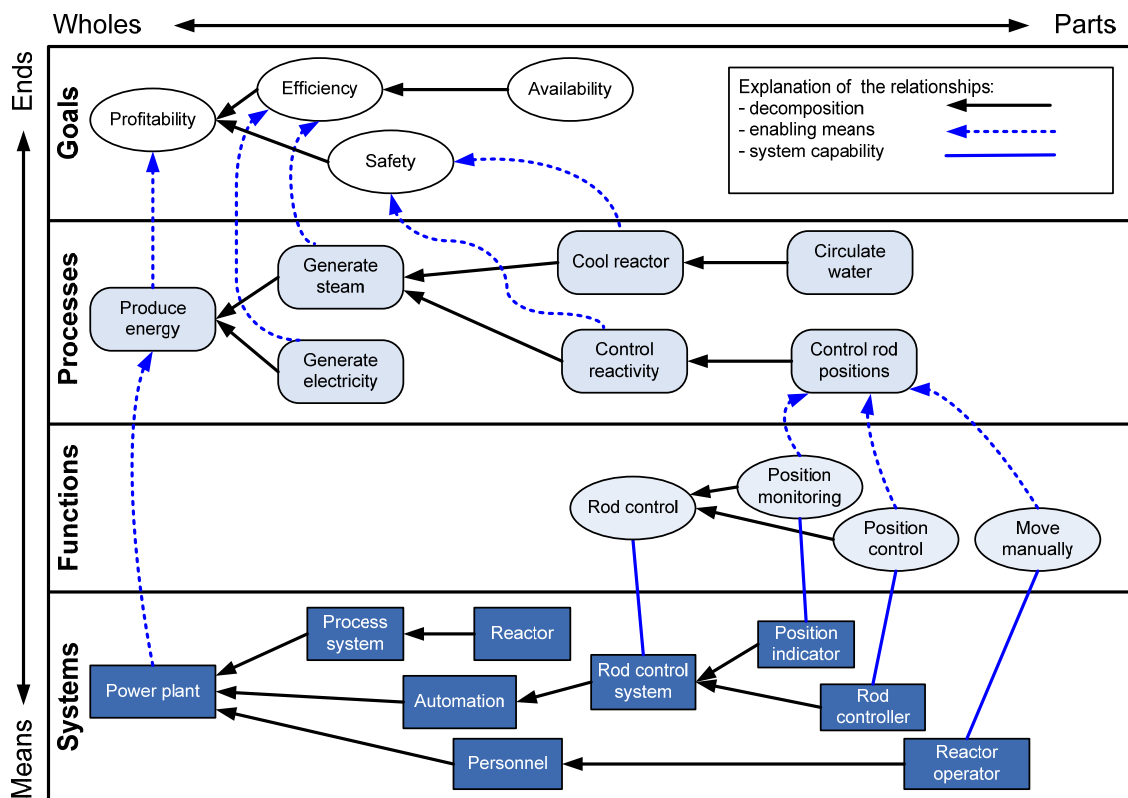


Figure 38. Summary of concepts with a few examples from the NPP domain.

Figure 38 summarises the discussion above with a few examples taken from a nuclear power plant. This figure has been freely adapted from the well-known but also argued abstraction hierarchy (also called a means-ends hierarchy) proposed by Jens Rasmussen in the 80'ies (Lind 1999, Vicente 1999, Lintern 2006). In our modified figure, goals, processes (activities), functions and systems are arranged vertically as means-ends hierarchy. Elements at a lower level are the means to achieve the ends at a higher level. In order to be not confused by goals and functions, we interpret “goal” as a desired but not mandatory state-of-affairs, for example “efficient power plant”. A goal may have “subgoals”, such as “good safety culture” and “reliable plant systems”. Here we have made a distinction between processes and functions by thinking that processes are unallocated “plant functions” while functions are capabilities of individual plant systems used

to accomplish the processes. The horizontal dimension is used to show the hierarchical decomposition of the elements separately at each level of the means-ends hierarchy. For example, the ultimate goal of a plant, to make profit, comprises two major “subgoals”, efficiency and safety. Control of reactivity is one process needed to support the achievement of the safety goal. It includes control of rod positions as one of its “subprocesses”. This “subprocess” is carried out with the help of the tasks (“move manually”) performed by the reactor operator and the functions designed into the rod control system (“position monitoring” and “position control”).

7.3 On safety aspects in NPP system modelling

In nuclear power plants safety is a major issue even if the financial interests of the owners should also be considered. The sections above discussed the terminology of nuclear power plants in general. Here, we make some comments on safety requirements and safety assessment.

Today, it seems that the focus of the debate is on safety regulations while other stakeholder interests and requirements are treated separately (Raatikainen et al. 2011). In general, safety is understood in its broadest sense as the absence of danger to humans, the environment and assets. In nuclear power plants, the main interest is directed to *nuclear safety*, i.e. on the protection of people and the environment against radiation risks (IAEA 2007). The viewpoint has an impact on the terminology. For example, the IAEA safety glossary (2007) gives the following definition of the term requirement: “Required by (national or international) law or regulations or by IAEA Safety Fundamentals or Safety requirements. The more general sense of something that is necessary should be expressed using other words.”

Different interests and backgrounds of various stakeholders and engineering disciplines may lead to inconsistent terminology and confusion. A partial remedy could be to always clearly indicate the viewpoint taken, for example by associating a namespace with each concept used as is often done in computer programming. The namespaces refer to the definition of the terms and make their interpretation unambiguous. A more practical way is to use compound phrases for various viewpoints. In some IEC/SC45A documents, for example, the following types of requirement are distinguished (IEC 61513 2011):

- Safety requirements – Requirements imposed by authorities on the safety of the NPP in terms of impact³ on individuals, society and environment during the NPP lifecycle.
- Operational requirements – Requirements on the operational capacity and ability of the plant imposed by the owner.

In addition to clarifying the terminology, safety-relevant information should be maintained and traced in the system model from requirements to implementation. Some special concepts, properties and relationships are needed to do this. **Figure 39** illustrates some of the basic concepts needed for modelling safety aspects.

³ Obviously caused by radioactive radiation. Note that these definitions don't clearly say what the primary basis for the classification is, the safety relevance or the stakeholder imposing the requirement.

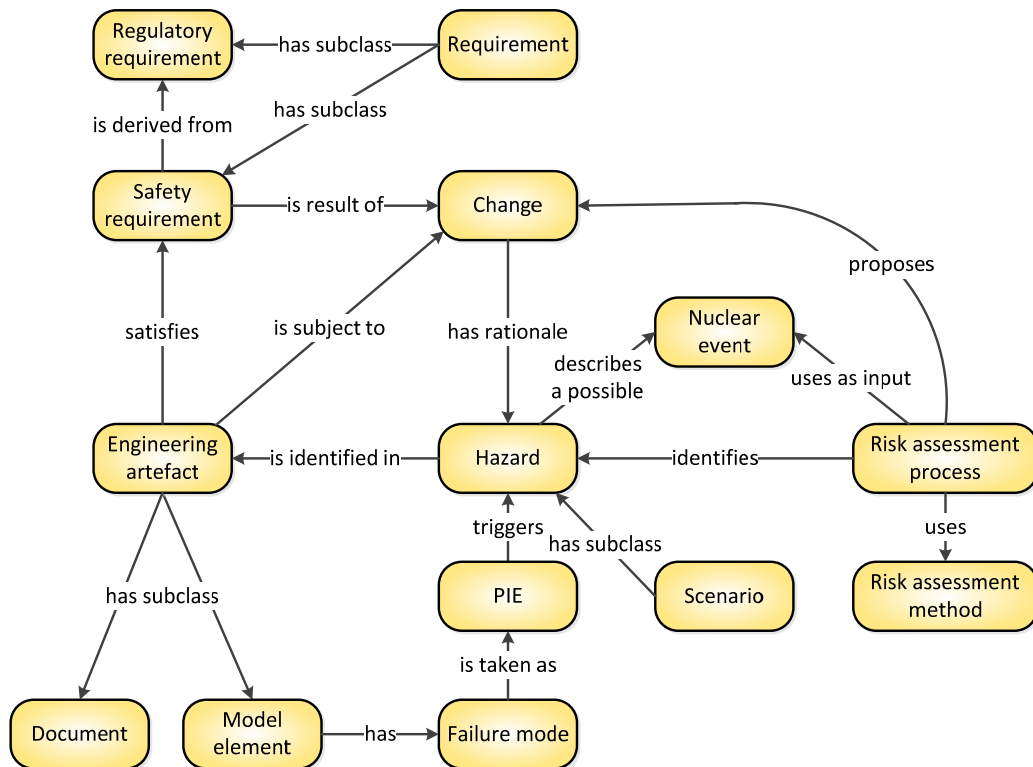


Figure 39. Examples of basic safety concepts for modelling NPP safety requirements.

In this publication we understand the term *safety requirement* as any *requirement* that is sufficiently significant for safety. Safety requirements can be derived from regulatory requirements and risks identified in a proposed design. *Hazard* is generally understood as an “event having the potential to cause damage to plant personnel, components, equipment or structures” (IEC 61513 2011). Here, a *nuclear event*, such as the events reported in the INES database, is understood as an episode rather than as an instantaneous change in system state. A nuclear event can be a “single event” (failure of a device) or a complex accident *scenario* consisting of single events and conditions (IAEA 2007). As discussed in Section 3.4.5, we suggest, however, to understand events as occurrences with zero duration, at least when the design of I&C systems is concerned. Consequently, hazard is interpreted as an unwanted *scenario* describing a possible nuclear event. A *postulated initiating event* (PIE) is an event that results from a *failure mode* associated with a model element and triggers a hazardous scenario such as an anticipated operational occurrence or accident.

A hazard is associated with one or more documents and model elements. Known *failure modes* of various system elements and functions can be used to identify potential sources of problems (Authén et al. 2014). Hazards are identified and analysed in the risk assessment process going on in parallel with design activities. The risk associated with the hazard is analysed with deterministic and probabilistic methods in terms of the severity of the potential consequences and their probability. In order to reduce the risk to an acceptable level some mitigation measures are suggested. This can lead, for example, to changes in existing artefacts or to adding new safety requirements concerning plant systems, as well as their design process. In I&C systems, an identified initiating event often leads to the addition of a safety function to prevent the occurrence of the event.

The risk of hazards and the importance of requirements must be determined and allocated to their solutions during the design process. Various importance measures are used in different application domains and standards. As a measure of safety-relevance, IEC 61508-1 (2010) uses safety integrity levels (SIL). IEC 61513 (2011) defines *categories of I&C functions* and *classes of I&C systems*. It also establishes a relation between the category of the function and the minimal required class for the associated systems and equipment. An I&C function classified into one of the three categories A, B and C obviously becomes a “safety function of an I&C system important to safety” in the sense of IAEA (2007).

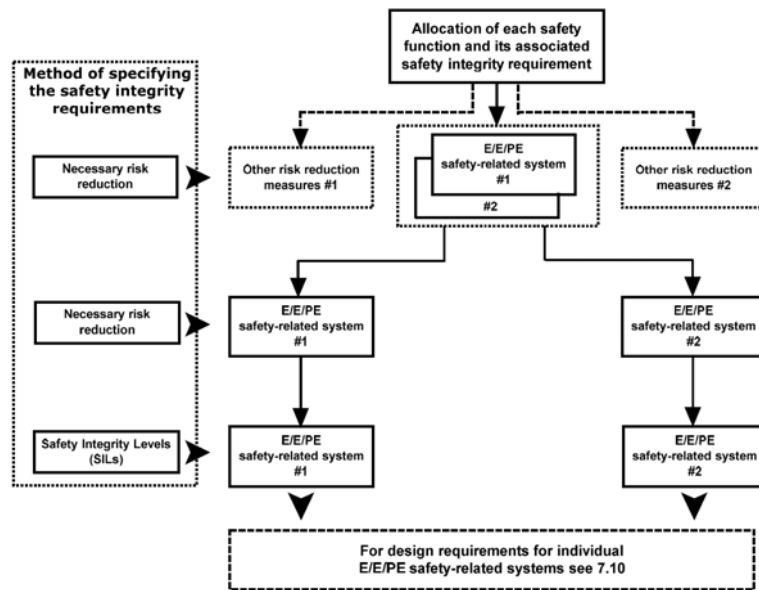


Figure 40. Allocation of overall safety requirements to two safety-related systems (IEC 61508-1 2010).

According to IEC 61508-1 (2010) risk analysis leads to a set of “overall safety functions” needed for achieving or maintaining a safe state of the “equipment under control” (EUC). This constitutes the specification for the overall safety functions requirements. Each overall safety function has an associated “overall safety integrity requirement” specified in terms of the risk reduction required or the tolerable hazardous event rate. In the next step, each overall safety function is iteratively allocated to one or more “safety-related systems” or and other “risk reduction measures” (**Figure 40**). Allocation to several subsystems (see Section 3.4.2) means that each of them takes care of a certain part of the overall safety function and the required risk reduction. As a result we can conclude that each subsystem has to implement a set of system-specific safety functions that together implement the *fundamental safety functions* (see IAEA 2005) at the plant level. As discussed in Chapter 3, a system model can first include a “required safety function” which will later be accompanied by a “specified safety function”.

7.4 Modelling the I&C system life-cycle

For systematic requirements engineering, we need to understand the design process of an I&C system. A life-cycle model is defined in almost all relevant standards and guidelines. However, a sound and widely accepted version is hard to find. The purpose of this chapter is to outline a

high-level approximation of the systems engineering process of I&C systems. The generic concepts discussed in Chapter 4 are used as a basis.

Construction of a large and safety-critical system, such as a nuclear power plant, requires successful implementation of all the processes in several organisations. As illustrated in **Figure 41**, the primary safety requirements and acceptance criteria come from the regulator. The task of the license applicant and its partners is to interpret these requirements in the particular context, to develop the solutions and to demonstrate their acceptability to the regulator in the licensing process. The contacts to the authority should start as early as possible, and a licensing plan should be developed to guide the implementation of the process (see Tommila et al. 2014 for a literature review). In the middle of the figure, the technical engineering process is shown as a “V-Model”, a widely used and illustrative but vague depiction of the development process. A more detailed interpretation of the V-model is given below in **Figure 43**.

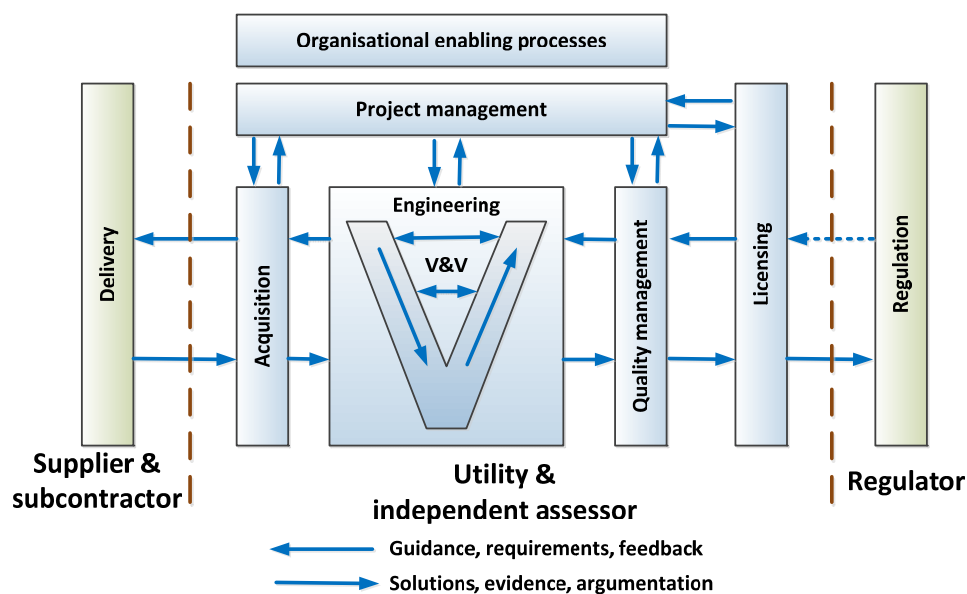


Figure 41. Main life-cycle processes of a safety-critical system.

Engineering design proceeds usually in an incremental and iterative fashion. Therefore, various *activities* can be distributed over the whole project time-line. For example, *requirements definition* is most intensive during the early stages of design but must be continued later also. Some activities, such as testing, can be performed several times. This means that activities are overlapping in time. Successful project management and decision making, however, require that the life-cycle is divided into subsequent *life-cycle phases* separated by *milestones*. At each milestone design artefacts should have matured sufficiently for rational decisions to be made. Each system, subsystem and specialty engineering activity (e.g. risk assessment) can have its own milestones, some of which should coincide with the major milestones at the project level.

In industrial automation, projects are typically divided into preliminary design focusing on requirements, basic design focusing on functions and detailed design producing the software and hardware implementation. After a successful *Factory Acceptance Test (FAT)* the system is delivered to the customer’s site for commissioning. Requirements for the development of safety-related systems are given in IEC 61508.

In the nuclear domain, IEC 61513 (2011) sets out requirements applicable to I&C systems that perform functions important to safety. IEC 61513 has adopted a presentation format similar to the basic safety publication IEC 61508 with an overall safety life-cycle framework and a system life-cycle framework. It provides an interpretation of the general requirements of IEC 61508 for the nuclear application sector and describes the activities of a typical overall safety life-cycle of an I&C system as shown in **Figure 42**. As a foundation, the review of the plant safety design base collects functional, performance and independence requirements, functional categorisation and constraints from the plant context. This results in the definition of the overall requirements specification of the *I&C functions*, systems and equipment important to safety. The I&C architectural design defines the overall *I&C architecture*, i.e. the arrangement of all necessary *I&C systems*, some of which may not be safety-relevant. The *I&C functions* representing the functional requirements originating from the design base are then assigned to individual systems and equipment. The realisation of individual systems takes place according to a safety life-cycle including activities like requirements specification, selection and analysis of pre-existing equipment, system specification, design and implementation, and finally integration, validation and installation.

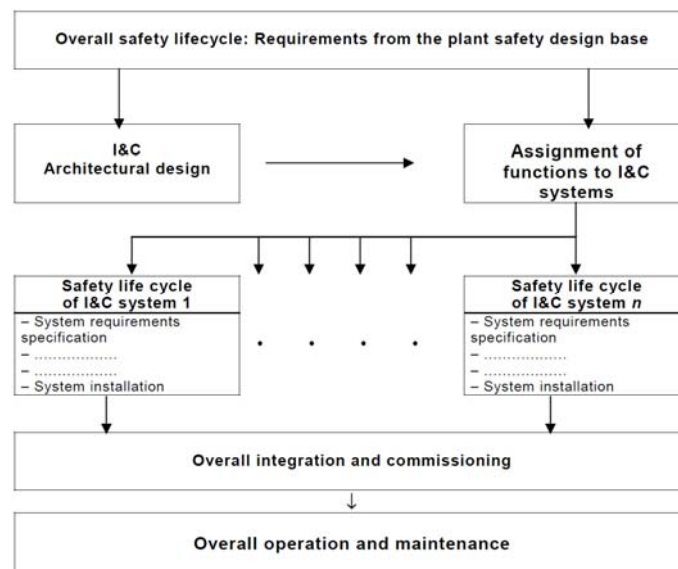


Figure 42. Connections between the overall safety life cycle of the I&C and the safety life cycles of the individual I&C systems (IEC 61513 2011).

The actual life-cycle of a system depends on the particular application, industrial domain and standards and regulations applied. So, an exact model for I&C can't be given. **Figure 43** gives one suggestion and shows how the design contents and activities can be approximately mapped to a set of generic *life-cycle phases*. By freely combining several standards and domains we can identify the following design *activities*:

- Mission and scope: The concept phase begins with initial recognition of a need for a new system-of-interest or for the modification to an existing system. This is an initial exploration, fact finding, and planning period, when economic, technical, safety and regulatory bases are assessed (adapted from ISO/IEC TR 24748-1).
- Conceptual design: The I&C system is seen in the context of its *enclosing system*, for example as a part of a process plant or a process section and operating organisation. The

system is modelled as a black-box without unnecessary internal structure. Its main elements are, however, described if they are already known to be there. So, there is nothing but a *concept* of the system or possibly several *I&C systems* forming an *I&C architecture*. The role of the I&C system as one element of the enclosing system is described in the *Concept of Operations* (ConOps, see Tommila et al. 2013) that is a basis for elicitation of stakeholders' needs. The overall Defense in Depth (DiD) concept is related to ConOps but with the focus on safety issues. The design bases, regulatory requirements and identified hazards are the principal starting point for safety requirements.

- Requirements definition: The *stakeholder requirements* identified in the conceptual design are allocated to the system and its possibly known elements and further decomposed/formulated as technical and verifiable *system requirements*. I&C requirements are derived from the plant safety design base (IEC 61513 2011) regulatory requirements and risk assessment. The initial system concept, sufficiently detailed requirements, technical constraints and cost estimates allow an investment decision to be made by the plant owner.
- Architectural design: Structure is added to the system model. This includes functional architecture, information architecture and implementation architecture covering software and hardware (see ISO/IEC/IEEE 42010 2011). Supposing that requirements above focus on the stakeholders' interests, the architecture is often in the supplier's answer to the requirements expressed in the invitation to tender. In principle, a *functional specification* written by the supplier should focus on detailing the system's application functions. However, implementation aspects are also present. When an agreement is achieved, a contract can be signed between the customer and the supplier. And, when sufficient information concerning the system and the project organisation is available, for example in the form of a conceptual design plan or a preliminary safety analysis report and a quality plan, the utility can apply for a construction licence.
- Detailed design and implementation: Detailed software and hardware specifications are produced, and the system components are manufactured and programmed according to them. Modules are tested individually and, as far as possible, integrated at the supplier's premises for the *Factory Acceptance Tests*. When the tests are successfully passed, the system can be transferred to the customer's site for installation.
- Installation and commissioning: Following delivery, receiving inspection and integration of the I&C systems on site, their configuration, functions and performance are verified against the specifications. In the pre-commissioning phase, non-operating adjustments, cold alignment checks, cleaning, and testing of machinery take place (IEC 62337 2011). Next, the operation of equipment and facilities is tested, first with safe materials (cold commissioning) and finally with actual process materials (hot commissioning).
- Acceptance: After commissioning is finished and the system has passed all tests, there is some paper work left. The plant owner needs to combine all the evidence collected during the whole project and demonstrate that the stated requirements are satisfied. When concerning the regulatory requirements set by the society, we use here the term *qualification*, which results in an operation license if successful. However, there are also other, operational requirements that the system must be validated against, for example the ones concerning the interests of the plant owner and the needs of control room operators.
- Operation and maintenance: The system is operated and maintained according the policies and procedures defined in the concept and development phases. The utilisation phase can start after the operation licence is granted by the regulator. During this period an I&C sys-

tem can experience several modifications and upgrades each of them repeating the life-cycle phases and activities in a smaller scale.

- **Decommissioning:** A control system can come to the end of its life when the whole facility is decommissioned or when the system is fully replaced. Decommissioning can be divided into preparatory and implementation phases. Preparations include the development of a decommissioning plan. Consideration of decommissioning shall begin early in the design stage and continue to the release of the facility from regulatory control (IAEA 2006).

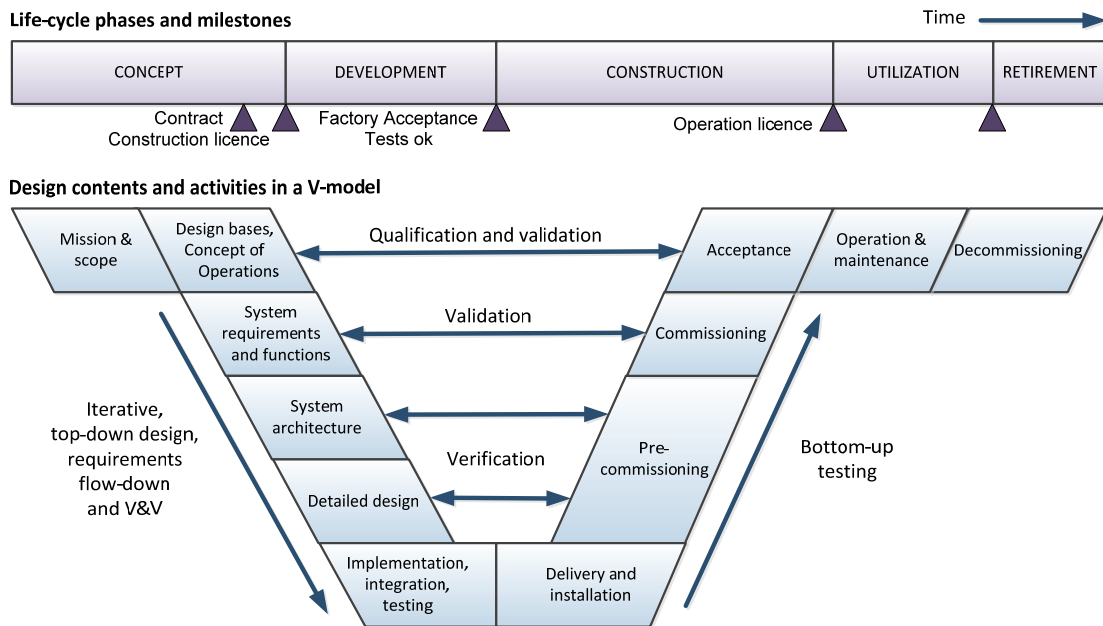


Figure 43. Two views to system life-cycle – consecutive phases and corresponding design activities shown in a V-model.

IEC 61513 (2011) highlights the need for complete and precise requirements derived from the plant safety goals as a pre-requisite for requirements of the I&C architecture and individual I&C systems. It places requirements upon the development of the overall plans to ensure that the requirements of I&C functions important to safety distributed over the I&C systems will be achieved and maintained throughout the life of the systems.

In the quality assurance of critical applications, *verification* and *validation* are particularly important engineering activities. The meaning of the terms is, however, not fully unambiguous and depends on the background and interests of the speaker. **Figure 44** tries to clarify our interpretations. The aim of *verification* is to confirm that the results of a life-cycle phase satisfy the requirements and specifications originating from the previous phase. For example design specifications are compared to requirements or system implementation to its specifications. *Requirements verification*, in particular, tries to show that the requirements are consistent and correctly derived from standards and higher-level requirements.

Validation, in turn, is an activity to confirm that artefacts, resulting from any life-cycle phase, describe a system that is “good” for the intended purpose and users. Usually artefacts being validated are compared to the original user requirements. This works well if domain knowledge and regulations are correctly reflected in the requirements. However, answering the question about the system being “good” or not may require also guidelines or other domain knowledge

available to the (independent) evaluator, to be used. According to this reasoning, *requirements validation* aims to ensure that the requirements describe a system that, if correctly implemented, would satisfy its users' stated or implied needs and relevant regulations.

An exact meaning for the term *qualification* is hard to find in the literature. In some definitions it is similar to verification (e.g. design qualification), in some others close to validation. Usually, the term is associated with people (e.g. qualified NPP operator) or pre-existing products (e.g. suitability analysis). It often has the nuance of anticipating whether people or equipment will successfully meet the requirements under specified conditions (e.g. in IAEA 2007), while verification and validation look back to see if design has been successful. From a purely regulatory viewpoint qualification becomes comparing artefacts to official acceptance criteria like standards and regulations as indicated in the lower part of **Figure 44**.

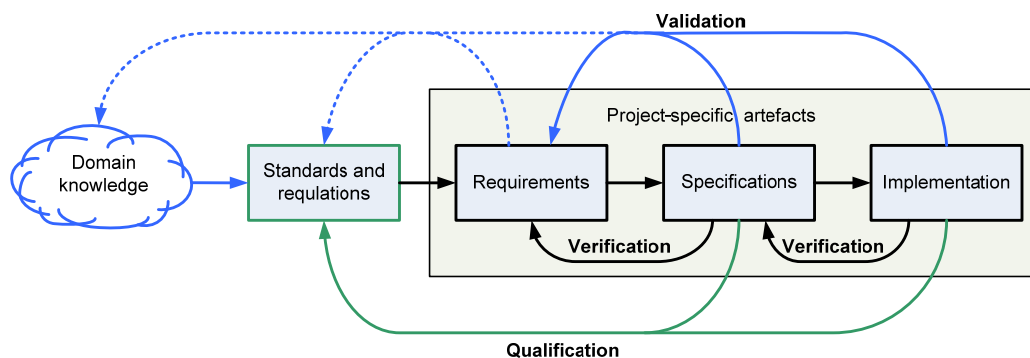


Figure 44. Verification, validation and qualification activities.

A quality assurance program or an integrated management system shall be planned and implemented within the NPP project. The *quality plan* shall present, among other things (YVL B.1 2013):

- the organisations designing the system with their responsibilities and interfaces
- the standards and guidelines applied in the design and implementation
- the stages of the design and implementation process;
- the documents, records and other data used and produced in each design stage;
- the reviews upon completion of individual stages
- the procedures used in the supervision of subcontractors;
- configuration and change management and procedures, management of non-conformities;
- the support processes utilised concurrently with design and implementation

So, the quality plan developed for a specific project or for wider use in an organisation seems to be a place for documenting the life-cycle phases, development processes, engineering artefacts and resources used. An aspect that must also be emphasised in NPP projects is the licensing process that requires its specific milestones and documentation, as well as continuous communication with the regulator and tracking of non-conformities and other open issues. Today engineering processes are company-specific. Also a variety of licensing and authorisation procedures exists in different countries (Tommila et al. 2014). So, a shared reference model would be needed both for the systems engineering processes within the utility and its contractors and for the licensing of safety-related I&C systems.

8. Summary

This Part I of our publication has discussed informally the principles of modelling nuclear power plants. The target has been on requirements definition and management and especially on the safety of instrumentation and control systems. These issues can, however, not be considered in isolation from other kinds of design data or systems at a nuclear power plant. In a similar fashion, requirements engineering processes are tightly coupled to other life-cycle activities. So, we have taken a broader scope and described power plant functions, systems, structures and their properties, let them be required the various stakeholders, specified by the developers or actually present at an existing power plant.

One aim has been to clarify the terminology in a way that supports communication and documentation in the current design practice. Unfortunately, the terminology found in existing standards and guidelines does not provide easy solutions. The definitions are often vague, and there are inconsistencies between different sources. Potential solutions can be sought for in recent developments in model-based system development, semantic web technologies, integrated engineering environments and international standardisation of product data models. The long-term vision is, however, in computer-supported engineering and exchange of design data. This has had implications to our work. The terms defined are understood as elements of a system model and a project model stored in an electronic repository. All these model elements express statements originating from various stakeholders, including designers, and have the purpose of communicating relevant information between them throughout the system life-cycle. Requirements are statements having a communicative intent different from other kinds of statements like user's needs, known facts, assumptions or organisation's intentions. We have tried to clarify their interpretations and ways to express the unambiguously, either in textual or graphical form. This further underlines the need to consider requirements as an integral part of system models and engineering processes.

For future computer tools to work, the concepts used for system modelling should be consistent and based on a sound philosophy. Laying these foundations has been the goal of this Part I of our publication. Part II takes a more technical viewpoint and defines an information model that has a more limited scope but that can be used as a basis for practical software tools.

PART II – IMPLEMENTABLE ARTEFACTS MODELS

Part II of this report presents some implementable information models for systems engineering artefacts. The focus is on I&C systems, but other disciplines, such as mechanical systems can be well accommodated, maybe with some customisations. The target is that the artefact models provided in this part can be implemented by a tool that supports structured storage of data. In practice it means database oriented tools or tools that store data in XML format. To implement the models, one has to adapt the model and define the attributes of the artefact types according to his/her needs and the capabilities of the selected tool. It is not necessary to implement all the artefact types provided in the models.

The artefacts models provided in this part mainly follow the principles discussed in Part I. However, as the artefact models are meant to be more practical, some simplifications have been made. Moreover, since this part is based on the models presented in (Alanen et al. 2011), the original focus of those models, i.e. machine automation, is reflected in some cases, especially in the risk assessment model. Nevertheless, a nuclear power plant includes machines, such as hoisting and transfer equipment of nuclear fuel, for which the machinery sector concepts and risk assessment models are relevant.

Contents

PART II – IMPLEMENTABLE ARTEFACTS MODELS	68
9. Engineering artefacts models	70
9.1 Introduction	70
9.2 Project data repository core model	71
9.3 System package.....	76
9.3.1 System artefacts model	77
9.3.2 Physical architecture package.....	77
9.3.3 Behaviour package	81
9.3.4 Properties package.....	84
9.3.5 Product type package	85
9.4 System context package.....	86
9.4.1 System context artefacts model	86
9.4.2 Operational environment model	87
9.5 Requirements and V&V package.....	87
9.6 Specialty engineering package.....	89
9.6.1 Risk assessment package.....	90
9.7 Projects package.....	93
10. Traceability information model.....	95
10.1 Traceability demonstration.....	97
11. Summary and conclusions	113
References	115
Appendix A: Acronyms and abbreviations.....	121
Appendix B: Glossary	123
Appendix C: Artefact types	135

9. Engineering artefacts models

9.1 Introduction

The engineering work is normally carried out within projects managed by a project organisation. The engineered system has a context, natural or engineered. In a project, the managers and engineers create various engineering artefacts, e.g. model elements and documents, related to the project, the system and its context. Examples of project artefacts are project plans, task descriptions and change orders; examples of system artefacts are system requirements, system elements and system functions; examples of system context artefacts are life cycle model, terminology, environmental specifications and external systems. The project artefacts constitute the project model, the system context artefacts constitute the system context model and the system artefacts constitute the system model according to which the system is produced, operated, maintained and finally disposed of (see **Figure 45** and Section 3.2 of Part I).

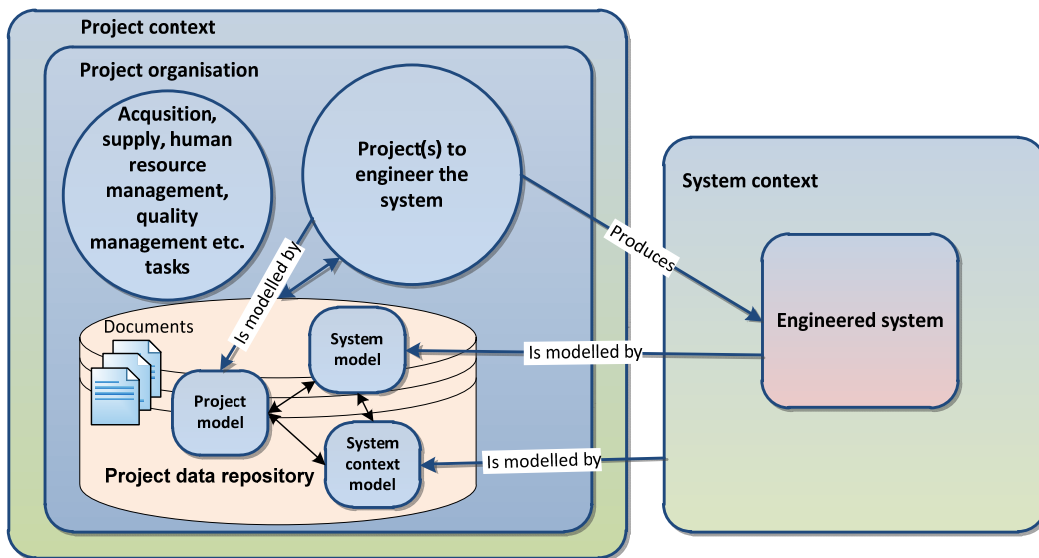


Figure 45. A real-world system is engineered by a project organisation creating various engineering artefacts stored in a project data repository.

Most of the engineering artefacts are electronic data in the form of textual descriptions, diagrams, simulation models etc. as described in Section 3.2 of Part I. Besides electronic data, physical mock-ups and prototypes can be created, but also these are preceded by electronic data. This data are stored in a project data repository. A common implementation of the project

data repository is a file system or a document management system. In this report, the goal is to structure the project data repository such that the granularity of the engineering artefacts is finer than that of document based engineering, and such that the relations between the artefacts are well defined to support traceability of the artefacts. We call the information model Systems Engineering Artefacts Model for Nuclear Power Plants, abbreviated SEAModel_{NPP}. The model is presented as a set of UML class diagrams. **Figure 46** provides a quick guide for interpreting the diagrams.

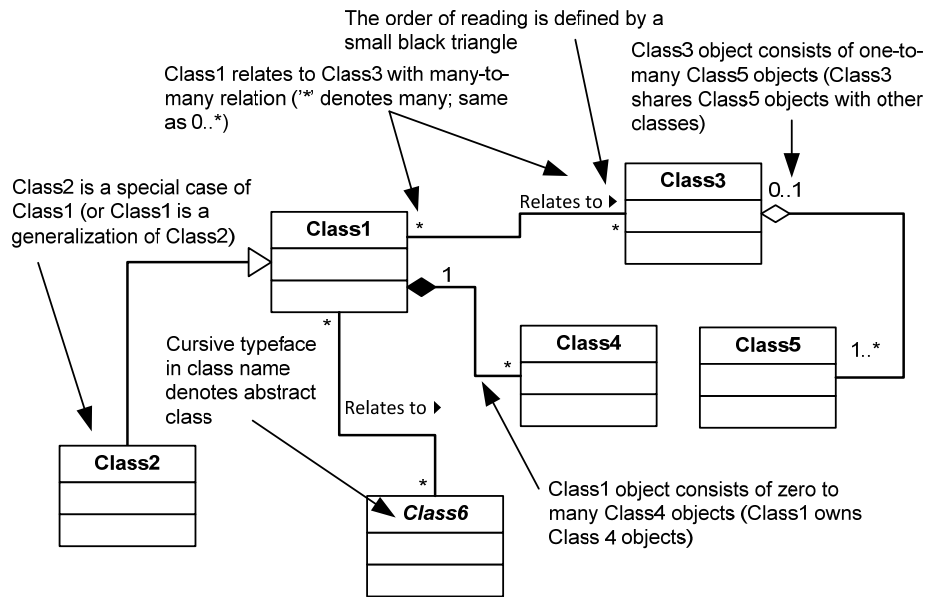


Figure 46. Guide for interpreting the UML class diagrams.

Each class in the UML diagrams presented in the following chapters represents an engineering artefact type, such as a requirement, system function, system element, document or entire system model. Each class, i.e. an artefact type, can be implemented as a database table, an XML data element or any structured data element in a project data repository tool. The specialisations (see Class2 in **Figure 46**) can be implemented in different ways, as a separate artefact type or by using an attribute or a Boolean flag to denote the specialisation type. The abstract classes (such as Class6 in **Figure 46**) need not to be implemented; they are only provided to help understand the models.

9.2 Project data repository core model

As discussed above, this part of our publication suggests an information model that can be used as a source of ideas when tools are developed for designing NPP I&C systems, expectedly on top of commercial software platforms. This information model classifies the pieces of engineering knowledge that populate a project data repository. In other words, we don't speak about a particular control application but define the kinds of knowledge fragments (i.e. artefact types) that can be used to implement a structured data repository with (traceability) relations between

various engineering artefacts. We describe the contents and structure of a data repository that manages artefacts and their relationships according to SEAModel_{NPP} starting from Section 9.3.

As a more formal presentation of **Figure 45** above, the UML diagram in **Figure 47** shows how we see the structure of the engineering data repository. First of all, the repository consists of *engineering artefacts*, all of them being pieces of data authored by designers or generated by their tools. Each engineering artefact is considered to contain at least the following attributes: Identification code (a prefix and an ID number), Name of the artefact, Description, Workflow state, Modification date, Modification person, Version number⁴ and External links. In most of the cases, additional attributes are needed, e.g. the Requirement artefact could own attributes such as Source, Type, Priority and Status. However, attributes of the artefact types are not provided in this report.

We advocate here the model-based design paradigm as the future alternative to current document-oriented practices (see Section 3.2 of Part I). However, documents can't be avoided. So, model elements and black box artefacts are the two main classes of engineering artefacts. Correspondingly, the project data repository shall contain both a structured *system model* and an archive for black box artefacts. The black box artefacts are called here 'black box' due to the fact that the internal structure of such artefacts is not modelled, or if it is, it is not known from the point of view of the project data repository model. The main types of black box artefacts are documents (such as international standards) and foreign models (such as a CAD-model managed by the CAD-tool, and exported to the project data repository as a model file or files).

A system model consists of a large number of model elements and must therefore be managed and shown to various users in suitable ways. The model part of the repository consists of one or more hierarchically organised *models* that are containers for the actual *model elements* and their *relations*. For example, the repository might include one model for the project and a second one containing the system model and the context model. Models can be used to encapsulate related model elements and their relationships into logical units. However, a model can import elements from other models.

Views are used to define, which relevant parts of the models and black box artefacts are combined and shown to the users. Information package is a view that supports provision and exchange of collections of documents and models for specific purposes such as risk assessment and formal approvals. A single document or model can exist in several collections. Version management of the shared artefacts needs to be carefully planned.

⁴ The actual implementation of the version control may be such that the actual version information is in a separate table, and there is a relationship between the engineering artefact and the version info table. Full configuration management with baselines etc. is not elaborated in this report. It is assumed that the implementation platform provides basic version control of the engineering artefacts.

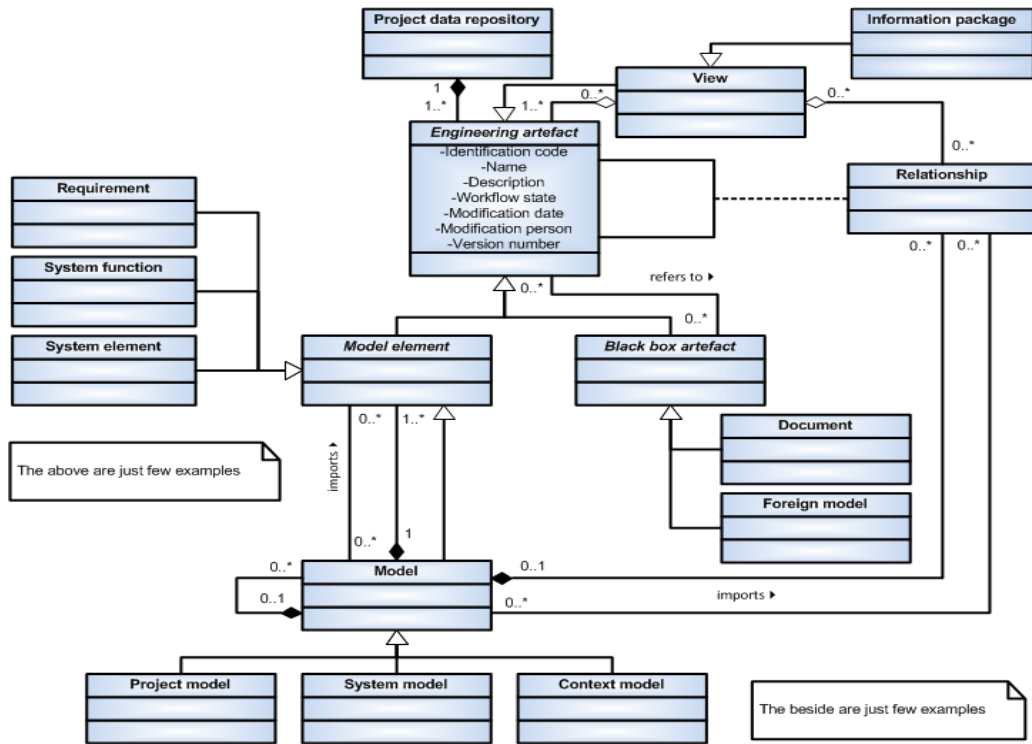


Figure 47. Project data repository core model.

As can be seen in **Figure 47**, any engineering artefact can be linked to a Document or Foreign Model artefact by applying relationship between the two engineering artefacts; for example, a requirement can be linked to a requirements specification document. Such links are not depicted in subsequent models starting from Section 9.3, except in some rare cases, such as the link between a requirement and requirements specification document that is depicted in the Requirements and V&V model in Section 9.5.

It should also be noted that the special case depicted in **Figure 33** in Part I, in which an artefact is a synthesis of two draft artefacts, is also supported by the model in **Figure 47**; the relationship is suggested to be named 'is derived from' in that case. Furthermore, two versions of the same artefact can also be linked with the 'is derived from' relationship, as is depicted in **Figure 33** in Part I. Such need may arise in case where a new version of an artefact is not derived from the previous version, but from an earlier version of the same artefact.

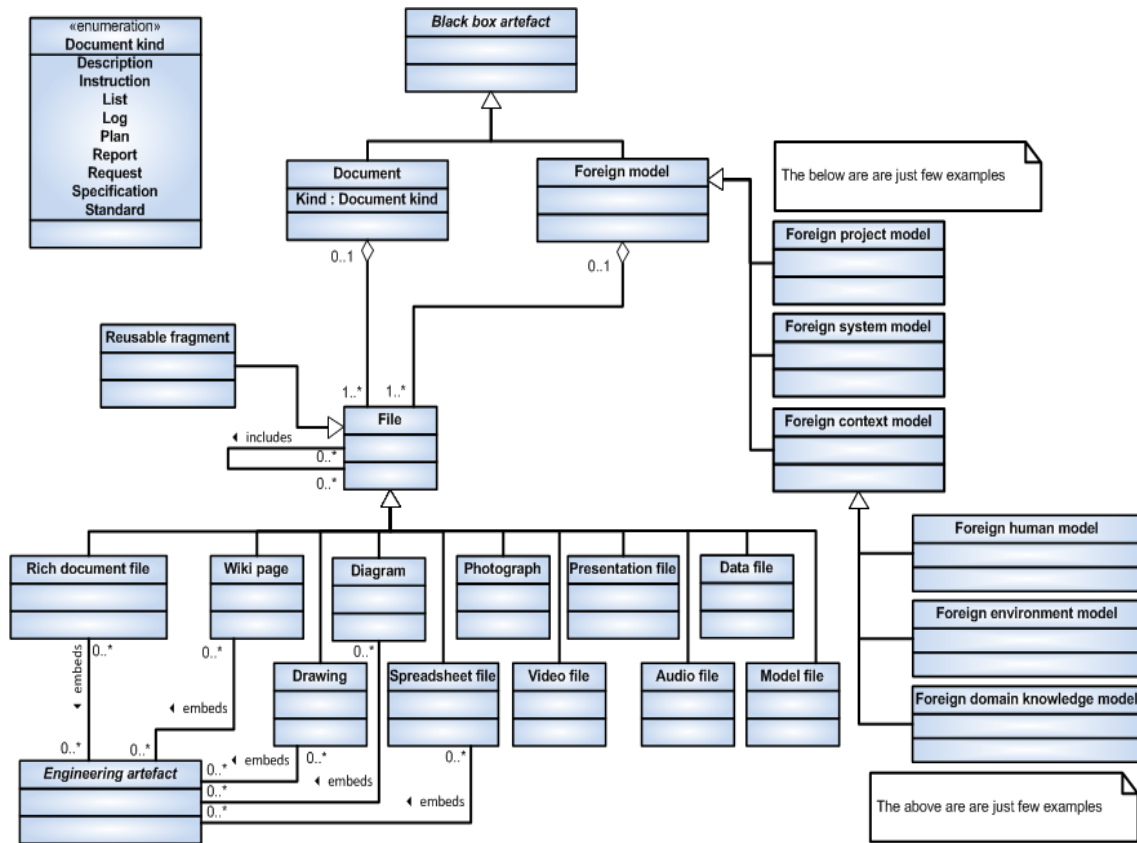


Figure 48. Model for black box artefacts: documents and foreign models.

The black box artefact is elaborated further in Figure 48. The key elements in the artefacts model are the Document artefact type and the Foreign model artefact type. Nevertheless, more black box artefact types can be added if needed.

The Document artefact is considered to be a multi-file document. E.g. a document can include the main word processing file plus several file attachments. Its most natural implementation is a folder, but the problem with a conventional folder structure is that the folders own the files inside them. Very often, it should be possible to share a file with several documents. Hence it is encouraged to consider other types of implementations. However, version management becomes a challenge in the shared files scheme: two documents sharing a file may want to include different versions of the shared file.

A set of Document kinds is suggested in **Figure 48**. These document kinds (except document kind Standard) are specified by IEC 61508-1 (2010)⁵. More advanced guidelines for the classification of documents are given in IEC 61355 (2008)

The File artefact type supports a suitable set of content types, some of which are illustrated in **Figure 48**. The Word processing file, Wiki page, Drawing and Spreadsheet artefact types possess a special property to include artefacts (e.g. requirement objects, system use cases and

⁵ The document kind *Diagram* of IEC 61508-1 is problematic, because it does not refer to the role of the document as the other document kinds (like *Description* and *Specification*) do, but to the document content type. This discrepancy is evident in the following fact: A diagram can specify or describe a system; hence specifications or descriptions can be or contain diagrams. The document kind *Diagram* is therefore omitted. Instead, *Diagram* is introduced as a specialisation of *File*.

other engineering artefacts defined in the subsequent models) within the free-flow text. This facilitates automatic or semi-automatic document generation⁶.

The concept of Reusable fragment supports reuse of pieces of documents, e.g. a piece of reusable text can be included into a word processing file.

A foreign model can include one or more files of any type. The Foreign model artefact can have several special cases; **Figure 48** includes some examples of such Foreign model specialisations.

The rest of this Part II goes deeper into the types of model elements actually used in describing the system of interest and its context. Only three classes, Requirement, System function and System element are shown in **Figure 47** as examples of model elements; more is provided in the following sections. Note here that the model elements can be textual, such as requirement sentences and system function specifications. What makes our approach model-based engineering lies in the fact that the textual artefacts are not stored within a free flow text in a word processing document, but in a structured way in a database or similar data repository. See more about model-based engineering in Section 3.2 of Part I.

To help the reader, SEAModel_{NPP} is packaged as depicted in **Figure 49**. Note that this is to structure the engineering artefact types into chunks of related concepts, not the actual project data repository. The actual implementation of SEAModel_{NPP}, i.e. the project data repository, does not have to follow the package model of Figure 49.

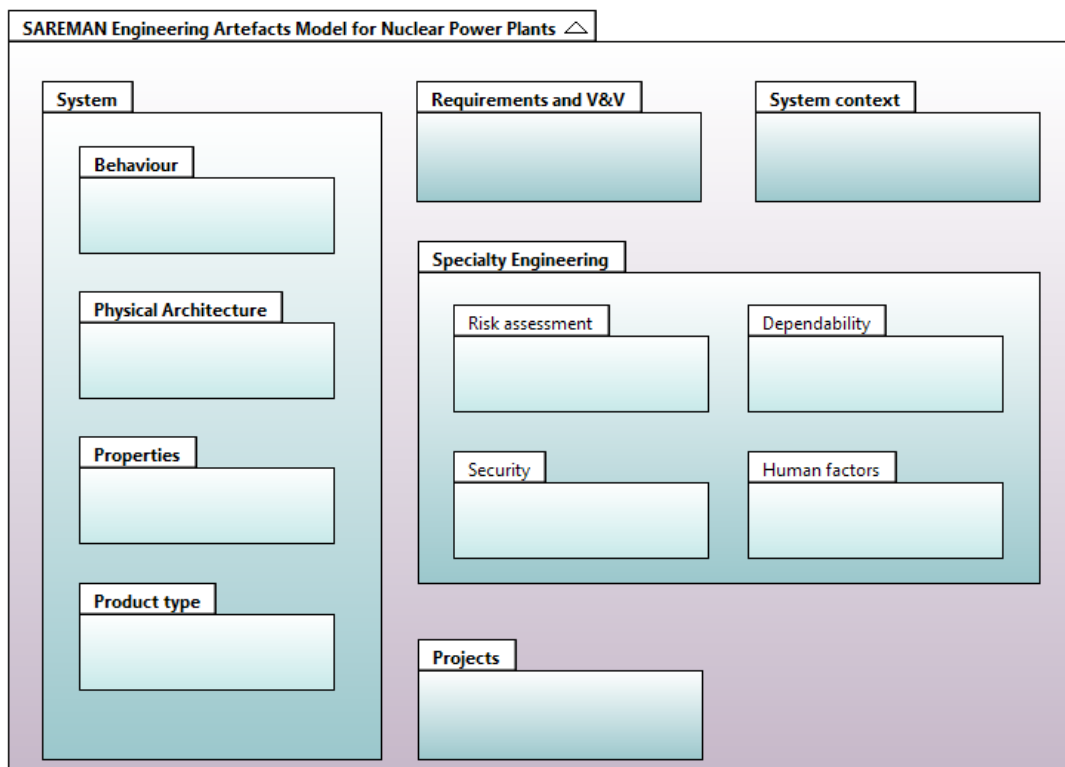


Figure 49. Package structure of SEAModel_{NPP}.

⁶ One could call such generated documents 'grey box artefacts' instead of black box artefacts due to the fact that such documents embed known model elements from the project data repository.

For some of the artefact types, it is not easy or possible to define, which package, project, system or system context, the artefact type belongs to. Furthermore, some artefacts apply to several sections of the project data repository. Examples of such artefacts are requirements, constraints, hazards and verification and validation artefacts. Hence the artefacts are packaged according to the kind of knowledge they represent. This is visualised in Figure 49 by the fact that, besides the three packages mentioned above, two additional packages are defined, namely Requirements and V&V, and Specialty Engineering. The packages in Figure 49 import elements from each other.

The System package includes nested packages for behaviour, physical architecture⁷, properties⁸ and product type. The specialty engineering package includes nested packages for risk assessment, security, dependability and human factors engineering; currently only the Risk assessment package is defined in more detail. The Speciality Engineering package can of course include models from other fields, such as environmental sustainability.

The packages are presented in more detail in the following sections. Each of the artefact types shown in the UML diagrams is explained in Appendix C.

9.3 System package

The purpose of the System package is to define the types of model elements used for building the system model. To better understand the subsequent models, it is necessary to understand the hierarchical structure of the systems presented in Sections 3.1 and 3.3 of Part I. The System package consist of the System artefacts model (Section 9.3.1) and four sub-packages, Physical architecture package (9.3.2), Behaviour package (9.3.3), Properties package (9.3.4) and Product type package (9.3.5).

The System main package includes only one model, the System artefacts model presented in Figure 50.

⁷ In some modelling handbooks and standards, phrase '(physical) structure' is used instead of 'physical architecture'. Here 'physical architecture' is used to distinguish from building structures and functional structure (i.e. functional architecture, which is here included within the Behaviour package).

⁸ Behaviour and physical architecture are also properties of a system, but instead of using a phrase 'other properties' or 'physical etc. properties', we simply use the phrase 'properties'.

9.3.1 System artefacts model

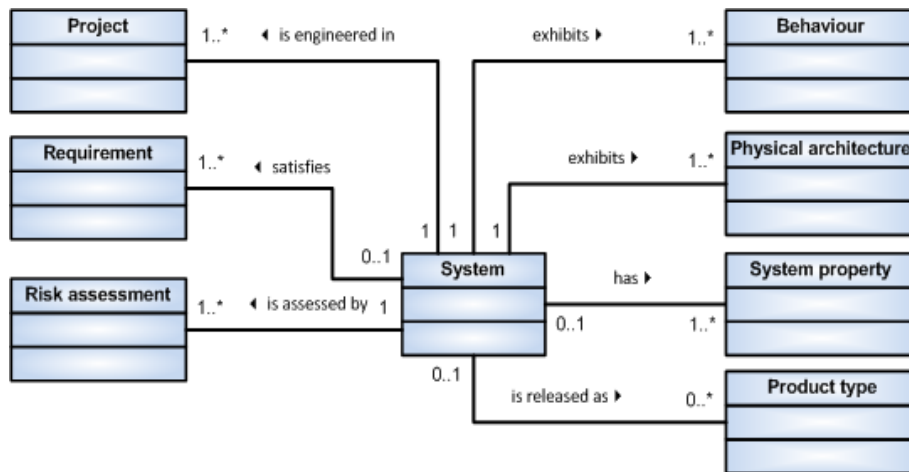


Figure 50. System artefacts model diagram.

The System artefact model (Figure 50) is the main and top level model to define the artefacts relating to the system under study. The main artefact types of the system model (besides the System artefact type) are the Physical architecture, Product type, Behaviour, System property, Requirement, Risk assessment, and the Project artefact types. A separate model for each of these is provided in later sections.

The main contents of the System artefact are the title and a short description of the system (i.e. system identification) to help all developers understand, what the system under development is.

9.3.2 Physical architecture package

The model of the physical architecture is based on the AP233 system structure artefacts model presented in Figure 51.

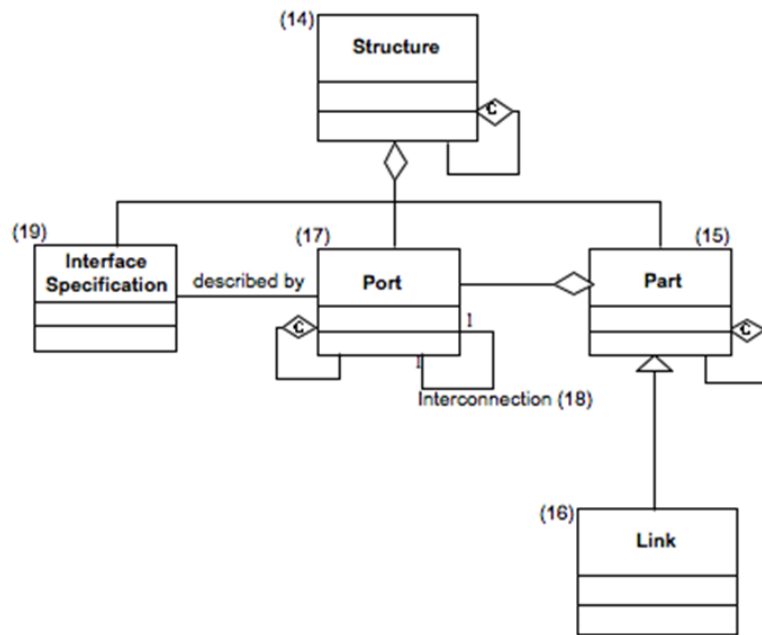


Figure 51. System structure artefacts concept model of AP233 (ISO 10303-2332012⁹).

The Physical architecture sub-package contains the model relating to the physical architecture (i.e. the structure) of the system¹⁰. The main idea in SEAModel_{NPP} is that the physical architecture is defined by its System elements (Parts in the AP233 model above) and their interfaces. The Physical architecture package currently consists of only one model, the Physical architecture artefacts model, but it is divided into two diagrams, the system decomposition artefacts diagram in **Figure 52** and the system interfaces artefacts diagram in **Figure 53**.

The model in **Figure 6** in Part I for hierarchical decomposition of a system into sub-systems and atomic components is re-illustrated in a semi-formal manner in **Figure 52**.

⁹ The figure is from the 2009 draft version of ISO 10303-233.

¹⁰ In literature, physical architecture is often denoted 'structure'. We do not use the word 'structure' here to avoid confusion with building structures. Hence, in **Figure 51**, the Structure artefact corresponds to the Physical architecture artefact of SEAModel_{NPP}. The term 'physical architecture', however, has a drawback that it may lead the reader to regard physical architecture elements as hardware elements, which is not a correct conclusion; physical architecture elements can be, for example, software elements, information items and persons.

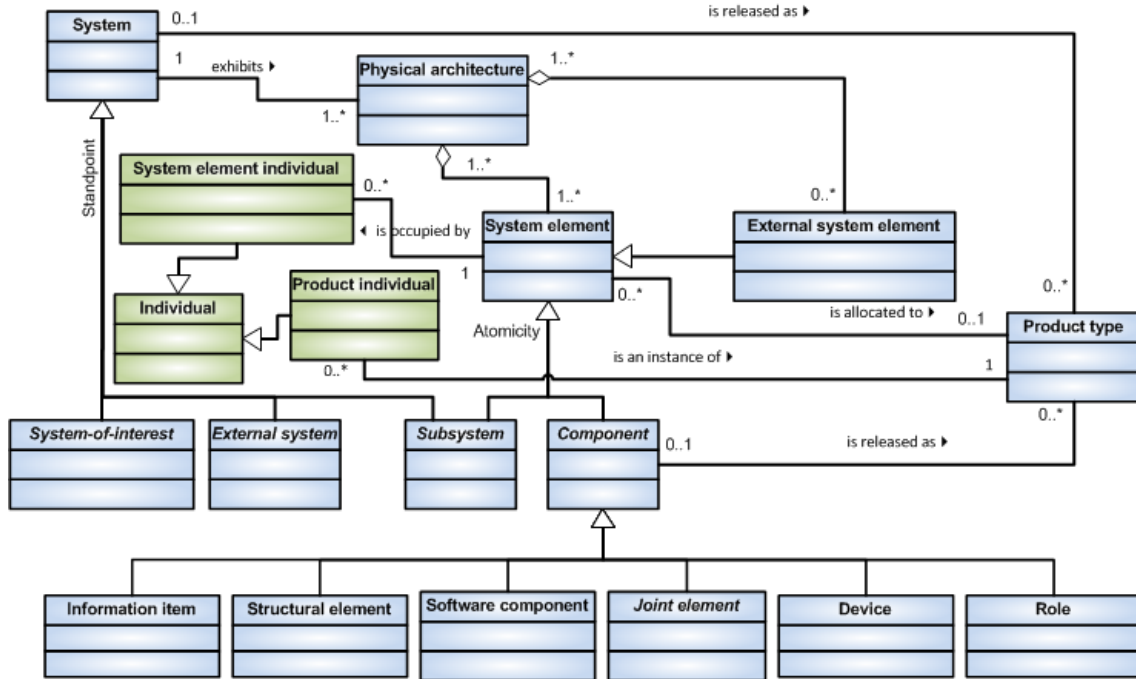


Figure 52. System decomposition artefacts diagram.

The hierarchical decomposition diagram depicts the physical system composition: Physical architecture consists of system elements that can be atomic, i.e. components, or non-atomic, i.e. subsystems. Subsystem and a System-of-interest are both specialisations of System. Furthermore, a subsystem is a system-of-interest from the point of view of the subsystem provider. Besides the system-of-interest and its subsystems and components, there normally are one or more external systems the system-of-interest interacts with. Now again, an external system is a system-of-interest from the point of view of its provider.

Any system, and hence a sub-system or an external system, can be discipline specific, e.g. I&C system or a mechanical system. In other words, the model does not exclude mechanical, hydraulic, etc. systems although the focus in this report is in I&C systems. This applies to all the models presented in this chapter (Chapter 9). However, some special artefact types might be needed to be added to the models presented in this chapter if comprehensive coverage of mechanical, hydraulic etc. systems is required.

Each system element of a system is allocated to a product type. A product type can be a system or a component type. When the supplier of a subsystem releases his system(-of-interest) as a product type, the engineer of the (main) system-of-interest can incorporate the particular subsystem via the Product type artefact into his system-of-interest to implement a particular system element.

System elements that belong to the system-of-interest are distinguished from the system elements of an external system. The external systems can have interfaces (i.e. ports) that connect to the system-of-interest. This fact is depicted later in the system interfaces diagram (Figure 53).

The atomic components can be devices (such as instruments), joint components (such as cables or mechanical links), software components (such as function blocks), structural elements (such as cabinets), person roles (such as operators) or information items (such as messages on a fieldbus).

The artefact types (i.e. the classes in **Figure 52**) the titles of which are written in *Italics* typeface are abstract, i.e. it is assumed that in the implementation platform of SEAModel_{NPP}, such artefact types do not have any dedicated placeholder. Such artefact types are System-of-interest, Subsystem, Component and External system. The reason is that System-of-interest, Subsystem and External system depend on the point of view; i.e. in the implementation platform, it cannot be fixedly defined whether a system is a system-of-interest, subsystem or an external system, because the system can be any of these depending on the viewpoint; furthermore, Subsystem and Component are manifested as System element or System and thus do not need a dedicated placeholder in the data repository implementation.

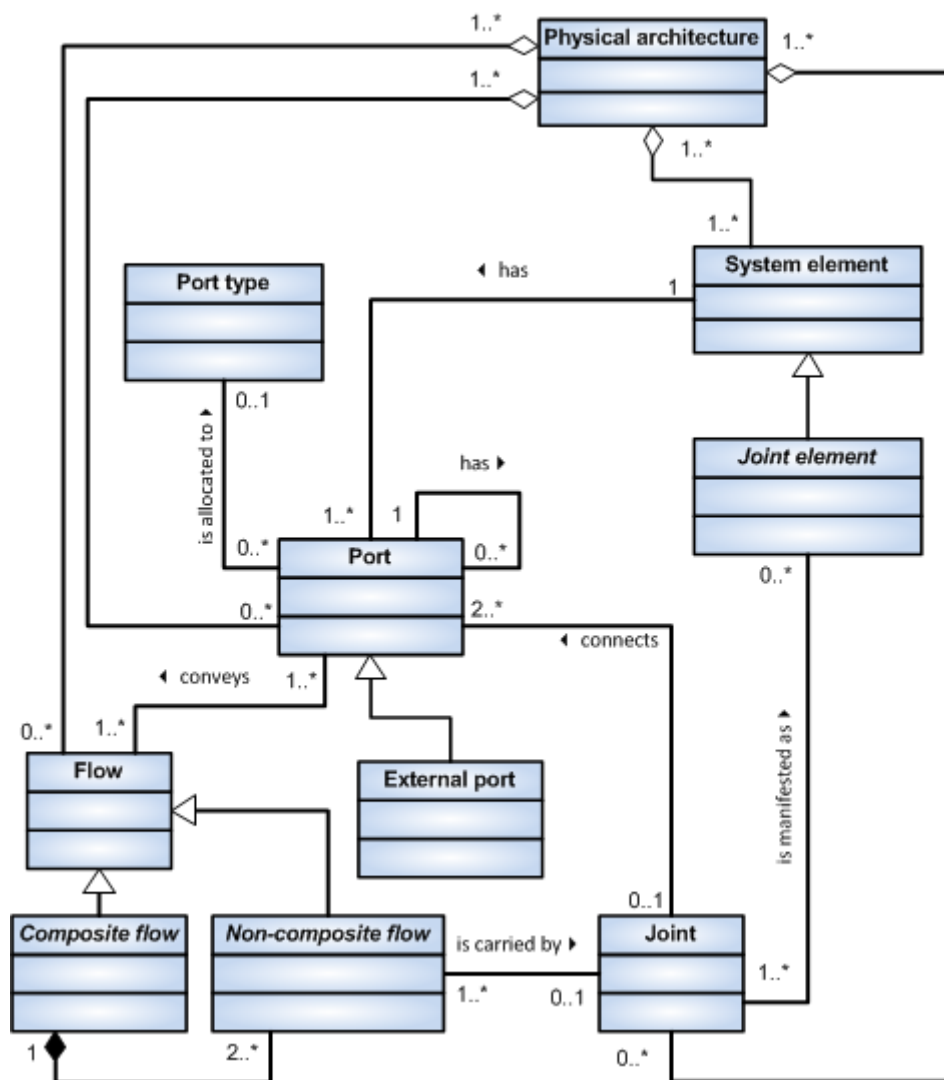


Figure 53. System interfaces artefacts diagram.

The System interfaces diagram is supplied in Figure 53. The interfaces are modelled by Ports specified by Flows that flow through the ports. Port has one or more sub-ports, e.g. a connector port can have several electrical pin ports. Ports can be connected together with Joint. A joint is a logical entity that may manifest itself in the architecture model as Joint components (specialisations of System element), e.g. as a mechanical rod or as a cable assembly with cable splices

that are defined during the electrical CAD work. There may be cases in which a joint is not manifested as a joint component, such as in case of software elements.

In principle, the association from Physical architecture to Port seems unnecessary, because a Physical architecture artefact consists of all the Port artefacts under the System element artefacts anyway. In the model, however, a port, the joints between the ports and the flows can be directly associated with a Physical architecture artefact if necessary. This association is in fact useful in the case of the Behaviour model (see Section 9.3.3); a system function can be allocated to a physical architecture of its own such that the system elements, ports, joints and flows that are used and needed by the particular system function are pointed out. Without the direct relation, a system function specific physical architecture would be impossible to present, because a system element inherits all its ports and a port inherits all its sub-ports, not only the ports and sub-ports that are relevant to the particular system function. Such a system function specific physical architecture is a partial structure of the whole system structure and is required e.g. by the safety analyst. It is also helpful for the maintenance persons to see, which system elements, ports, joints (and the corresponding joint elements) and flows need to be faultless for a specific function to work correctly.

There are two types of Flows: Non-composite flow and Composite flow. The concept of a composite flow is needed in cases, in which the actual flow is composed of two or more non-composite flows. Such an example is a quadrature encoder sensor, in which the position signal flow is composed of two primitive flows, channel A pulses and channel B pulses.

Flow is mapped to Joint. This mapping can be used during risk analyses: During signal-based HAZOP, the cause of a deviation can be pointed out in the model, e.g. it can be shown that a possible cause of a deviation 'no signal' is a break in the joint between two ports. If, however, a more detailed estimation about the probability of the connection break is needed for the safety analysis, the Joint element artefact is consulted. In the case that the joint element has not yet been designed, the analysis may provide requirements for the structure and quality of the joint components. It is of course suggested that the risk analyses of system functions are carried out before implementation of the joint elements.

A system element is allocated to a product type, and a port is allocated to a port type of the relating product type. Both system elements and ports may use application specific properties to apply the product types and port types to the specific application. The Property artefact is not depicted in **Figure 52**, nor in **Figure 53**; see Section 9.3.4 instead.

Physical architecture can be described and illustrated in one or more documents, models and diagrams, such as a block definition diagram and internal block diagram according to SysML. The documents, foreign models and diagrams are linked to the corresponding artefacts as black box artefacts (see **Figure 48** in Section 9.2).

9.3.3 Behaviour package

The Behaviour sub-package contains the models relating to the functionality of a system. The package currently consists of one model, the Behaviour artefacts model, the diagram of which is provided in **Figure 54**. Interface model for the behaviour artefacts, i.e. system tasks, system use cases and system functions, is not provided in this report.

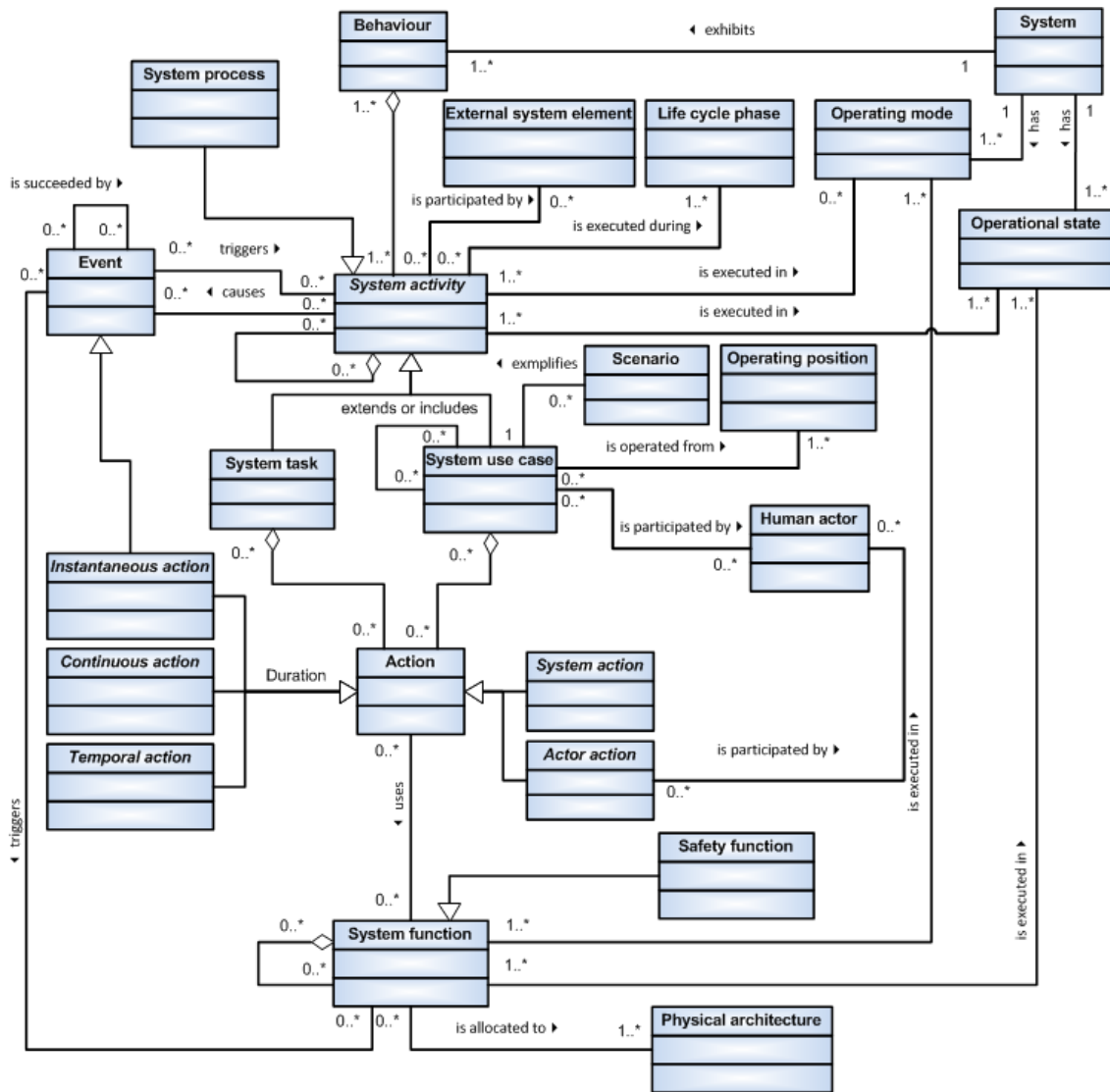


Figure 54. Behaviour artefacts model diagram.

The Behaviour artefacts model in **Figure 54** starts with the Behaviour artefact, which is exhibited by the system-of-interest from the functional point of view. The Behaviour artefact only gives a basic description of the system functionality in verbal format as captured from the stakeholders, i.e., a description of the work to be performed by the machine (the 'intended use' of the system, but it also stores the initial description of the reasonably foreseeable misuse that must be considered during the risk assessment). However, the main artefact type in the model is the System activity artefact. System activity is specialised by System process, System task or System use case artefact types. Action is specialised by System action and Actor action. System action, Actor action and System activity are abstract types, i.e. they do not have a special placeholder in the project data repository. Processes, system task and system use cases can be nested, i.e. they can have sub-processes, sub use cases and sub system tasks respectively, but actions are atomic (see Section in 3.4.1 Part I).

It is possible to define several Behaviour artefacts for a single system. This is useful in cases in which the system or machine has clearly separate ways of working or separate functional features.

The system behaviour is described in a more systematic way in system processes that can consist of sub-processes or system use cases or system tasks. The behaviour can also be described directly with system use cases or system tasks without defining system processes. Nevertheless, the system use cases and system tasks are the core artefacts to describe the behaviour, the former presenting the human actor view and the latter the system view. The use cases describe the sequence of actor actions to get the service, the added value, out of the system. The System task artefact is the system realisation view of the behaviour. It defines the sequence of system actions. It is especially useful in cases which do not involve human actors (such as in case of automatic or autonomous systems), but it is also used to identify system functions that cannot be identified from the actor actions. System use cases and System tasks use System functions provided by the (physical) system to provide the specified behaviour.

System use cases and system tasks are created to identify, analyse, and describe the functional requirements stated by the stakeholders in a systematic way. Therefore, the functional requirements shall be traced to the system processes, system use cases or system tasks (see Section 9.5 and think the Behaviour artefacts as specialisations of the Engineering artefact in Figure 59).

One possible way to work with system use cases and their actions is to write the system use cases in a platform independent way (i.e. with no reference to the underlying system elements) and the actor actions with platform dependent way. In this scheme, the actor actions are written later than system use cases, i.e. after the first release of the physical architecture of the system is available.

A system use case can include finer grained use cases or extend another system use case. The sequence of actions of a system use case is stored in the Action artefacts. The reason for separating the Action artefact from the System use case artefact is that during Operating Hazard Analysis (OHA) we need to be able to link a single atomic action to an identified hazard to provide traceability. It must be ensured, however, to extend traceability such that if e.g. the set of Human actor artefacts is changed not only the related Action artefacts are marked suspect, but also the related hazards.

A system function is specified exhaustively such that e.g. the software engineers can implement the software for the system function based on the particular system function artefact contents. The model allows for a system function to consist of one or more sub-functions. A system function is allocated to a Structure artefact of its own. Such a system function specific structure is a partial view of the actual system structure to illustrate the part of the system structure that takes part in executing the system function. Besides risk analysis, the function specific structure is useful for the maintenance personnel to understand, which components and sub-systems shall be suspected if the particular system function does not work correctly.

A system function is specified by a set of Requirement artefacts and configured by System properties. This is not depicted in **Figure 54**, but is implicitly included in **Figure 59** and **Figure 55** respectively.

System use cases, system tasks, actions and system functions can be specified, described and illustrated with any type of behaviour related documents, foreign models and diagrams, such as activity diagram, sequence diagram, state machine diagram or use case diagram, or any other functionality related diagram. The documents, foreign models and diagrams are linked to the corresponding artefacts as black box artefacts (see **Figure 48** in Section 9.2).

9.3.4 Properties package

The Properties sub-package currently provides only one model, the model for the properties of the system. The Properties artefacts model is presented in **Figure 55**.

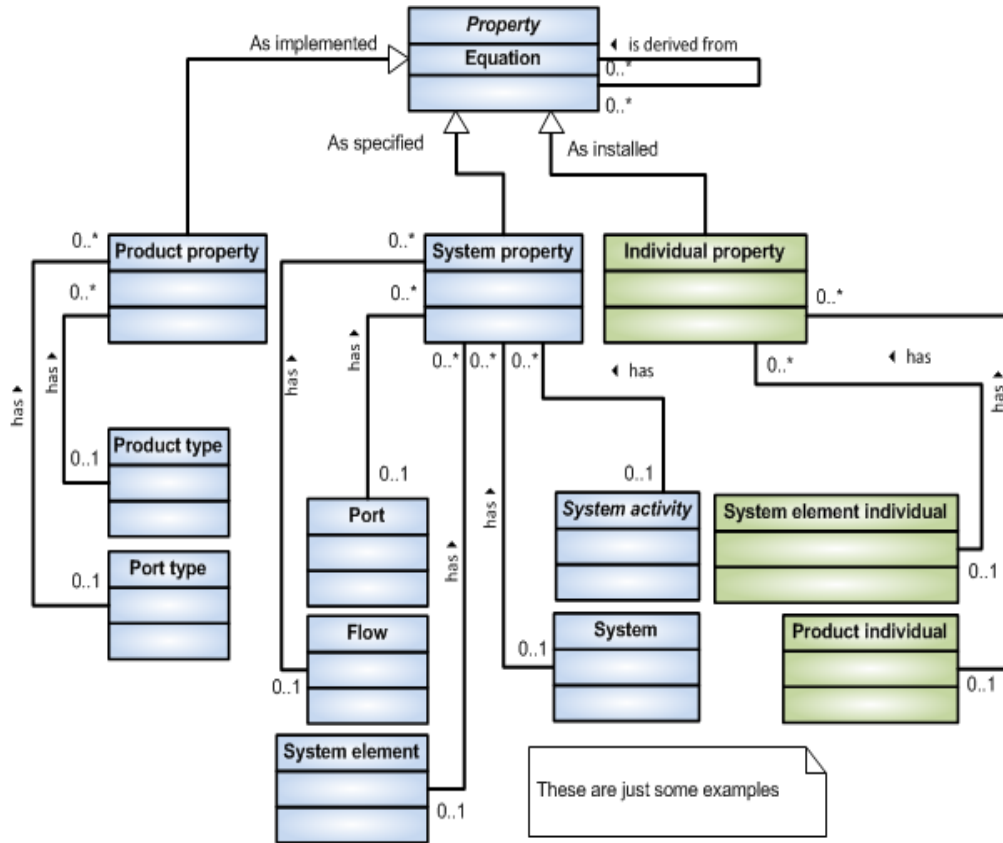


Figure 55. Properties artefacts model diagram.

The Property artefact facilitates the modelling of attributes, state and other variables, parameters and characteristics of a system and its constituents. The Property artefact is categorised by fixed properties (Product property), configurable properties (System property) and Individual properties. Fixed properties are assigned to Product type and Port type artefacts. The system properties are the parameters related to the system type and the individual properties are operation time parameters (such as operating hours) of a system type instance. An individual property of a system element provides the system integrator view (record kept by the system integrator, i.e. the provider of the system-of-interest), whereas a product individual property provides the component or sub-system view (record kept by the component or sub-system provider or by the system integrator).

Properties can be derived from other properties. For this purpose, the attributes of the Property artefact include an attribute (of string type) called Equation.

A property may need further elaboration. This is achieved by documents, models and diagrams, such as a parametric diagram according to SysML. The documents, foreign models and diagrams are linked to the corresponding artefacts as black box artefacts (see **Figure 48** in Section 9.2).

9.3.5 Product type package

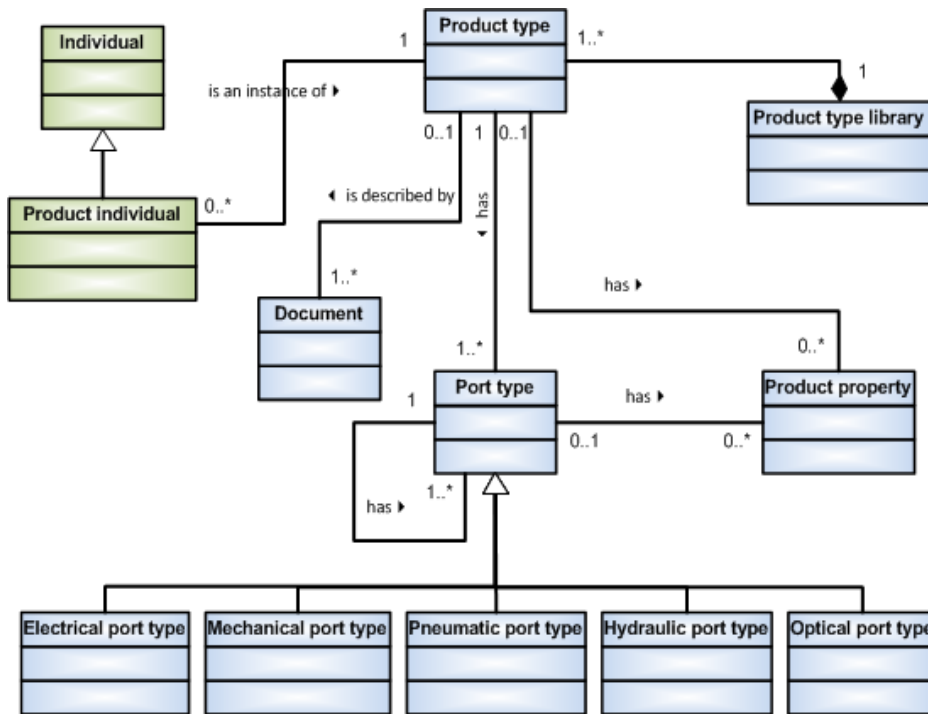


Figure 56. Product type artefacts model diagram.

The product type sub-package only contains one model, the Product type artefacts model depicted in **Figure 56**. It covers the description of a released system or component to be applied by a systems integrator or acquirers of a system. Both subsystem and component types are stored using the Product type artefact. But it also provides the platform for the developer of the system-of-interest to publish the system under development when it is ready.

The set of Product types and their ports (the port types) constitute a product type library that contains all the generic information about the product types and their interfaces (ports). A system element is allocated to a product type, and a port is allocated to a port type (see Figure 53). In other words, a product type is the physical implementation¹¹ of a system element, which is a logical architecture element.

Product type and its port types can be described by several product properties. The idea is that the datasheet information of the product types is stored into a structured data repository. However, it is also possible to attach a conventional datasheet or a CAD-model with a product type if necessary as a black box artefact as depicted in **Figure 48**. The product properties of subsystems and components cannot be changed by the system-of-interest developer, because they have been defined by the subsystem or component supplier; e.g. a component can have weight, dimensions, allowable temperature range, etc. as its product properties. Such information is normally presented in an easily readable format in conventional datasheets. But the provision of such a structured way of storing product properties in the Product property artefact

¹¹ 'Physical' does not mean that the subsystem or component is really manufactured; e.g. a CAD-model can be considered to be a physical implementation.

facilitates automatic embedding of the specification parameters in the application specific documents or drawings generated from the artefacts repository. The Product property artefact has been motivated by the MSRSYS specification (MSR Consortium 2002), in which it is called *specification parameter*. The optimal workflow would be such that the component manufacturers and sub-system suppliers provide the product type properties in XML files that can easily be incorporated into the product type library.

The product individual artefact type provides storage for the product individual data such as serial number and maintenance and warranty information.

9.4 System context package

The system context package consists of system context model (Section 9.4.1) and operational environment model (Section 9.4.2).

9.4.1 System context artefacts model

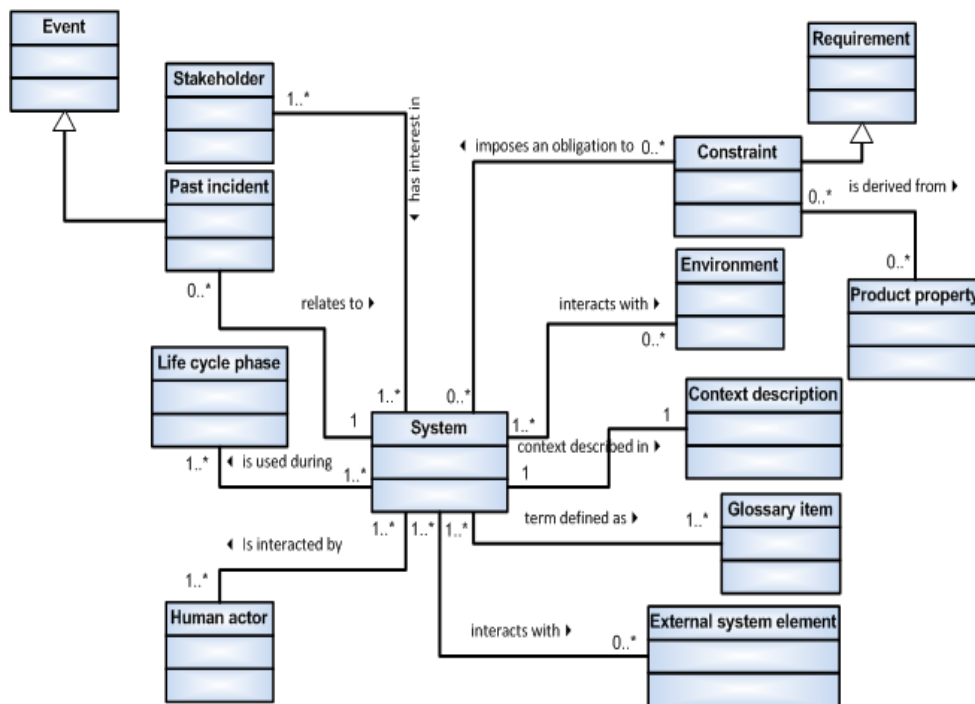


Figure 57. System context artefacts model diagram.

The system context model (Figure 57) introduces artefact types that can be used to describe the context in which the system will be used. The context includes both concrete (such as environment and human actors) and abstract issues (such as past incidents and glossary).

The model in Figure 57 introduces Constraint artefact to encompass constraints caused by the system context to the system-of-interest. An example is a case where the space of the operational environment sets requirements on the dimensions of the system-of-interest. Note that, if a constraint is caused by an external system, such constraints are directly derived from the properties of the external system, i.e. they are presented in the Product property artefacts of that system.

9.4.2 Operational environment model

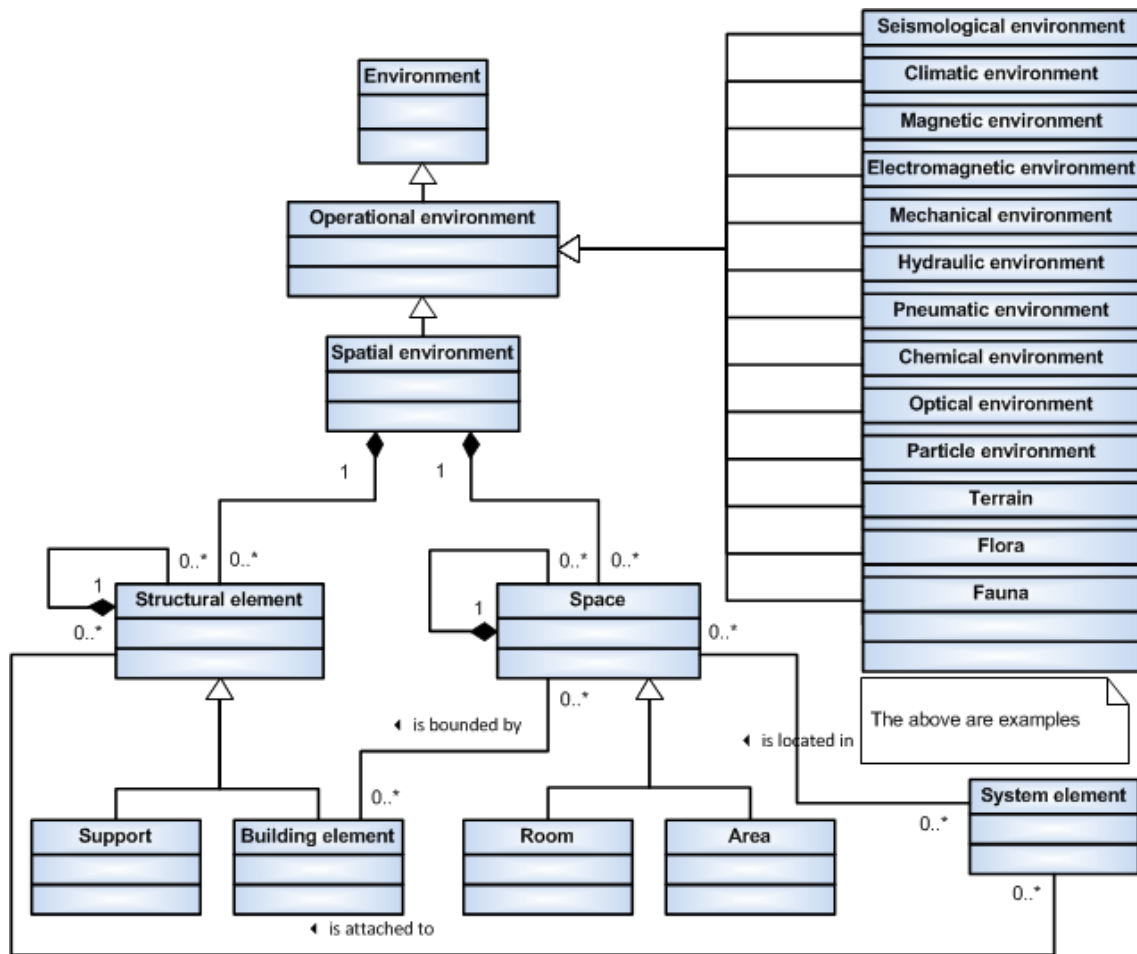


Figure 58. Artefacts model diagram for operational environment.

Operational environment is part of the System context package. It elaborates the environmental model from the operational environment point of view. There are several environment types to consider, some of which are provided in Figure 58. Only the spatial environment of those is elaborated further. The spatial environment models the structural elements and spaces the system-of-interest is going to be located in during operation.

9.5 Requirements and V&V package

The Requirement and V&V package consists of only one model, the Requirements and V&V artefacts model, which is presented in **Figure 59**.

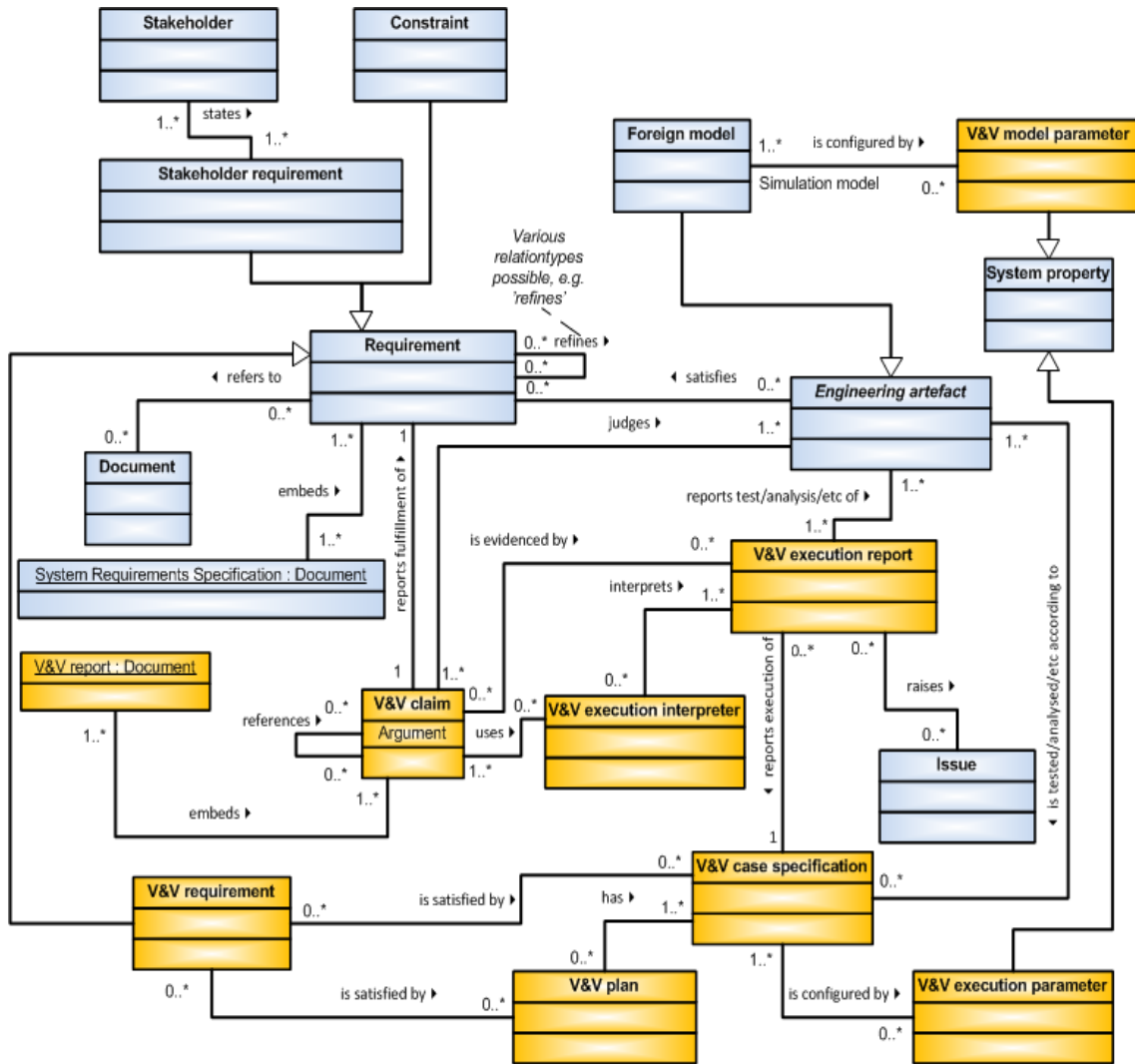


Figure 59. Requirements and V&V artefacts model diagram. (Verification and validation artefact types are distinguished by yellow colour.)

The diagram in Figure 59 presents the model for tracing requirements to design and implementation, from implementation to test execution, and from test execution to verification or validation reporting¹². We call this the ‘core loop of Systems Engineering’. The requirements and V&V artefacts model is the starting point for implementing SEAModel_{NPP}.

The central point for the requirement artefacts model is the Requirement artefact that stores both the stakeholder and system requirements. The Stakeholder requirement is a special case of a requirement. It relates to one or more Stakeholders. During the system validation at the end of the project or at project gate points, the system design is compared to the stakeholder requirements to prove that the system fulfils the expectations of the stakeholders. The system requirements reflect the engineers’ understanding of the stakeholder requirements. The top

¹² It should be noted that the particular diagram is not a traceability information model (TIM) as such. An example of TIM is provided in Chapter 10.

level system requirements are derived from the stakeholder requirements¹³. A requirement can have child requirements that refine the parent requirement. In this case, the trace role is 'refine', but other roles can also be created. Documents and foreign models can be linked to the requirements where necessary according to the principles described in Section 9.2.

The engineering artefacts¹⁴ (including project process activities) that are claimed to satisfy the requirements are verified or validated during the development work. A special set of artefact types are provided for this purpose: V&V requirement, V&V plan, V&V case specification, V&V model parameter, V&V execution parameter, V&V execution report, V&V execution interpreter and V&V claim. A V&V case specifies tests, analyses, simulations or any other methods to be used to evidence fulfilling of the requirements for the design or project process activity. A V&V execution report must be traced to the artefact under verification or validation to prompt for a test (or analysis etc.) re-execution if the artefact under test is updated. The V&V execution interpreter provides a means to relate a V&V claim with instructions or tools to interpret the V&V execution results, such as measurement files. The result of the test or analysis is stored in the V&V execution report artefact, and the verification or validation result is recorded in the V&V claim artefact. It may be necessary to execute more than one test or analysis cases for the evidence of a successful result of the validation. Hence the V&V claim artefact has a zero-to-many relation to the V&V execution report artefact.

Based on the results of the execution of a test or an analysis, an issue can be raised to initiate corrective actions. The issue is managed according to the modification procedure of the project.

V&V plan collects a set of V&V case specifications to form a specific sequence of tests for a specific purpose, such as for Site Acceptance Test (SAT).

9.6 Specialty engineering package

The Specialty engineering package contains the models for different disciplines such as risk assessment (human safety), security, sustainability engineering, human factors engineering (ergonomics), dependability and logistics. Currently only artefacts model for risk assessment has been defined (see Section 9.6.1).

¹³ The rule is that there should be a trace from each stakeholder requirement to at least one system requirement (to ensure full coverage of stakeholder requirements), and that there should be a trace from each system requirement to at least one stakeholder requirement either directly or through parent system requirements (to disallow not required system requirements).

¹⁴ Engineering artefacts can include various types of artefacts presented in the artefacts models, such as System use case, System task, System function, System element, Port, Flow, Joint and Product type. Figure 59 does not depict the special artefact but only their generalisation artefact type 'Engineering artefact'.

9.6.1 Risk assessment package

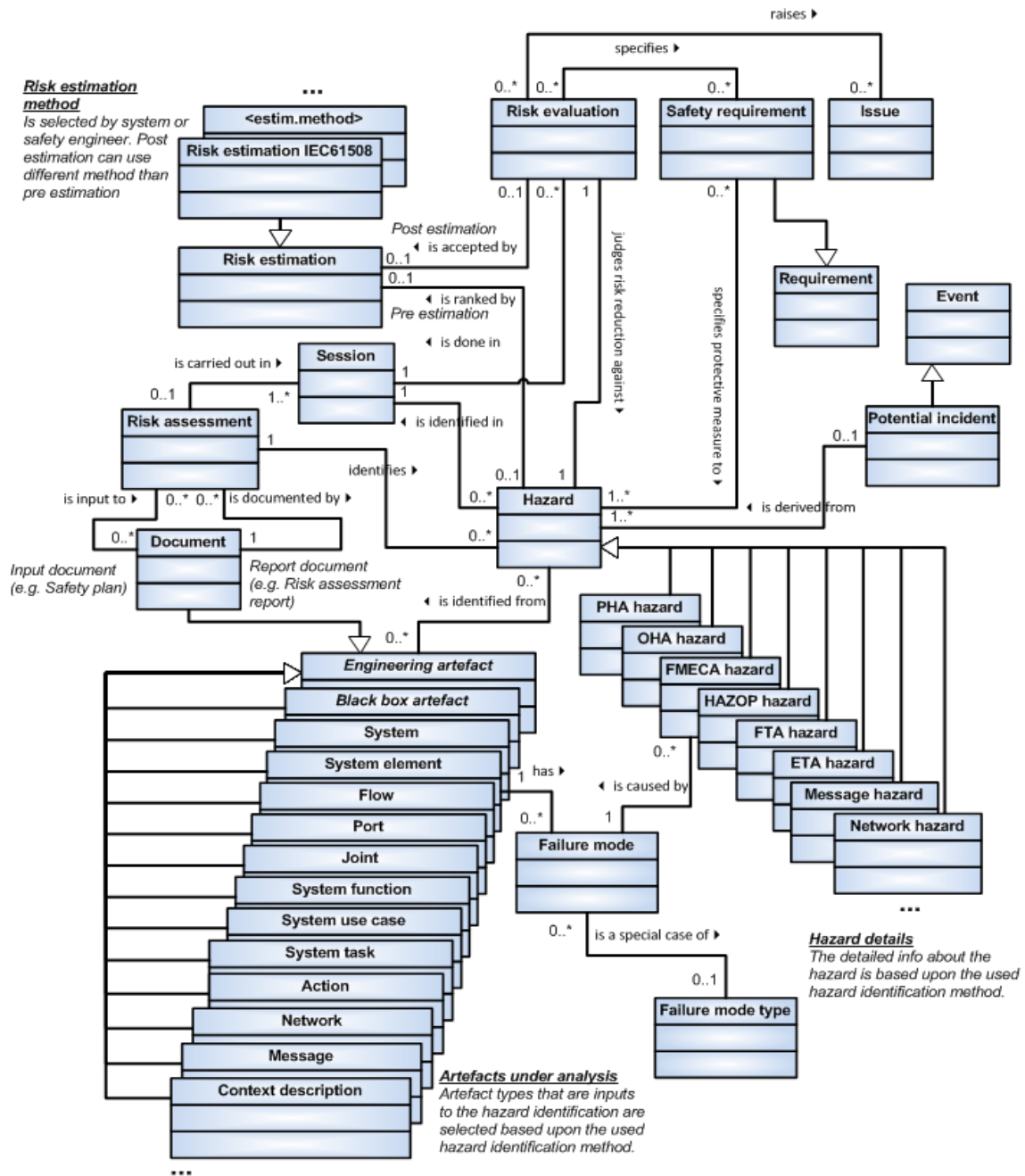


Figure 60. Risk assessment artefacts model diagram.

The Risk assessment sub-package only contains the risk assessment artefacts model presented in Figure 60. It is based on the ISO 12100 (2010) risk assessment model depicted in Figure 61.

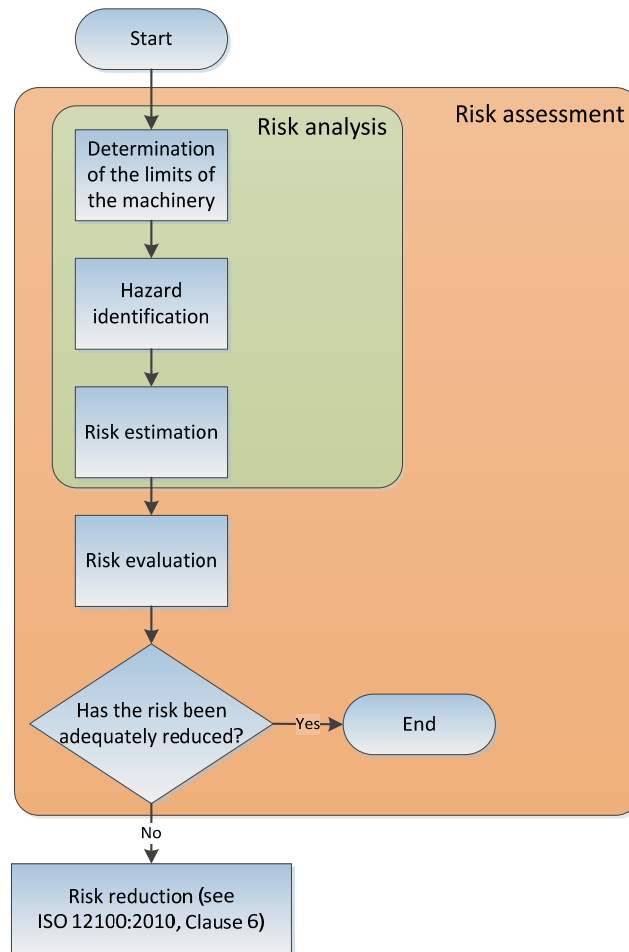


Figure 61. Risk assessment process model according to ISO 12100 (2010) (Risk analysis = combination of the specification of the limits of the machine, hazard identification and risk estimation; Risk estimation = defining likely severity of harm and probability of its occurrence; Risk evaluation = judgment, on the basis of risk analysis, of whether the risk reduction objectives have been achieved; definitions from ISO 12100).

The generic information concerning the risk assessment is stored in the Risk assessment artefact. It specifies e.g. the type of risk analysis used for the particular assessment; currently the following analysis types are supported: Preliminary Hazard Analysis (PHA), Operating Hazard Analysis (OHA), Failure Mode Effects and Criticality Analysis (FMECA), Fault Tree Analysis (FTA), Hazard and operability study (HAZOP)¹⁵, message safety analysis and network safety analysis.

A risk assessment is performed in several analysis sessions the minutes of which are recorded in the Session artefact. During the PHA-, OHA- etc. sessions, Hazards are identified based on the analysis type specific methods. The source information (the items under analysis) for the analyses is typically found among the following set of engineering artefacts:

- System(-of-interest) (typically for PHA)
- System context related artefacts (typically for PHA)

¹⁵ It is assumed here that HAZOP is used for Functional Hazard Analysis (FHA). If not, HAZOP can be replaced or complemented with a specific FHA method.

- System use case (typically for PHA and OHA)
- System task (typically for PHA and HAZOP)
- Action (typically for OHA and HAZOP)
- System function (typically for FMECA, FTA and HAZOP)
- Flow (typically for HAZOP)
- System element (typically for FMECA)
- Port (typically for FMECA, but only for rare cases if any)
- Joint ([or in practice Joint component, i.e. a system element] typically for FMECA)
- Message (for message safety analysis only)
- Network (for network safety analysis only)
- any other artefact type, including Document and Foreign Model.

The analysis will result in different types of Hazards. In the model, they are categorised according to the analysis method that revealed the hazard. Hence there are seven special cases of the Hazard artefact:

- PHA Hazard
- OHA Hazard
- FMECA Hazard
- HAZOP Hazard
- FTA Hazard
- Message Hazard
- Network Hazard.

After a hazard has been identified, its risk level will be estimated and recorded in the Risk estimation artefact. The model enables several alternatives for the risk estimation method; currently only the risk estimation method of IEC 61508 is depicted. The risk estimation method is determined by an attribute in the Risk assessment artefact; this attribute is set by the systems engineer or safety engineer.

Corrective actions will be recommended if the existing protective measures are not sufficient to reduce the risk. The existing protective measures must be evidenced in the form of existing safety requirements and linked to the particular hazard; e.g., during analysis sessions a person may point out that there is an overload limiting device in the system and thus claims that the risk of the identified hazard is negligible. The analyst must not simply write down the claimed protective measure, but he or she must browse the Requirement artefacts (a database or similar storage in practice) and pick up the requirement for the overload limiter and link the requirement to the hazard. This ensures that if a change is made to the specifications, e.g. the requirement for the overload limiter is removed, the particular hazard automatically becomes suspect, and an update to the particular analysis of the hazard is promptly requested.

Recommendations for corrective actions will be handed over to a team of evaluators who will judge upon the adequacy of the suggested protective measures and decide upon the final implementation of the protective measures against the identified hazard. The judgement is recorded in the Risk evaluation artefact, but the actual result of the risk evaluation is one or more new safety requirements (if needed). The resulting safety requirements are not necessarily a direct copy of the corrective action recommendations by the risk analysis team, but may be modifications of the corrective action recommendations. Hence the Risk evaluation artefact includes rationale on the modifications or direct acceptance of the corrective action recommendations. The resulting safety requirements are linked to the Risk evaluation artefact to provide a trace to

the hazard causing the particular safety requirements. In the end, the particular safety requirements are validated according to the requirements model in Section 9.5.

However, there are cases in which risk evaluation may lead to a change in the original specifications instead of creating new safety requirements; e.g., the risk analysis team may recommend equipping a machine with a collision avoidance system, but the risk evaluation team may find it too expensive to implement and creates an issue to change the original specifications, e.g. to strip off features that are difficult to implement cost-efficiently with an acceptably low safety risk. The raised issue is handed over to the modification procedure of the project.

The evaluator team together with the safety engineer can redo the risk estimation to ensure that the acceptable risk level has been reached with the stated new safety requirements.

The communications analysis is performed in two parts: a message safety analysis and a network safety analysis. The former is performed according to the model of IEC 61784-3 (2007) and the latter according to the network validation questions by the Swedish PÅLBUS project (Hedberg and Wang 2001) with VTT modifications.

Besides the well-structured input artefacts, one or more Documents may be provided for the analysis team as input to the analysis. Such documents include e.g. the relevant safety standards. Also a safety plan is a typical document to guide the analysis team in carrying out the analysis requested by the Risk assessment artefact.

The results of the risk assessment are recorded in a Document artefact, e.g. in a collective Risk Assessment Report.

9.7 Projects package

The project package is not modelled in detail in this report. Nevertheless, **Figure 62** illustrates some examples of project artefacts.

It should be noted that typically many of the stakeholder or system requirements are project process requirements, such as documentation or licensing requirements. The requirements and V&V artefacts model in Section 9.5 allows linking of requirements to relevant project artefacts such as Engineering process, Project activity and Project task.

Note also that, as stated in Section 9.2, in some cases it is not easy or possible to define, whether an artefact is a project, system or system context artefact. Examples of such artefacts are requirements, constraints, hazards and verification and validation artefacts. The examples of project artefacts in **Figure 62**, except Role and Person are such that they need to be added if project management data is to be stored together with the system data in the project data repository. The Role and Person artefacts may be needed also in the system model if some of the system elements are persons. The Issue artefact is already present in the requirements and V&V artefacts model (Section 9.5) and in the Risk assessment artefacts model (Section 9.6.1). Modification request, Impact analysis report and Modification log items are implicitly included through the Document model (Section 9.2) through the document kinds (request, report, log, etc.), and hence they do not necessarily cause any additional artefact types into the project data repository, but it is advised to implement the modification procedure artefacts as model elements, and then generate the appropriate documents out of them where needed.

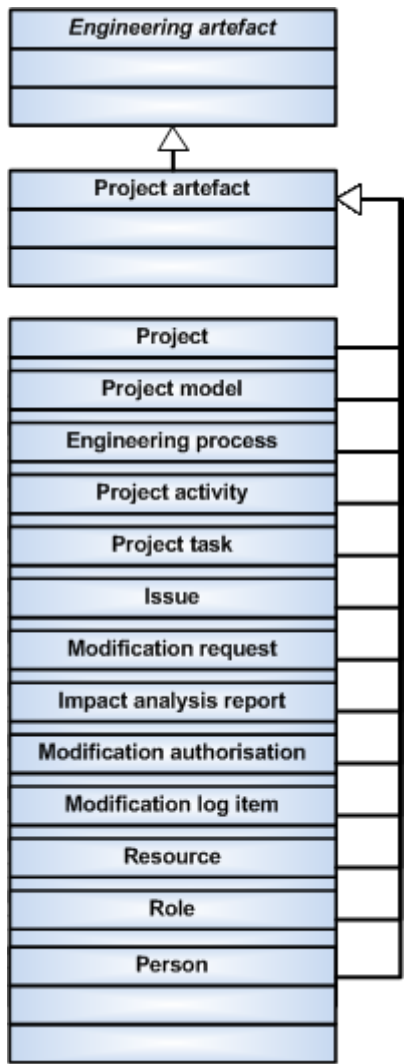


Figure 62. Examples of project artefacts.

10. Traceability information model

Traceability (see Chapter 6 in Part I) makes it possible to find out how and why certain design solutions have been derived. It requires some information to be created and maintained as part of the design data. Engineering tools should show the dependencies and indicate potential impacts of the changes made to one artefact. Traceability information models (TIM) depict how different engineering artefacts trace to one another. The concept model diagrams provided in Chapter 9 are very close to what the traceability information diagrams look like. The differences between the traceability information diagrams and artefacts model diagrams are as follows:

- Not all the relations in the artefacts model diagrams need to be trace links.
- The direction of a trace link is from the younger information to the older; in artefacts model diagrams the link directions (shown in the context of the relation name) do not necessarily reflect the trace direction.
- Relation names are changed to their reciprocal (inverse) versions in some cases to provide reading order from the source to target direction.

An example traceability information model, derived from the Requirements and V&V artefacts model is illustrated in **Figure 63**.

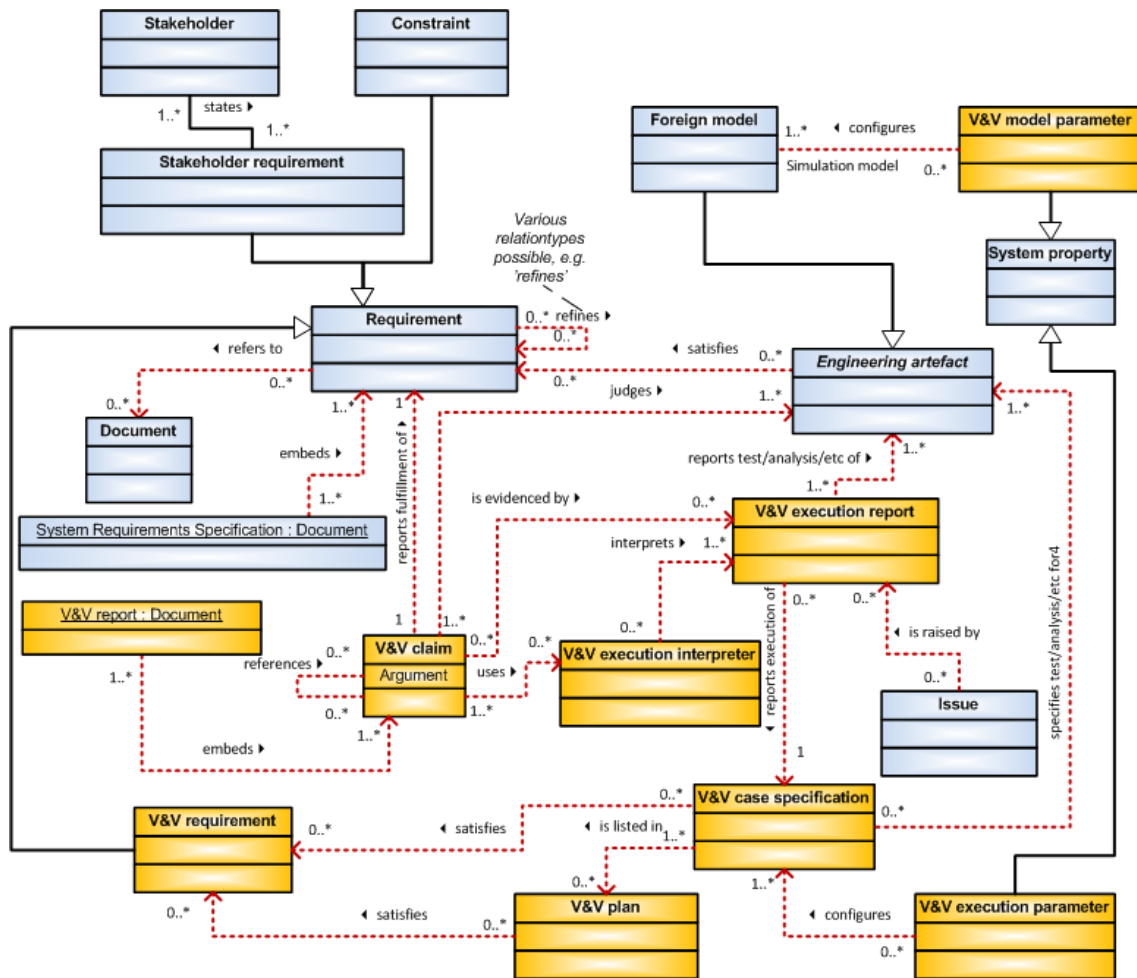


Figure 63. Traceability information model out of Requirements and V&V artefacts model. (Trace links are denoted by red colour, dashed line and with an arrowhead.)

The relationship names of Figure 59 are updated to their reciprocal (inverse) versions in **Figure 63** in some cases to provide reading order from the source to target direction. The arrowed trace link points in each case from the younger information to the older information. If the artefact in the target end is changed the source artefact becomes suspect, and possibly needs to be updated consequently. If the artefact in the source end is changed, it has to be checked that it still is consistent with its unchanged parent. As can be seen in **Figure 63** compared to Figure 59, all the other except the relationship between Stakeholder and Stakeholder requirement¹⁶ manifest themselves as trace links.

In a similar fashion, all the other concept diagrams in Chapter 9 can be converted to traceability diagrams.

¹⁶ It is possible to make the relation between Stakeholder and Stakeholder requirement a trace link as well, but it is assumed here that if the information about a stakeholder changes, the stakeholder requirements do not change. If a stakeholder role is occupied by a new person or organisation, the stakeholder requirements might be updated consequently. Nevertheless, updating the stakeholder requirements in that case is not initiated by the trace analysis (i.e. impact analysis), but by more powerful project procedures.

TIM is applied to each system, whether a system-of-interest or a subsystem developed in-house. For off-the-shelf sub-systems and components, the particular TIM may or may not have been used; it does not matter for the systems engineer of the system-of-interest as long as he or she receives the necessary data from the sub-system or part manufacturer to be stored into the *Product type* artefact. In case the system developer's TIM is not used by the system element developer, it is difficult to arrange seamless traceability of requirements from the main system to the system elements, and traceability of verification artefacts from the system elements to the main system. Such a full blown traceability in the hierarchy from top to bottom and back may be needed only in rare cases.

To implement the traceability information model, the project data repository platform has to support traceability management including impact analysis.

10.1 Traceability demonstration

The TIM in **Figure 61** (and some of the artefacts in **Figure 53**) was demonstrated in context of an imaginary instrumentation and control (I&C) system for a nuclear power plant. The demonstration case is illustrated in Figure 64. The I&C system architecture is depicted in SysML notation in **Figure 65**.

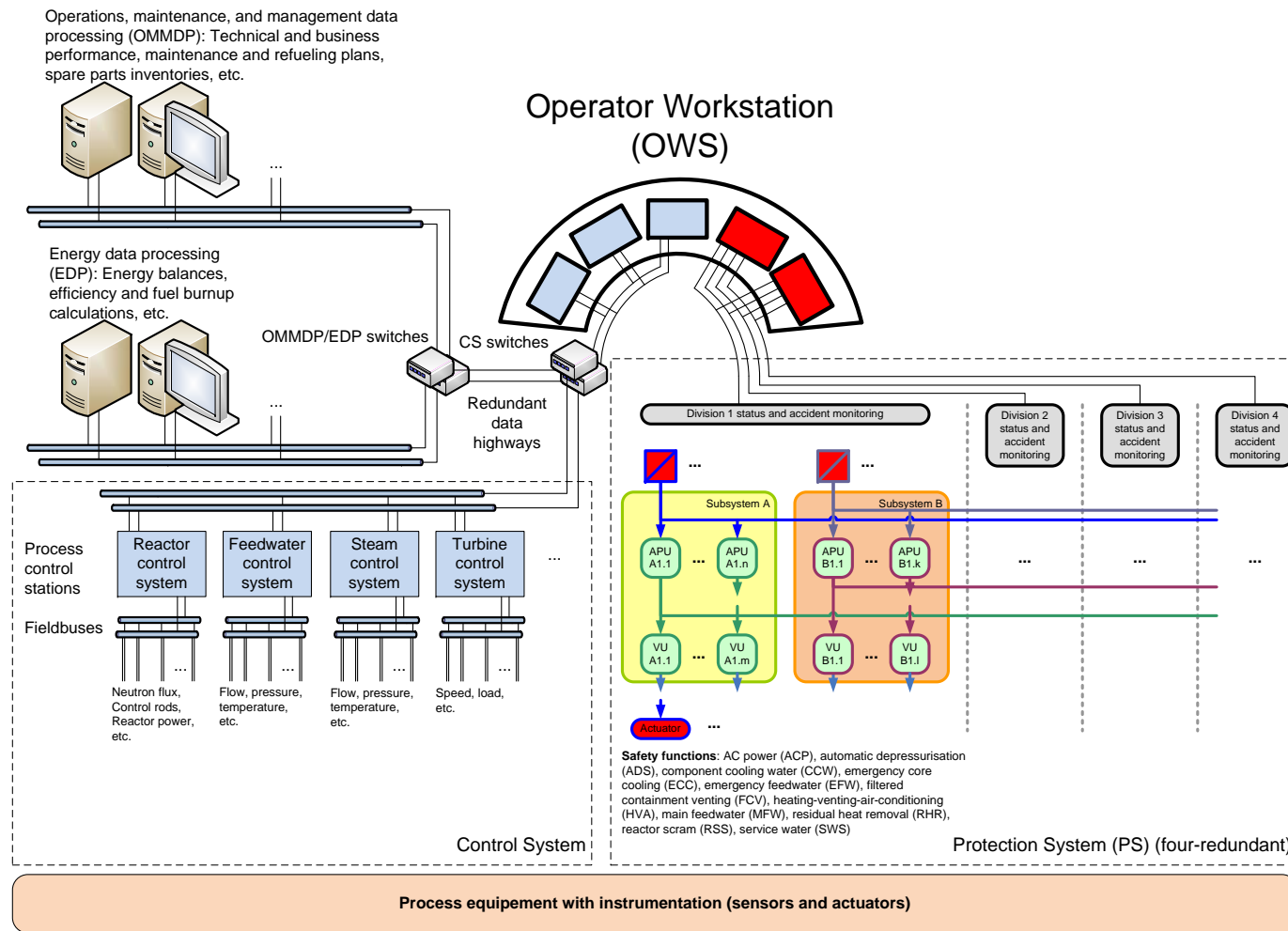


Figure 64. The artificial I&C system demonstrator. (The system architecture is an adaptation from the I&C systems architecture examples in (NRC 1997) and (Authén & Holmberg 2013).)

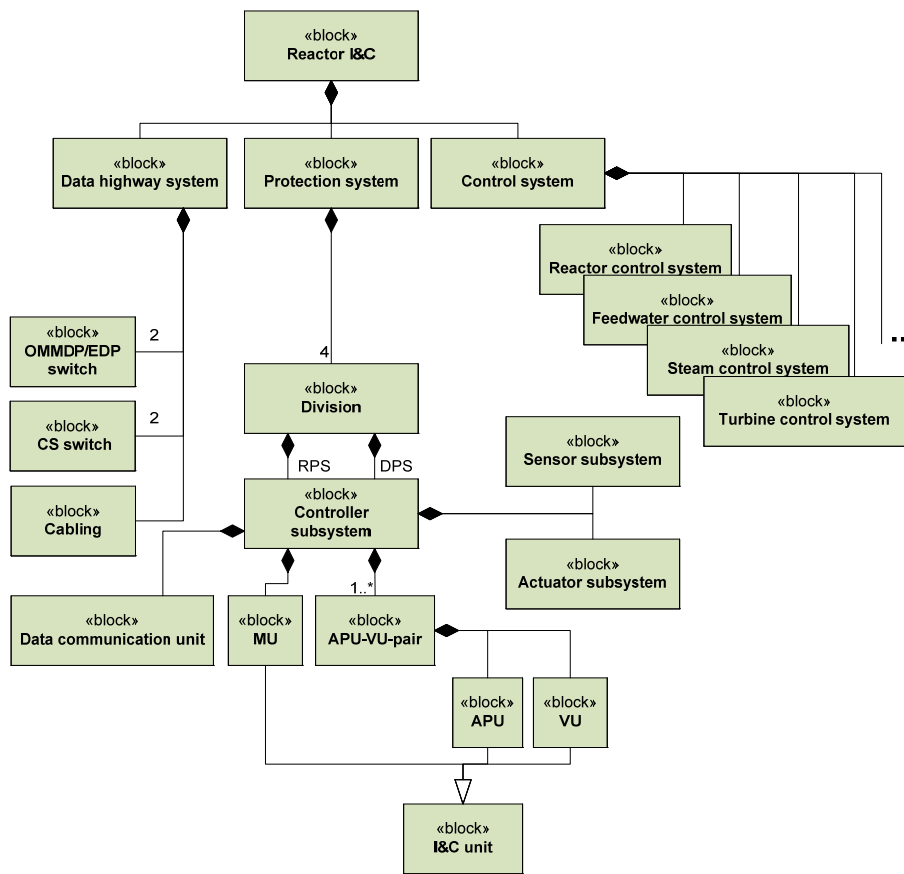


Figure 65. Demonstration case system architecture in SysML notation.

The goal of the demonstration was to implement tracing of a single stakeholder requirement up till validation of the system. The trace was followed from the I&C system level to the next sub-system level, Data highway system, to simulate the contractor – subcontractor requirements engineering data flow. The example requirement was derived from the Finnish nuclear regulation YVL Guide B.1 (2013). The example requirement concerns physical isolation between the I&C system and the administrative computer system:

The interface of the I&C architecture to administrative computer systems shall be implemented by making the transmission of data unidirectional in such a way that any transmission of data towards the I&C architecture is prevented through separation at the physical level. (YVL-B.1-5.2.6-5245)

The example requirement is a stakeholder requirement that is analysed, and a system requirement is created upon the results of the analysis. The system requirement is passed on to the developer of the Data highway system. The particular system requirement becomes thus a stakeholder requirement of the Data highway system developer. The requirement is analysed by the Data highway system developer, and four system requirements are created upon the analysis. The example requirements to be traced are depicted in **Figure 66**. The architecture design and its implemented system elements are then validated as depicted in **Figure 67** and **Figure 68**. (Note that in some cases the relation names in the following diagrams are not consistent

with those in Chapter 9. This is due to the fact that reciprocal relation names are used to provide reading order according to the trace link arrow.)

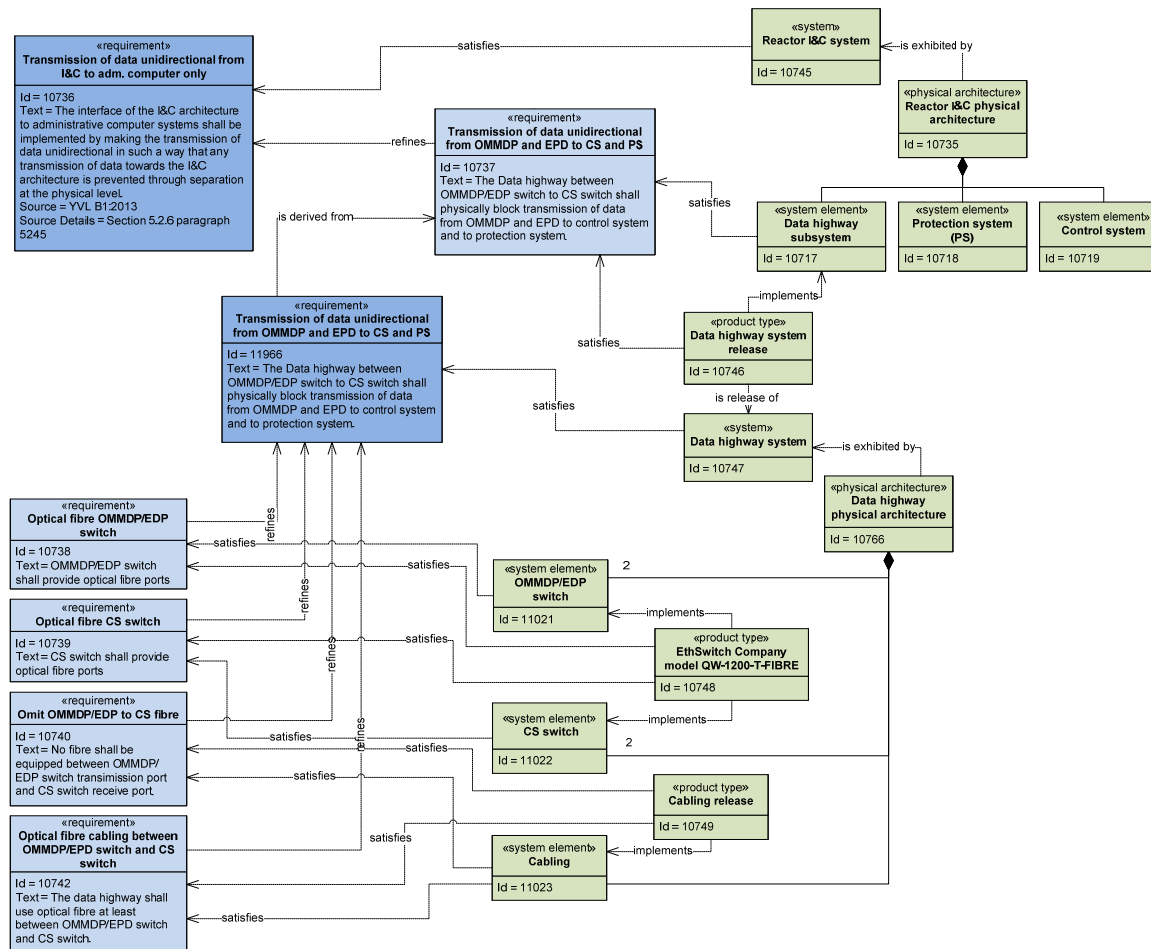


Figure 66. Demonstration case example requirements. (The dark blue artefacts are stakeholder requirements and the light blue artefacts are system requirements; the green artefacts are physical architecture artefacts.)

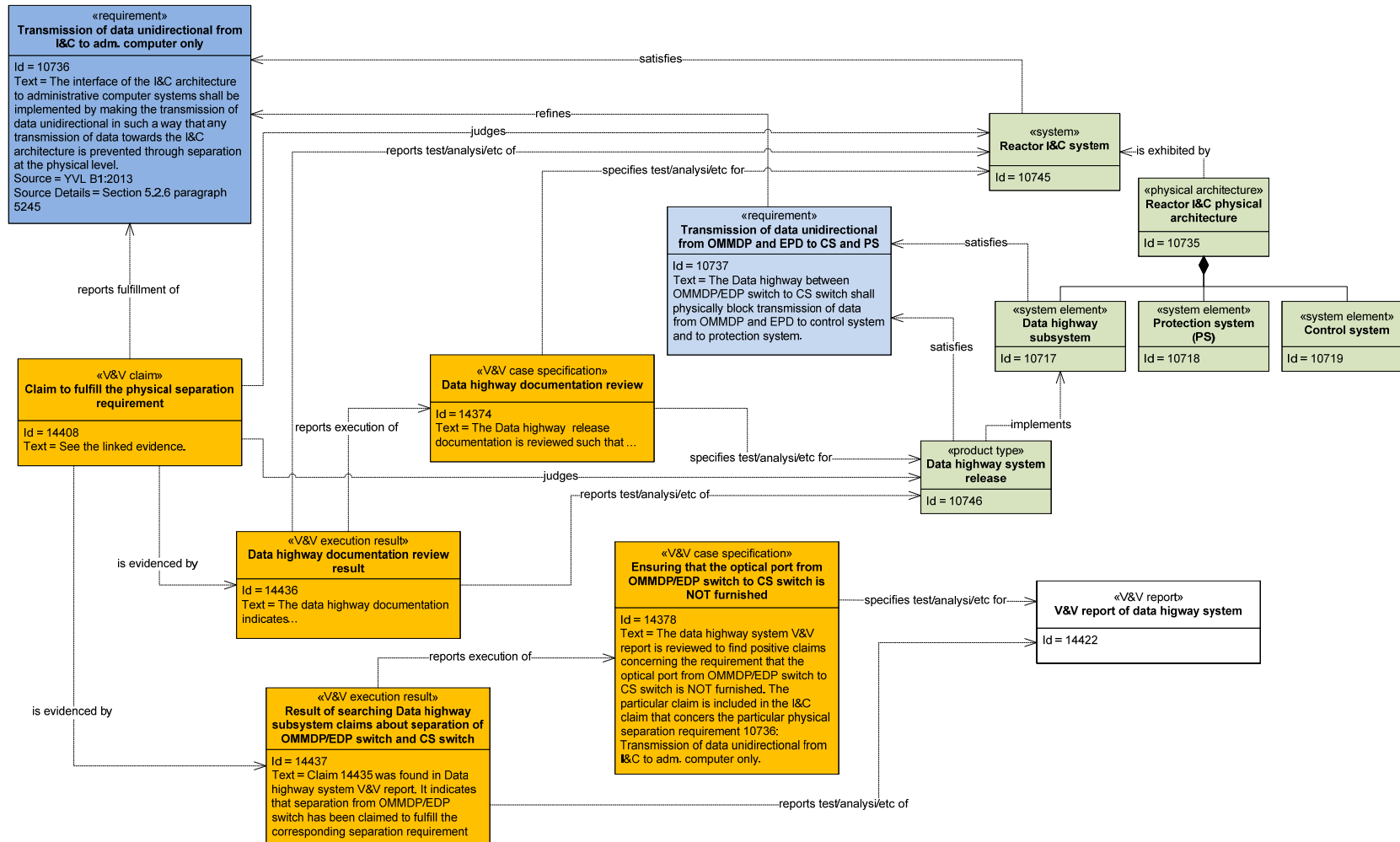


Figure 67. Tracing of verification and validation artefacts (the yellow artefacts); I&C system level.

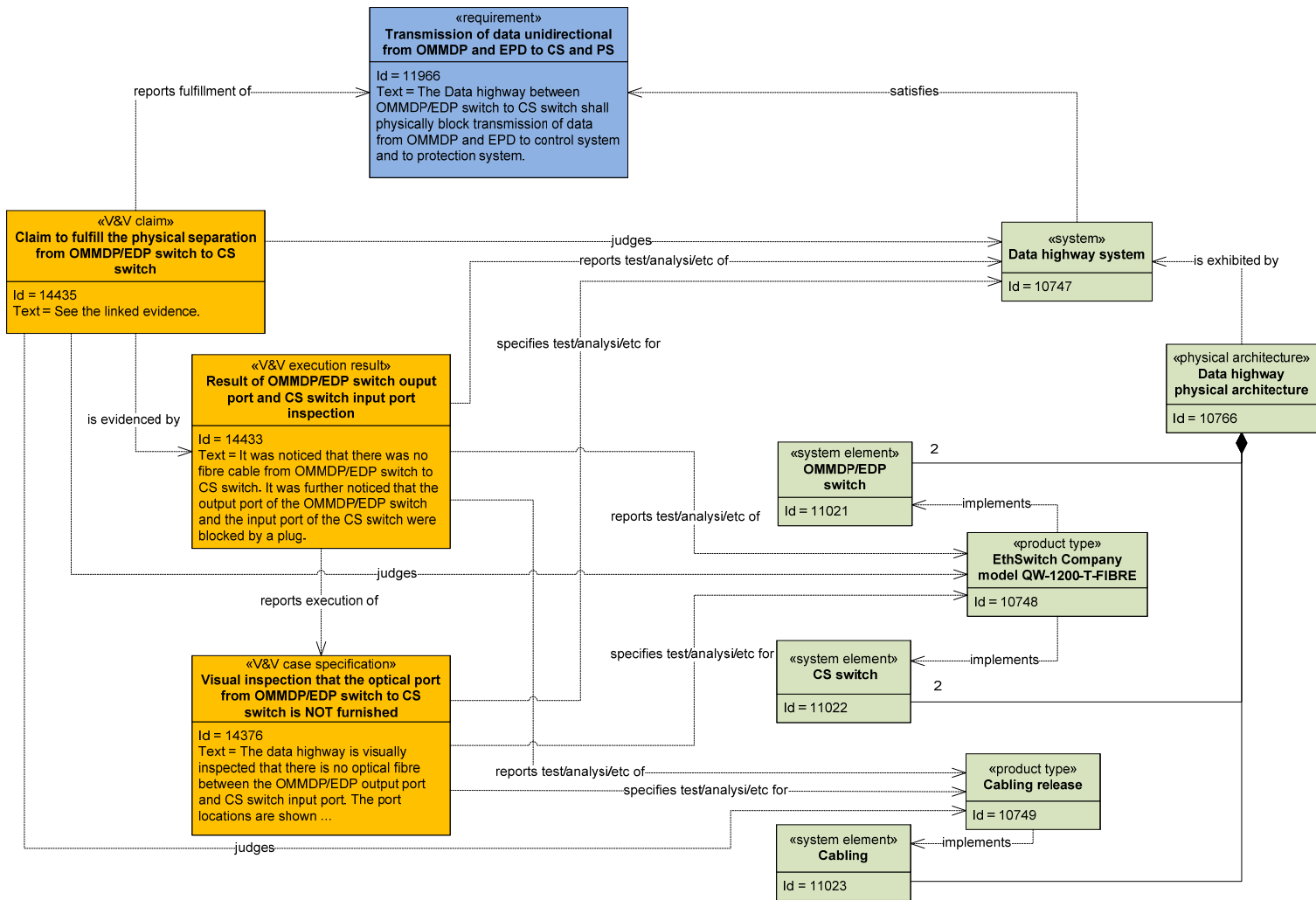


Figure 68. Tracing of verification and validation artefacts; Data highway system level.

The trace examples in the figures above were implemented using the IBM® Rational® Doors® Next Generation (NG) requirements management tool. The Doors NG tool was prepared to accommodate the engineering artefacts according to the SEAModel_{NPP} by arranging the project data repository structure as follows (see **Figure 69**).

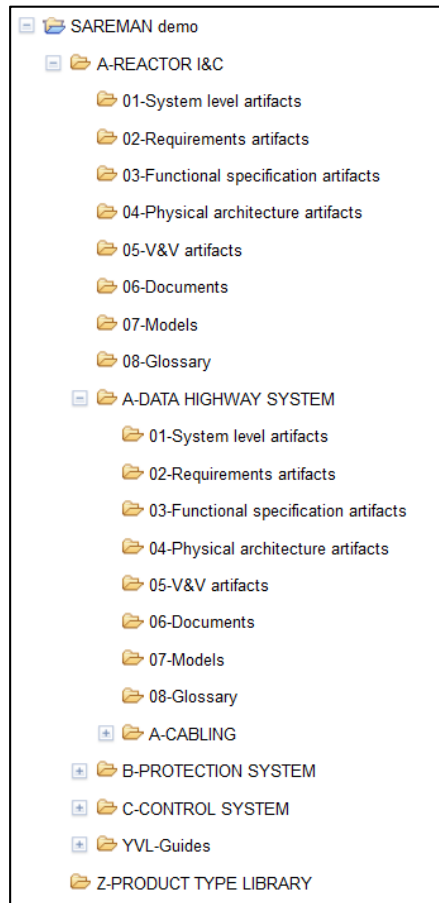


Figure 69. The folder structure of the Doors NG project¹⁷.

Doors NG provides two types of objects, modules and artefacts. A module resembles a traditional document, but each of its headings and paragraphs is a Doors NG artefact. A single module can include several types of artefacts (not only requirements artefacts).

The modules are located in the 'system folders' (the folders the name of which starts with a letter), and the artefacts are located in the artefact folders (the folders the name of which starts with a number). The product types (that is the released implementations) are published in the common product type library (Z-PRODUCT TYPE LIBRARY in **Figure 69**). An additional folder for the Finnish nuclear regulations (the YVL guides) was created. The YVL Guides folder was occupied by one example guide, YVL B.1, both as a Doors NG module and as a PDF-

¹⁷ The folder structure reflects here the hierarchical decomposition of the system, but in some cases it may be better to use flat structure for the system folders. This is because a system can be a subsystem to two or more systems.

document; the former provides tracing of the original requirement at a single paragraph level, the latter at a document level.

The following Doors NG modules were created (see **Figure 70**). As can be seen, the V&V plan and V&V report are implemented as modules, whereas artefacts such as Requirement, System element, V&V specification, V&V execution report and V&V claim were implemented as individual artefacts that were embedded into the modules listed in **Figure 70**.







Name ^	Artifact Type
 A - System concept and context description	System
 B - System requirements specification	Requirements Module
 C - Behaviour specification	Functional specification Module
 D - Physical architecture description	Physical architecture Module
 E - V&V plan	V&V plan Module
 F - V&V report	V&V report Module

Figure 70. Doors NG modules created for the demonstration.

Figure 71 and **Figure 72** illustrate tracing of requirements in the Doors NG tool. The figures present the implementation of the requirement traces depicted in **Figure 66**. A plan to validate the design to claim conformance to the requirements was created (see **Figure 73** and **Figure 74**). The figures present the implementation of the V&V case specification traces depicted in and **Figure 68**. The validation was executed according to the plan. The results of the validation were reported in a validation report (see **Figure 75** and **Figure 76**). The figures present the implementation of the V&V execution report and V&V claim traces depicted in **Figure 67** and **Figure 68**.







ID	Artifact Type	Contents	Is Stakeholder Req	Is Derived From	Refines	Is Satisfied By
10812	Heading	-4.12 Safety requirements				
10823	Heading	-4.12.1 Independence and separation requirements				
10736	Requirement	The interface of the I&C architecture to administrative computer systems shall be implemented by making the transmission of data unidirectional in such a way that any transmission of data towards the I&C architecture is prevented through separation at the physical level.	True	 14110:YVL-B.1-5.2.6-5245		 10745:Reactor I&C system
10737	Requirement	The Data highway between OMMDP/EDP switch to CS switch shall physically block transmission of data from OMMDP and EPD to control system and to protection system.	False		 10736:Transmission of data unidirectional from I&C to adm. computer only ( B - System requirements specification)	 10717:Data highway subsystem  10746:Data highway system release

Figure 71. An excerpt from the System requirements specification module of the I&C system. (Read first the Contents column cell and then the trace type name and then the corresponding cell, such as “The interface of the ... is derived from 14410: YVL-B-1.5.2.6:5242”).

ID	Artifact Type	Contents	Is Stakeholder Req	Is Derived From	Refines	Is Satisfied By
11822	Heading	-4.12 Safety requirements				
11823	Heading	-4.12.1 Independence and separation requirements				
11966	Requirement	The Data highway between OMMDP/EPD switch to CS switch shall physically block transmission of data from OMMDP and EPD to control system and to protection system.	True	10737:Transmission of data unidirectional from OMMDP and EPD to CS and PS		10747:Data highway system
10738	Requirement	OMMDP/EDP switch shall provide optical fibre ports	False		11966:Transmission of data unidirectional from OMMDP and EPD to CS and PS (B - System requirements specification)	11021:OMMDP/EDP Switch component 10748:EthSwitch Company model QW-1200-T-FIBRE
10739	Requirement	CS switch shall provide optical fibre ports	False		11966:Transmission of data unidirectional from OMMDP and EPD to CS and PS (B - System requirements specification)	11022:CS Switch component 10748:EthSwitch Company model QW-1200-T-FIBRE
10740	Requirement	No fibre shall be equipped between OMMDP/EDP switch transmission port and CS switch receive port.	False		11966:Transmission of data unidirectional from OMMDP and EPD to CS and PS (B - System requirements specification)	11023:Cabling subsystem 10749:Cabling release
10742	Requirement	The data highway shall use optical fibre at least between OMMDP/EPD switch and CS switch.	False		11966:Transmission of data unidirectional from OMMDP and EPD to CS and PS (B - System requirements specification)	11023:Cabling subsystem 10749:Cabling release

Figure 72. An excerpt from the System requirements specification module of the Data highway system.




ID	Artifact Type	Contents	Specifies Test/Analysis/Etc For
14361	Title	V&V plan of the I&C system	
14363	Heading	- 1 Introduction	
14366	Heading	1.1 Purpose of the document	
14368	Heading	1.2 Definitions, acronyms and abbreviations	
14369	Heading	1.3 References	
14364	Heading	2 Description of the system under validation	
14370	Heading	3 V&V strategy	
14371	Heading	- 4 Validation activities	
14372	Heading	4.1 V&V tasks, schedules and responsibilities	
14373	Heading	- 4.2 V&V case specifications	
14374	V&V case specification	The Data highway release documentation is reviewed such that ...	 10745:Reactor I&C system  10746:Data highway system release
14378	V&V case specification	<p>The data highway system V&V report is reviewed to find positive claims concerning the requirement that the optical port from OMMDP/EDP switch to CS switch is NOT furnished. The particular claim is included in the I&C claim that concerns the particular physical separation requirement 10736:</p> <p>Transmission of data unidirectional from I&C to adm. computer only.</p>	 14422:F - V&V report

Figure 73. An excerpt from the V&V plan module of the I&C system.




ID	Artifact Type	Contents	Specifies Test/Analysis/Etc For
14380	Title	V&V plan of the Data highway system	
14381	Heading	-1 Introduction	
14382	Heading	1.1 Purpose of the document	
14383	Heading	1.2 Definitions, acronyms and abbreviations	
14384	Heading	1.3 References	
14385	Heading	2 Description of the system under validation	
14386	Heading	3 V&V strategy	
14387	Heading	-4 Validation activities	
14388	Heading	4.1 V&V tasks, schedules and responsibilities	
14389	Heading	-4.2 V&V case specifications	
14376	V&V case specification	The data highway is visually inspected that there is no optical fibre between the OMMDP/EDP output port and CS switch input port. The port locations are shown ...	 10747:Data highway system  10749:Cabling release  10748:EthSwitch Company model QW-1200-T-FIBRE

Figure 74. An excerpt from the V&V plan module of the Data highway system.

ID	Artifact Type	Contents	Result	Reports Execution Of	Reports Test/Analysis/Etc Of	Is Evidenced By	Reports Fulfilment Of	Judges
14402	Heading	4.2 V&V case execution results						
14436	V&V execution report	The data highway documentation indicates...		14374:Data highway documentation review	10745:Reactor I&C system 10746:Data highway system release			
14437	V&V execution report	Claim 14435 was found in Data highway system V&V report. It indicates that separation from OMMDP/EDP switch has been claimed to fulfill the corresponding separation requirement.		14378:Ensuring that the optical port from OMMDP/EDP switch to CS switch is NOT furnished	14422:F - V&V report			
14405	Heading	4.3 V&V claims						
14408	V&V claim	See the linked evidence.	Passed			14437:Result of searching Data highway subsystem claims about separation of OMMDP/EDP switch and CS switch 14436:Data highway documentation review result	10736:Transmission of data unidirectional from I&C to adm. computer only	10745:React or I&C system 10746:Data highway system release

Figure 75. An excerpt from the V&V report module of the I&C system.










ID	Artifact Type	Contents	Result	Reports Execution Of	Reports Test/Analysis/Etc...	Is Evidenced By	Reports Fulfilment Of	Judges
14432	Heading	4.2 V&V case execution results						
14433	V&V execution report	It was noticed that there was no fibre cable from OMMDP/EDP switch to CS switch. It was further noticed that the output port of the OMMDP/EDP switch and the input port of the CS switch were blocked by a plug.		 14376: Visual inspection that the optical port from OMMDP/EDP switch to CS switch is NOT furnished	 10747: Data highway system  10749: Cabling release  10748: EthSwitch Company model QW-1200-T-FIBRE			
14434	Heading	4.3 V&V claims						
14435	V&V claim	See the linked evidence.	Passed		 14433: Result of OMMDP/EDP switch output port and CS switch input port inspection	 11966: Transmission of data unidirectional from OMMDP and EPD to CS and PS	 10747: Data highway system  10749: Cabling release  10748: EthSwitch Company model QW-1200-T-FIBRE	

Figure 76. An excerpt from the V&V report module of the Data highway system.

The traces of claim 14408 presented in **Figure 75** are presented in **Figure 77** as a trace diagram.

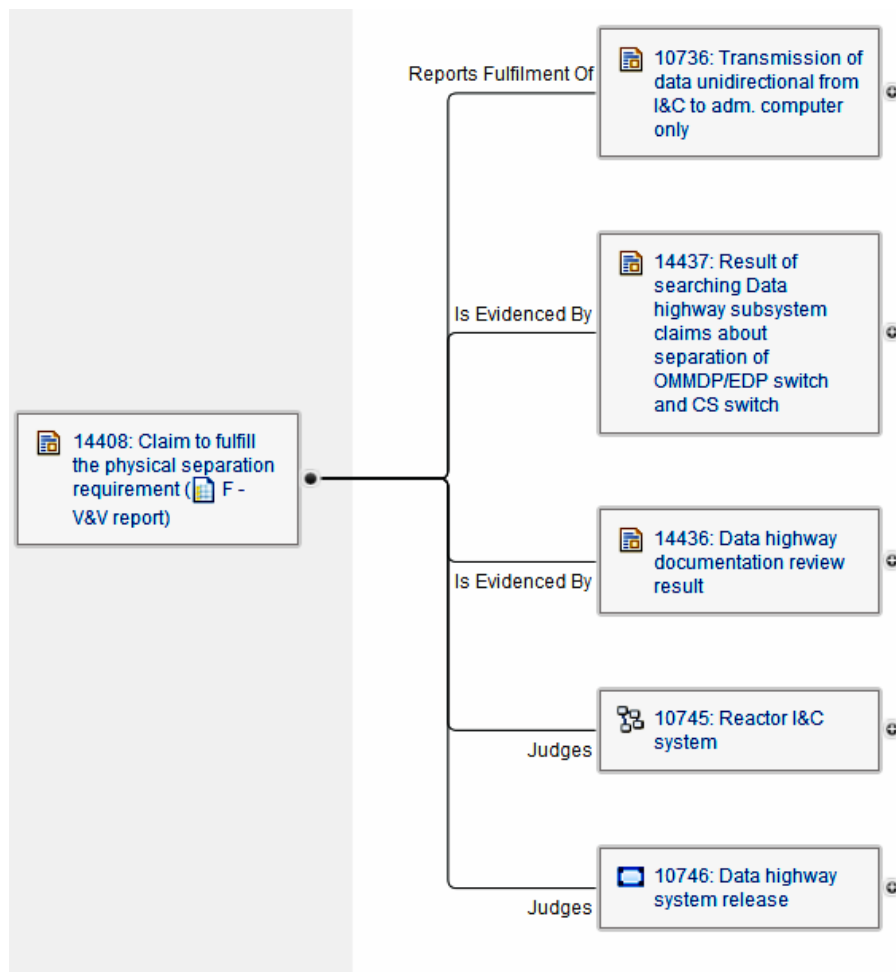


Figure 77. Visualisation of traces in the Doors NG tool.

In all cases, the trace is marked suspect if one of the artefacts linked by the trace is updated. This provides the main feature needed for impact analysis. The demonstration proves that implementation of the concept models, and the consequent TIMs, presented in Chapter 9 is rather straightforward using a tool that provides a repository with configurable set of objects, their attributes and relationships, and that provides a traceability feature with impact analysis. It shall be noted, however, that the demonstration is not complete and cannot be used a reference design. Besides these features, version control with baselines is mandatory for successful tracing of engineering artefacts.

11. Summary and conclusions

The purpose of this publication is to provide a basis for disciplined Requirements Engineering in nuclear power plant automation. This overall goal brings along several challenges. Poor requirements are still a source of problems even if lots of research has been carried out and numerous textbooks and training are available. This is probably due to the uncertainties during the early stages of design and the need for abstract thinking and communication with many stakeholders in requirements definition. Design of instrumentation and control systems, in particular, has a multidisciplinary character with links to other technical domains, human users, safety assessment and regulatory practices. Since design is iterative and limited by many practical constraints, requirements can't be treated in isolation but as an integral part of system descriptions. Consequently, the definition and management of requirements must be seen in the wider context of Systems Engineering and instrumentation and control systems as part of a nuclear power plant. For successful communication all participating stakeholders should have a shared vocabulary and a common, rational way of thinking about systems and their life-cycles processes. Computer tools and information models are necessary for efficient management and exchange of all the design information. This creates a further challenge, since describing a complex system unambiguously and exhaustively is difficult both in theory and in practice, as can be seen from the long development times and massiveness of many international standards.

For these reasons this publication is not limited to requirements and control systems alone but tries to present a generic model of the concepts needed for describing systems and their life-cycles. This work is expected to have impacts on two time scales. In the short term the discussions and the glossary in this publication is aims to clarify the somewhat confusing terminology currently used in the nuclear community. In a more distant future the conceptual model hopefully gives industrial companies ideas for developing systems engineering tools and data models of exchanging information in the supply chain and regulatory communication. Corresponding to these two goals this publication is divided into two parts, Part I defining the modelling concepts in general and Part II presenting a more practical information model for tool development. Both parts cover topics that are relevant to the overall goals of the publication, such as functions and physical structure of systems, life-cycle models, safety aspects, requirements and traceability relationships.

Part I discusses informally the principles of modelling nuclear power plants. One aim is to clarify the terminology in a way that supports communication and documentation in the current design practice. Unfortunately, the terminology found in existing standards and guidelines does not provide easy solutions. The definitions are often vague, and there are inconsistencies between different sources. Therefore, some new terms and interpretations of old ones are introduced. The target is on requirements definition and management and especially on the safety of instrumentation and control systems. Requirements are, however, not be considered in isolation from other kinds of design data or systems at a nuclear power plant. In a similar fashion, re-

requirements engineering is tightly coupled to other life-cycle activities. So, Part I discusses power plant functions, systems, structures and their properties, let them be required the various stakeholders, specified by the developers or actually present at an existing power plant. The terms defined are understood as elements of a system model and a project model stored in an electronic repository. All these model elements express statements originating from various stakeholders, including designers, and have the purpose of communicating relevant information between them throughout the system life-cycle. Requirements are statements having a communicative intent different from other kinds of statements such as user's needs, known facts, assumptions or organisation's intentions. We have tried to clarify their interpretations and ways to express the unambiguously, either in textual or graphical form. This further underlines the need to consider requirements as an integral part of system models and engineering processes.

For future computer tools to work, the concepts used for system modelling should be consistent and based on a sound philosophy. Laying the general foundations is the goal of Part I of our publication. Based on previous experiences collected in machine automation Part II takes a more technical viewpoint and suggests an information model that has a more limited scope but can be used as a basis for practical software tools. The long-term vision is in computer-supported, model-based engineering and exchange of design data. Potential solutions can be sought for in recent developments in model-based system development, semantic web technologies, integrated engineering environments and international standardisation of product data models. In practice however, existing nuclear power plants have large amounts of traditional documents and drawings, and electronic documents will be needed also in the years to come, also when new plants are considered. Therefore, the information model in Part II represents a hybrid, "database-oriented" approach. Some engineering artefacts, such as system elements and requirements, are treated as semiformal model elements and stored in the form of database records. However, also unstructured artefacts such as documents, diagrams and even formal models produced by foreign software tools can be managed. Status data and traceability links are maintained for all kinds of engineering artefacts stored in the project data repository. To demonstrate the applicability of the information model for maintaining traceability, Part II describes a small implementation built upon a commercial requirements management system.

Remembering the challenges listed above it is clear that the proposals made in this publication can't be final or complete. Instead they should be understood as an attempt to adjust the extensive research and standardisation carried out in many other areas of industry to the practices of nuclear power plant engineering and regulation. One conclusion to be made from the insights collected is that the nuclear domain would probably benefit from a more structured terminology and design and licensing processes. Hopefully this publication gives useful inputs to discussion among and within various organisations.

References

- Alanen, J., Vidberg, I., Nikula, H., Papakonstantinou, N., Pirttioja, T. & Sierla, S. 2011. Engineering data model for machine automation systems. VTT Tiedotteita 2583. Espoo, Valtion Teknillinen Tutkimuskeskus, 131 p. <http://www.vtt.fi/inf/pdf/tiedotteet/2011/T2583.pdf>
- Authén, S. & Holmberg, J. 2013, Guidelines for reliability analysis of digital systems in PSA context – Phase 3 Status Report, Nordisk kernesikkerhedsforskning, NKS Secreteriat.
- Authén, S., Holmberg, J.-E., Lanner, L. & Tyrväinen, T. 2014. Guidelines for reliability analysis of digital systems in PSA context – Phase 4 Status Report, NKS-302, Nordic nuclear safety research (NKS), Roskilde.
- Booch, G., Rumbaugh, J. & Jacobson, I. 1999. The Unified Modeling Language User Guide. Reading, Massachusetts, Addison Wesley Longman Inc., 482 p.
- Chambers, C., Croll, P. & Bowell, M. 1999. A study of incidents involving programmable electronic safety-related systems. *Interacting with Computers* 11 (1999), pp. 597–609.
- Chandrasekaran, B. & Josephson, J. 2000. Function in Device Representation. *Engineering with Computers*, Vol. 16, Numbers 3–4, p. 162–177.
- Cockburn, A. 2001. *Writing effective use cases*. Addison-Wesley, 270 p.
- Fanmuy, G., Fraga, A. & Llorens, J. 2011. Requirements Verification in the Industry. *Proceedings of the Second International Conference on Complex Systems Design & Management, CSDM 2011*.
- Friedental, S., Griego, R. & Simpson, M. 2007. INCOSE Model Based Systems Engineering (MBSE) Initiative. INCOSE2007, San Diego, June 24–29 2007. A presentation.
- Garcia, M. & Fleisher, L. 2010. Systems Engineering Cost Control using CONOPS and introducing a process for “WILLS” vs. “SHALLS”. ASNE Day 2010 “Engineering the Affordable Global Navy Through Innovation”, April 8–9, 2010, Hyatt Regency Crystal City, Arlington, VA.
- Gotel, O. et al. 2012. The Quest for Ubiquity: A Roadmap for Software and Systems Traceability Research. The 20th IEEE International Requirements Engineering Conference, September 24th–28th, 2012. Chicago, Illinois, USA. 10 p.
- Hedberg, H. and Wang, Y. 2001. PALBUS Work Package 10.10.2001. Methods for Verification and Validation of Distributed Control Systems. Borås: SP Swedish National Testing and Research Institute. 66 p.
- HSE 2003. Out of control: Why control systems go wrong and how to prevent failure. The Health & Safety Executive (HSE), <http://www.hse.gov.uk/pubns/books/hsg238.htm>.
- Hubka, V. & Eder, W. 1988. *Theory of technical systems: A total concept theory for engineering design*. Berlin, Springer-Verlag, 275 p.
- Hull, E., Jackson, K. & Dick, J. 2002. *Requirements engineering*. London, Springer-Verlag, 213 p.

- IAEA 2002. Harmonization of the licensing process for digital instrumentation and control systems in nuclear power plants. International Atomic Energy Agency, IAEA-TECDOC-1327, 123 p.
- IAEA 2004. Format and Content of the Safety Analysis Report for Nuclear Power Plants. Vienna, International Atomic Energy Agency (IAEA), safety guide no. GS-G-4.1, 91 p.
- IAEA 2005. Assessment of Defence in Depth for Nuclear Power Plants. IAEA Safety Reports Series no. 46.
- IAEA 2006. Decommissioning of facilities using radioactive material. IAEA safety standards series No. WS-R-5, 38 p.
- IAEA 2007. IAEA safety glossary – Terminology used in nuclear safety and radiation protection. 2007 edition, 238 p.
- IDEF0 1993. Integration definition for function modelling (IDEF0). Draft Federal Information Processing Standards Publication 183, 116 p., <http://www.idef.com/pdf/idef0.pdf>.
- IEC 60848 2002. GRAFCET specification language for sequential function charts.
- IEC 60964 2007. Nuclear Power Plants – Control Rooms – Design. Committee draft for vote 2007-03-02, 38 p.
- IEC 61131-3 2013. Programmable controllers – Part 3: Programming languages, Ed. 3.0, Feb 20th, 2013.
- IEC 61508-1 2010. Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements. Edition 2.0 (2010-04-30). Geneva: International Electrotechnical Commission.
- IEC 61226 2009. Nuclear power plants – Instrumentation and control important to safety – Classification of instrumentation and control functions. Ed. 3.0, 36 p.
- IEC 61355 2008. Classification and designation of documents for plants, systems and equipment – Part 1: Rules and classification tables. Edition 2.0. Geneva, International Electrotechnical Commission, 85 p.
- IEC 61508-1 2010. Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements. 2nd edition. Geneva, International Electrotechnical Commission, 127 p.
- IEC 61513 2011. Nuclear power plants – Instrumentation and control important to safety – General requirements for systems. Ed. 2, 104 p.
- IEC 61784-3 2007. Industrial Communications – Fieldbus Profile – Part 3: Profiles for functional safety communications in industrial networks. 2007-12 ed. Geneva: International Electrotechnical Commission. 47 p.
- IEC 61987-10 2009. Industrial-process measurement and control – Data structures and elements in process equipment catalogues – Part 10: Lists of Properties (LOPs) for Industrial-Process Measurement and Control for Electronic Data Exchange – Fundamentals. Final draft international standard 2009-05-08, 49 p.

- IEC 62337 2011. Commissioning of electrical, instrumentation and control systems in the process industry – Specific phases and milestones. Ed. 2, final draft international standard 2011-10-28, 36 p.
- IEC 81346-1 2007. Industrial systems, installations and equipment and industrial products – Structuring principles and reference designation – Part 1: Basic rules. Committee draft 2007-03-30, 60 p.
- IEEE Std 1362 1998. IEEE guide for information technology – System definition – Concept of Operations (ConOps) document. New York, The Institute of Electrical and Electronics Engineers, 21 p.
- IFC 2011. Industry Foundation Classes IFC4, release candidate 3, available at <http://buildingSMART-tech.org/ifc/IFC2x4/rc3/html/index.htm>.
- INCOSE 2007. INCOSE Systems engineering handbook – A guide for system life cycle processes and activities. San Diego, CA, International Council on Systems Engineering, 304 p.
- ISA 2013. Machine and Unit States: An Implementation Example of ANSI/ISA 88.00.01. ISA Technical Report dTR88.00.02, 85 p.
- ISO 10303-233. 2012. Industrial automation systems and integration – Product data representation and exchange – Part 233: Application protocol: Systems engineering. Geneva, International Organization for Standardization (ISO), 800 p.
- ISO 10628 1997. Flow diagrams for process plants – General rules. Geneva, International Organization for Standardization.
- ISO 12100. 2010. Safety of machinery – General principles for design – Risk assessment and risk reduction. Edition 1. Geneva, International Organisation for Standardization, 77 p.
- ISO 15926-2 2003. Industrial automation systems and integration – Integration of life-cycle data for process plants including oil and gas production facilities – Part 2: Data model. 241 p.
- ISO/IEC/IEEE 15288 2015. Systems and software engineering – System life cycle processes. Edition 1. Geneva: International Organization for Standardization, Geneva, International Electrotechnical & New York, Institute of Electrical and Electronics Engineers, 108 p.
- ISO/IEC/IEEE 29148 2011. Systems and software engineering – Life cycle processes – Requirements engineering. Final draft international standard, March 2011, 94 p.
- ISO/IEC TR 24748-1 2010. Systems and software engineering – Life cycle management – Part 1: Guide for life cycle management. Technical report, 2010-10-01, 86 p.
- ISO/IEC/IEEE 42010 2011. Systems and software engineering – Architecture description. First edition, 37 p.
- Jureta, I., Mylopoulos, J. & Faulkner, S. 2008. Revisiting the core ontology and problem in requirements engineering. 16th IEEE International Requirements Engineering Conference, 2008.
- Kelly, T.P. 1999. Arguing safety – A systematic approach to managing safety cases. University of York. Thesis, Department of Computer Science Green Report YCST 99/05.

- Kotonya, G. & Sommerville, I. 1998. Requirements engineering: processes and techniques. Chichester, John Wiley & Sons, 400 p.
- Lind, M. 1999. Making sense of the abstraction hierarchy. CSAPC'99, Villeneuve d'Ascq, France 21–24 September 1999, 6 p.
- Lind, M. 2005. Modeling Goals and Functions of Control and Safety Systems – theoretical foundations and extensions of MFM. Nordic nuclear safety research, report NKS-114, 45 p.
- Lintern, G. 2006. Foundational issues for work domain analysis. Proceedings of the Human Factors and Ergonomics Society 50th annual meeting 2006, 5 p.
- Löffelmann, G., Polke, B. & Zgorzelski, P. 2007. PROLIST Lists of Properties – an important step toward an integrated electronic engineering and business workflow. *Atp international* 5 (2007) No. 1, pp. 34–40.
- Mäder, P. & Gotel, O. 2011. Towards automated traceability maintenance. *J. Syst. Software* (2011), 23 p.
- Mäder, P., Gotel, O. & Philippow, I. 2009a. Getting Back to Basics: Promoting the Use of a Traceability Information Model in Practice. In: Proc. 5th Int'l Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE2009), Vancouver, Canada, 5 p.
- Mäder, P., Gotel, O. & Philippow, I. 2009b. Enabling automated traceability maintenance through the upkeep of traceability relations. In: Proceedings 5th European Conference on Model-Driven Architecture Foundations and Applications (ECMDA2009), Enschede, Netherlands, 16 p.
- Malm, T. & Kivipuro, M. 2004. Safety-related control systems in machinery. Examples (in Finnish). VTT Research Notes 2264. Espoo, VTT, 90 p. + app. 4 p. <http://www.vtt.fi/inf/pdf/tiedotteet/2004/T2264.pdf>
- Mavin, A., Wilkinson, P., Harwood, A. & Novak, M. 2009. EARS (Easy Approach to Requirements Syntax). 17th IEEE International Requirements Engineering Conference, 6 p.
- MSR Consortium 2002. MSRSYS – ECU control system description. Association for Standardisation of Automation and Measuring Systems (ASAM e.V.) [Online]. Available at: <http://www.msr-wg.de/medoc/download/msrsys/v120a/msrsys-sp-en/msrsys-sp-en.pdf> [referenced 18.09.2015]
- NRC 1997. Digital instrumentation and control systems in nuclear power plants: safety and reliability issues, final report. National Research Council (U.S.), Committee on Application of Digital Instrumentation and Control Systems to Nuclear Power Plant Operations and Safety. Washington, D.C., National Academy Press.
- Nejati, S. et al. 2011. A SysML-Based Approach to Traceability Management and Design Slicing in Support of Safety Certification: Framework, Tool Support, and Case Studies. Simula Research Laboratory Oslo, Norway, Preprint submitted to *Information and Software Technology*, 51 p.
- NIST 2006. Capital Facilities Information Handover Guide, Part 1. National Institute of Standards and Technology (NIST), 91 p., <http://fire.nist.gov/bfrlpubs/build05-/PDF/b05037.pdf>

- NORSOK I-005 2005. System control diagram. Standards Norway, NORSOK standard I-005, rev. 2, April 2005.
- Oedewald, P. 2011. Safety management and organisational learning (MANOR). In: Puska, E. K. & Suolainen, V. (Eds.) 2011. SAFIR2010, The Finnish Research Programme on Nuclear Power Plant Safety 2007–2010, Final Report, pp. 27–35. VTT Research Notes 2571. Espoo, VTT. Available at: <http://www.vtt.fi/inf/pdf/tiedotteet/2011/T2571.pdf>.
- OMAC 2006. Packaging Machine Language V3.0, Mode & States Definition Document. OMAC Motion for Packaging Working Group, PackML Subcommittee.
- OMG 2010. OMG Systems Modeling Language (OMG SysML™), version 1.2. Available at <http://www.omg.org/spec/SysML/1.2/> (referenced 23.11.2012), 260 p.
- Pahl, G. & Beitz, W. 1996. Engineering design – A systematic approach. London, Springer-Verlag, 544 p.
- Pihlanko, P. 2013. The use of SysML in the reactor automation renewal project (in Finnish). Aalto University, School of Electrical Engineering, Master's Thesis, 82 p.
- Raatikainen, M., Männistö, T., Tommila, T. & Valkonen, J. 2011. Requirements engineering in nuclear power plant automation (in Finnish). Final report of the VAHAYA project, 60 p.
- Searle, J. & Vanderveken, D. 1985. Speech acts and illocutionary logic. In: Searle, J. & Vanderveken, D. 1985. Foundations of illocutionary logic. Cambridge University Press.
- Shum, S. B. & Hammond, N. 1994. Argumentation-Based Design Rationale: What Use at What Cost? International Journal of Human-Computer Studies, 1994, 40 (4), 603–652.
- Sivertsen, T. et al. 2005. The TACO Approach for Traceability and Communication of Requirements. In: R. Winther, B.A. Gran, and G. Dahll (Eds.): SAFECOMP 2005, LNCS 3688, pp. 317–329. Berlin Heidelberg, Springer-Verlag, 13 p.
- Stahl, T. & Völter, M. 2005. Model-driven software development – Technology, engineering, management. John Wiley & Sons, Ltd, 428 p.
- Tang, A. 2007. A Rationale-based Model for Architecture Design Reasoning. Available at: <http://www.ict.swin.edu.au/personal/atang/>
- Tommila, T., Hirvonen, J., Jaakkola, L., Peltoniemi, J., Peltola, J., Sierla, S. & Koskinen, K. 2005. Next generation of industrial automation. Concepts and architecture of a component-based control system. VTT Research Notes 2303. Espoo, VTT Technical Research Centre of Finland, 104 p. Available at: <http://www.vtt.fi/inf/pdf/tiedotteet/2005/T2303.pdf>.
- Tommila, T., Valkonen, J., Parviainen, P. & Raatikainen, M. 2011. Requirements definition and management in critical automation applications. Proceedings of the Automation XIX seminar, Helsinki 15.–16.3.2011, Finnish Society of Automation, 6 p.
- Tommila, T. 2013. On modelling traceability in requirements engineering. A working report of the SAREMAN project, 25 p.
- Tommila, T., Laarni, J. & Savioja, P. 2013. Concept of operations (ConOps) in the design of nuclear power plant instrumentation & control systems. A working report of the SAREMAN project, 68 p.

- Tommila, T. & Pakonen, A. 2014. Controlled natural language requirements in the design and analysis of safety critical I&C systems. Research report VTT-R-01067-14. Espoo, VTT, 56 p. + app. 8p. Available at: <http://www.vtt.fi/inf/julkaisut/muut/2014/VTT-R-01067-14.pdf>.
- Tommila, T., Savioja P. & Valkonen, J. 2014. Role of requirements in safety demonstrations. A working report of the SAREMAN project, 49 p.
- Vicente, K. J. 1999. Cognitive work analysis – Toward safe, productive and healthy computer-based work. Mahwah, NJ, Lawrence Erlbaum & Associates, 392 p.
- West, M. 2011. Developing high quality data models. Burlington, Morgan Kaufmann Publishers, 389 p.
- YVL A.6 2013. Conduct of operations at a nuclear power plant. Radiation and Nuclear Safety Authority (STUK), 15 November 2013, 14 p.
- YVL B.1 2013. Safety design of a nuclear power plant. Radiation and Nuclear Safety Authority (STUK), 15 November 2013, 46 p.

Appendix A: Acronyms and abbreviations

BIM	Building Information Model
BWR	Boiling Water Reactor
ConOps	Concept of Operations
COTS	Commercial Off-The-Shelf
DCS	Distributed Control System
EPC	Engineering, Procurement and Construction
EPRI	Electric Power Research Institute
EUC	Equipment Under Control
ETA	Event Tree Analysis
FBD	Function Block Diagram
FMECA	Failure Mode, Effects and Criticality Analysis
FTA	Fault Tree Analysis
HAZOP	Hazard And Operability Study
HFE	Human Factors Engineering
HSI	Human-System Interface
IAEA	International Atomic Energy Agency
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFC	Industry Foundation Classes
I&C	Instrumentation & Control
I/O	Input/output
ISO	International Organization for Standardization
MDSD	Model-Driven Software Development
MBSE	Model-Based Systems Engineering
NIST	National Institute of Standards and Technology
NPP	Nuclear Power Plant
O&M	Operations & Maintenance
OHA	Operating Hazard Analysis
OLC	Operational Limits and Conditions.
PHA	Preliminary Hazard Analysis
PLC	Programmable Logic Controller
PM	Project Management
PWR	Pressurized Water Reactor
RE	Requirements Engineering
SE	Systems Engineering
SFC	Sequential Function Chart
SRS	Software Requirements Specification
StRS	Stakeholder Requirements Specification

SyRS	System Requirements Specification
UML	Unified Modeling Language
URS	User Requirements Specification
V&V	Verification & Validation

Appendix B: Glossary

This appendix contains in alphabetical order definitions of the most important terms used in this publication, primarily in Part I. The specific artefact types used in Part II are explained in Appendix C.

The definitions have been adopted from relevant standards and guidelines whenever appropriate (see references in the end of Part II). Some of them have, however, been given a new meaning consistent with our approach. These definitions are marked with an asterisk (*). In addition to the definitions, notes are used to give examples or to refer to other definitions for comparison.

The reader should be aware of that our ultimate goal is to serve a computer-assisted and model-driven design process in the future. This means that many of the definitions actually concern *model elements* stored in a design database. So, a definition of the type “x is” should in most cases be read as “x is a model element that represents...”. In addition to model elements, some of the definitions are just intended to clarify the jargon currently used in the nuclear domain.

Action (toimenpide): An atomic operation performed on a target object, for example, opening a valve. An action is part of a *task* or an *activity*. (*)

Note: An action is usually understood to be performed at a given point of time. However, we also need actions that are continuous or performed cyclically, for example scanning the process inputs of an I&C system.

Activity (aktiviteetti): An activity is something happening or changing. In this report activity is understood as goal-oriented transformation of inputs to outputs in which various physical actors such as system can participate (freely adapted from ISO 15926-2 2003). As a model element an activity describes what should be accomplished without considering the specific resources that will be used. An activity can be divided into lower-level activities, *tasks* and *actions*. (*)

Note: In many cases, the term process can be used instead of activity, e.g. “business process”, “design process” or “energy production process”

Note: According to IAEA (2007) activities include the production, use, import and export of radiation sources for industrial, research and medical purposes; the transport of radioactive material; the decommissioning of facilities; radioactive waste management activities such as the discharge of effluents; and some aspects of the remediation of sites affected by residues from past activities.

Area (alue): A two or three dimensional *space* defined with respect to structural elements. (*)

Baseline (vaihetuote): Specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures (ISO/IEC/IEEE 29148 2011).

Capability (kyky): Feature or set of features that make a *system* useful for its intended purposes. A capability may, for example, be based on structure, physical properties or *functions* of the system. (*)

Commissioning (käyttöönotto): The process by means of which systems and components of facilities and activities, having been constructed, are made operational and verified to be in accordance with the design and to have met the required performance criteria (IEC 61513 2009).

Communicative intent: Attribute of a *statement* indicating whether it should be interpreted as a fact, problem, requirement, plan, etc. (*)

Component (komponentti): Smallest *system element* that is not further divided into parts for the purposes of the on-going system modelling. (*)

Note: When looked from a different perspective, also components can have parts. For example, a maintainer can replace faulted piece of electronics on a printed circuit.

Note: A component may be hardware or software or a combination of both. The reference definition in IAEA Safety glossary only applies this term to hardware (IEC 61513 2009).

Concept (konsepti): High-level description of how a problem will be (or has been) in principle solved. (*)

Note: A concept may define, for example, the main functions, physical principles and realisation (embodiment) of a system. On a very high level, combustion and nuclear power are alternative physical principles of generating energy, and nuclear reactor is the typical embodiment of nuclear power. For data communication, optics and electrical signals are potential physical principles while laser, optical fibre and copper wire corresponding implementations.

Note: *Concept of operations*, automation concept and alarming concept are examples of concept.

Concept of operations (operointikonsepti, toimintakonsepti): A high-level description of how the elements of the system and its operating environment (*context*) communicate and collaborate in order to achieve the stated goals of the system. (*)

Note: ISO/IEC/IEEE 29148 (2011) defines ConOps as a verbal and graphic statement, in broad outline, of an organisation's assumptions or intent in regard to an operation or series of operations. The concept of operations frequently is embodied in long-range strategic plans and annual operational plans. In the latter case, the concept of operations in the plan covers a series of connected operations to be carried out simultaneously or in succession. The concept is designed to give an overall picture of the organisation operations. It provides the basis for bounding the operating space, system capabilities, interfaces and operating environment.

Conceptual design (periaatesuunnittelu): *Design* activity producing alternative, high-level solutions to satisfy specified goals and requirements, including the analysis to determine the feasibility of each alternative. Conceptual design determines the principle of a solution, i.e. the *con-*

cept. Conceptual design is closely related and, due to the need for iteration, overlapping with *requirements definition*. (*)

Configuration item (hallinta-alkio): Set of *model elements* and other design information to which the rules of *configuration management* are applied. (*)

Configuration management (konfiguraationhallinta): The process of establishing and maintaining the integrity of all outputs developed by a project or process and making them available to concerned and authorised parties. The activities of configuration management include identification, version control, tracing, auditing and status accounting of all design information. (*)

Note: This report focuses on design information, but configuration management can be also applied to an existing, real-world system, such as a power plant.

Note: As defined by IAEA (2007), configuration management is the process of identifying and documenting the characteristics of a facility's structures, systems and components (including computer systems and software), and of ensuring that changes to these characteristics are properly developed, assessed, approved, issued, implemented, verified, recorded and incorporated into the facility documentation.

Constraint (rajoitus): Externally imposed limitation on system requirements, design, or implementation or on the process used to develop or modify a system. Constraints are *requirements* that restrict the design solution or implementation of the systems engineering process (ISO/IEC 29148 2011).

Note: Examples of constraints include: interfaces to existing systems where the interface cannot be changed; physical size limitations; laws of a particular country; pre-existing technology platforms; and operator capabilities and limitations (ISO/IEC/IEEE 29148 2011).

Note: In this document, a statement's communicative intent "constraint" is used to indicate a requirement that intentionally limits the freedom that the designer would otherwise have. For example, a pre-existing technology platform can be understood as a "fact", while the requirement to use that same platform is a "constraint". Unnecessary constraints should be avoided as they may exclude potentially optimal solutions from the design options.

Context (ympäristö): The interrelated conditions in which something exists or occurs (www.merriam-webster.com/dictionary/context). The context of a system includes, for example, the users and external systems the system interacts with, as well as the *operational environment* where it is located. Here, context also includes more abstract entities to be considered, such as cultural issues, practices and terminologies.

Control task (hallintatehtävä): A *task* required for monitoring and controlling an industrial *process* and process equipment. (*)

Derived requirement (johdettu vaatimus): *Requirement* deduced or inferred from the collection and organisation of requirements into a particular system configuration and solution (ISO/IEC/IEEE 29148 2011).

Design (suunnittelu): The process (and the result) of developing a concept, detailed specifications and supporting calculations for a facility and its parts (modified from IAEA 2007).

Note: This definition includes the synthesis of solutions but excludes their analysis (e.g. testing), which is another mode of engineering. Neither does it mention the purpose of the results, the construction, operation and maintenance of the facility.

Design bases (suunnitteluperusteet): Set of *statements* (each called a “design basis”) that form the essential input information and acceptance criteria for the *design* of a system. (*)

Note: The design bases of a nuclear plant or plant system may include, for example, postulated initiating events to be handled, laws and regulations to be applied, system requirements and possible design constraints.

Note: As stated by NRC in 10 CFR 60.2 (1), design bases means that information that identifies the specific functions to be performed by a structure, system, or component of a facility and the specific values or ranges of values chosen for controlling parameters as reference bounds for design. These values may be restraints derived from generally accepted "state-of-the-art" practices for achieving functional goals or requirements derived from analysis (based on calculation or experiments) of the effects of a postulated event under which a structure, system, or component must meet its functional goals.

Document (dokumentti): Uniquely identified unit of information for human use, such as a report, specification, manual or book in printed or electronic form (ISO/IEC/IEEE 29148 2011).

Engineering process requirement (prosessivaatimus): A *requirement* imposed on the development activities, development organisations and methods and tools used, e.g. mandating a particular design method or documentation practice. (*)

Note: This longer term is used for the reason that the expression “process requirement” might be confused with requirements originating from the power generation process or the process equipment.

Function (toiminto): A *capability* of a *system* to generate one or more intended behaviours of the system and/or effects on the system environment. (*)

Note: Specific purpose or objective to be accomplished, that can be specified or described without reference to the physical means of achieving it (IEC 60964 2007, IEC 61226 2009).

Note: In the system model a function is a model element that can be used to describe the required, specified or actual behaviour of the system. A function is an abstraction since there exists no real-world entity that directly corresponds to it. Instead, there are devices, software components and structures that behave as defined by functions.

Note: A function is specific to a system. If the system is deleted from the model also the associated system functions should be deleted. This is different from activities which can be defined independently of any systems possibly used to carry out them.

Note: The term function has a dual role in the design process. It can represent an internal capability, such as a safety function of a nuclear power plant, not allocated to any internal system elements. On the other hand, a function can describe a “service” that is externally available to accomplish various activities. In this sense, for example “production of electricity” is one function of a power plant. Or, “reactor temperature measurement” is a function provided by a field instrument to the rest of the I&C system.

Functional requirement (toiminnallinen vaatimus): *Requirement* that describes *activities* to be accomplished, *functions* to be provided or behaviours to be exhibited by a *system* or system element (modified from ISO/IEC/IEEE 29148 2011).

Goal (tavoite): A desired or required state-of-affairs that should be achieved or maintained. (*)

Note: In this document, “goal” is also used as a value of the intent attribute of a statement indicating a desired but not mandatory objective for the design.

Hazard (vaara, uhka): Event having the potential to cause injury to plant personnel or damage to components, equipment or structures. Hazards are divided into internal hazards and external hazards. Fire, flooding, steam-line break and earthquake are examples of hazards (IEC 61513 2011).

Note: A hazard, identified during the safety analysis process, is associated with one or more elements in the system model, for example with process component, plant function or structural element. Hazards are usually the rationale behind safety requirements and should, therefore, be logged as part of the system model.

I&C architecture (automaatioarkkitehtuuri): Organisational structure of the *I&C systems* of the plant (modified from IEC 61513 2011).

Note: The architecture of a system can be described from different viewpoints. So, we can speak, for example, about functional, hardware and software architectures.

Note: In IEC 61513 the term designates only the systems important to safety in the whole I&C architecture of the plant. Here it's a given a broader scope including the operational automation as well.

I&C function (automaatio toiminto): *Function* to control, operate and/or monitor a defined part of the process (IEC 61513 2011).

Note: The term can be used by process engineers to structure the functional requirements for the I&C (IEC 61513 2011).

I&C system (automaatiojärjestelmä): *System*, based on electrical and/or electronic and/or programmable electronic technology, performing *I&C functions* as well as service and monitoring functions related to the operation of the system itself (IEC 61513 2009).

Note: In general, the boundaries of a system are specific to an application. Here, the term I&C system encompasses all elements of the system such as internal power supplies, sensors and other input devices, data highways and other communication paths, and interfaces to actuators and other output devices (IEC 61513 2011). So, sensors and actuators belong to I&C system, while pipes and valves are part of the *process system*. Similarly, display devices and controls form the boundary towards human users.

Note: According to their typical functionality, IAEA (2007) distinguishes between automation and control systems, HMI systems, *interlock systems* and *protection systems*.

Intent: See *communicative intent*.

Interlock (lukitus): A *function* that prevents hazardous conditions from occurring by blocking potentially unsafe actions. (*)

Note: Safety interlocks are designed for protecting people and the environment while device interlocks prevent damage of the equipment.

Job (työ): A set of *tasks* which are operationally related. The tasks within a job should be coherent with regard to required skill, knowledge and responsibility. (IEC 60964 2007)

Note: Here, *tasks* are considered as parts of *activities* and, by definition, unallocated. Therefore, the definition above should be understood as the combination of responsibilities of a worker role with respect to those tasks.

Licensing (lisensointi): The work process to generate and document evidence that certain solutions is acceptable from a safety point of view (from IAEA 2002).

Note: This definition obviously misses the regulatory viewpoint and the need to demonstrate the safety with specific acceptance criteria and systematic argumentation.

Life-cycle model (elinkaarimalli): A framework containing the processes, activities, and tasks involved in the development, operation, and maintenance of a software product, which spans the life of the system from the definition of its requirements to the termination of its use (IEEE Std 1471-2000).

Note: This definition misses people, responsibilities and the methods and tools used. Here, we recommend an extended interpretation.

Life-cycle phase (elinkaarivaihe): A period in a project or existence of a system bounded by two *milestones*. (*)

Location (sijainti): Intended or accomplished *space* (IEC 81346-1 2007).

Milestone (etappi): A point at the end of a *life-cycle phase* where the current state of a project or system is evaluated and decisions are made concerning the next phase. (*)

Model (malli): See *system model*.

Note: IAEA (2007) emphasises the system analysis instead of design by defining model as an analytical representation or quantification of a real system and the ways in which phenomena occur within that system, used to predict or assess the behaviour of the real system under specified (often hypothetical) conditions. A conceptual model is a set of qualitative assumptions used to describe a system (or part thereof).

Model element (mallialkio): An individually considered and maintained piece of information in the *system model*. A model element can contain other model elements. (*)

Non-functional requirement (ei-toiminnallinen vaatimus): *Requirement* that specifies system properties or conditions under which the system is required to operate or exist. It defines how a system is supposed to be. Performance requirements and human factors attributes are examples of this type (modified from ISO/IEC/IEEE 29148 2011).

Operating mode (toimintamoodi): One in a set of discrete alternative states that determines how a *system* functions, for example “manual” or “automatic”. (*)

Operational environment (toimintaympäristö): Physical spaces, areas and structures within which systems and people operate. (*)

Note: See also *context*.

Note: For example, distances and environmental conditions are attributes of the operational environment that are relevant for requirements definition and system design.

Operational limits and conditions (turvallisuustekniset käyttöehdot): A set of rules setting forth parameter limits, the functional capability and the performance levels of equipment and personnel approved by the regulatory body for safe operation of an authorised facility (IAEA 2004).

Note: The Operational Limits and Conditions (OLC) are a major source of requirements. According to YVL A.6 (2013) they state the technical and administrative requirements covering for example, process parameter limits; limits for the activation of protection systems; requirements for safety systems; maintenance, testing, inspection and surveillance programmes; instructions; and grounds for the requirements specified above.

Operational state (käyttötila): A region of states in the state space of a *system* considered equivalent from some point of view. (*)

Note: "Normal operation", "starting up" and "stopped" are examples of operational states. Sometimes the current operational state can be automatically deduced from the values of state variables. In more complex cases the current state is declared by a managerial decision.

Note: In IAEA (2007) operational states are defined only under normal operation and anticipated operational occurrences. In addition there are accident conditions, i.e. deviations from normal operation more severe than anticipated operational occurrences, including design basis accidents and severe accidents.

Performance requirement (suorituskykyvaatimus): A *requirement* that defines the extent or how well, and under what conditions, a function or task is to be performed (adapted from ISO/IEC/IEEE 29148 2011).

Process (prosessi): In the context of an industrial plant, process is a sequence of chemical, physical or biological operations (i.e. an *activity*) for the conversion, transport or storage of material or energy (ISO 10628 1997).

Note: IAEA (2007): 1. A course of action or proceeding, especially a series of progressive stages in the manufacture of a product or some other operation. 2. A set of interrelated or interacting activities that transforms inputs into outputs.

Note: In addition, the term process is often seen in the context of commercial endeavours. A "business process" can be defined as a collection of related, structured activities that produce a service or product for a customer or customers.

Process plant (prosessilaitos): Facilities and structures necessary for performing a *process* (ISO 10628 1997).

Process system (prosessijärjestelmä): A *system* primarily consisting of process equipment. (*)

Project organisation (projektiorganisaatio): Organisation(s) or individuals that have responsibility during the phases of the overall safety life cycle of the I&C and/or during the phases of the safety life cycles of the *I&C systems*, to define and perform all management and technical *activities* concerning the *I&C functions*, systems and equipment important to safety. This term is to be contrasted with "operating organisation" (IEC 61513 2009).

Protection (suojaus): A *function* that actively initiates actions to prevent an unsafe or potentially unsafe condition. (*)

Protection system (suojausjärjestelmä): System, which monitors the operation of a equipment under control, for example a nuclear reactor, and which, on sensing an abnormal condition, automatically initiates actions to prevent an unsafe or potentially unsafe condition (modified from IAEA 2002).

Purpose (tarkoitus): The purpose of a system refers to the *activities* or *goals* it is designed for by the developer or used for by the end-user. (*)

Qualification (pätevöinti, kvalifiointi): The process of determining by comparison to the criteria set by the society whether an organisation, person or technical system is suitable for its intended tasks. (*)

Note: *Verification* and *validation* are usually performed by the customer and supplier against application-specific criteria, such as a written specification or user's needs. From a regulatory viewpoint, an external point of reference might be appropriate. Qualification can be defined as an activity that uses objective evidence to confirm that a specific component, system or organisation is able to achieve all requirements imposed by standards and regulations.

Note: IAEA (2007) defines this term as the generation and maintenance of evidence to ensure that equipment will operate on demand, under specified service conditions, to meet system performance requirements.

Note: IEC 61513 (2011) defines qualification as the process of determining whether a system or component is suitable for operational use. Qualification is performed in the context of a specific class of the I&C system and a specific set of qualification requirements. Qualification is always a plant- and application -specific activity. However, it may rely to a large degree on qualification activities performed outside the framework of a specific plant design (these are called "generic qualification" or "pre-qualification").

Requirement (vaatimus): A *statement* of a condition or capability that shall be met or possessed by a system to satisfy a contract, standard, specification, or other formally imposed document. (*)

Note: More traditional definitions: (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability. (IEEE-610.12- 1990).

Note: ISO/IEC/IEEE 29148 (2011) says that requirement is a statement which translates or expresses a need and its associated constraints and conditions. Requirements exist at different tiers and express the need in high-level form (e.g. software component requirement).

Note: IEC 61513 (2011) defines requirement as an expression in the content of a document conveying criteria to be fulfilled if compliance with the document is to be claimed and from which no deviation is permitted. The following types of requirement can be distinguished:

- Safety requirements – Requirements imposed by authorities on the safety of the NPP in terms of impact on individuals, society and environment during the NPP lifecycle.
- Functional and performance requirements – Functional requirements state what actions the system must take in response to specific signals or conditions, and performance requirements define features such as response times and accuracy.
- Operational requirements – Requirements on the operational capacity and ability of the plant imposed by the owner.
- Plant design requirements – Technical requirements on plant general design for the fulfilment of the safety requirements and operational requirements on the plant.

- System design requirements – Design requirements on individual systems to give a design of the complete plant fulfilling the plant design requirements.
- Equipment requirements – Requirements on individual equipment for its fulfilment of the demands of the system design.

Requirements elicitation (vaatimusten hankinta): The process through which the stakeholders (e.g. acquirer, users and suppliers) of a *system* discover, review, articulate, understand the *requirements* on the system and its life-cycle processes. (*)

Requirements engineering (vaatimusten käsittely): Combination of *requirements definition* and *requirements management* activities. (*)

Note: ISO/IEC/IEEE 29148 (2011) defines it as an interdisciplinary function that mediates between the domains of the acquirer and supplier to establish and maintain the requirements to be met by the system, software or service of interest.

Note: The translation of the term to Finnish is problematic. In the current practice, the word “vaatimustenhallinta” is often used. It unfortunately conflicts with the direct translation of “requirements management”. Therefore, we use here the neutral expression “vaatimusten käsittely”.

Requirements definition (vaatimustenmäärittely): Activities of eliciting, analysing, documenting (modelling) and validating and verifying the requirements of a system. (*)

Requirements management (vaatimustenhallinta): *Configuration management* applied to requirements. (*)

Note: ISO/IEC/IEEE 29148 (2011) defines it as activities that ensure requirements are identified, documented, maintained, communicated and traced throughout the life cycle of a system, product, or service.

Requirements validation (vaatimusten kelpuus): Confirmation by examination that requirements (individually and as a set) define the right system as intended by the stakeholders (ISO/IEC/IEEE 29148 2011).

Requirements verification (vaatimusten todentaminen): Confirmation by examination that requirements (individually and as a set) are well formed. This means that a requirement or a set of requirements has been reviewed to ensure the characteristics of good requirements are achieved (ISO/IEC/IEEE 29148 2011).

Scenario (skenaario): Description of an imagined sequence of events that includes the interaction of the system, with its environment and users, as well as interaction among its system elements (modified from ISO/IEC/IEEE 29148 2011).

Safety (turvallisuus): The achievement of proper operating conditions, prevention of accidents or mitigation of accident consequences, resulting in protection of site personnel, the public and the environment from undue radiation hazards (IAEA 2002).

Note: This definition is limited to radiation hazards only.

Safety function (turvallisuustoiminto): A specific *function* that must be accomplished by a *system* for safety (modified from IAEA 2007).

Note: According to IAEA (2007) safety function is a specific purpose that must be accomplished for safety. The three main safety functions for nuclear power plants are (a) Control of re-

activity; (b) Cooling of radioactive material; (c) Confinement of radioactive material (IAEA 2007). Moreover, IAEA 1099 (2000) lists 19 more detailed safety functions that may be used for determining whether a structure, system or component performs or contributes to one or more safety functions and for assigning an appropriate gradation of importance.

Note: Here the term function has the flavour of a goal or objective. In addition to safety functions, IAEA (2007) lists three corresponding “safety requirements”: 1) capability to safely shut down the reactor and maintain it in a safe shutdown condition; 2) capability to remove residual heat from the reactor core; and 3) capability to reduce the potential for the release of radioactive material. This leads to the interpretation that safety function can be understood as a required *capability* of a nuclear power plant as a whole, i.e. a *function* and a *functional requirement* that has not been allocated to any plant systems. See also the definition of *function*.

Safety requirement (turvallisuusvaatimus): A requirement relevant for safety. (*)

Space (tila): An area or volume bounded actually or theoretically (IFC 2011).

Specification (määrittely): Document that specifies, in a complete, precise, verifiable manner, the requirements, design, behaviour, or other characteristics of a system or component, and, often, the procedures for determining whether these provisions have been satisfied (IEC 60880, 3.39, according to IEC 61513 2011).

Stakeholder (sidosryhmä): (system) individual, team, organisation, or classes thereof, having an interest in a *system* (ISO/IEC/IEEE 42010 2011). Individual or organisation having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations (ISO/IEC/IEEE 29148 2011).

Note: Stakeholders include, but are not limited to, end users, end user organisations, supporters, developers, producers, trainers, maintainers, disposers, acquirers, customers, operators, supplier organisations, accreditors and regulatory bodies (ISO/IEC/IEEE 29148 2011).

Stakeholder requirement (sidosryhmävaatimus): Requirement that represents the interests of one or more *stakeholders* and is intended to represent their mutual understanding (*).

Stakeholder requirements specification (sidosryhmävaatimukset): Describes the organisation's motivation for why the system is being developed or changed, defines processes and policies/rules under which the system is used and documents the top level requirements from the stakeholder perspective including needs of users/operators/maintainers as derived from the context of use. In a business environment, the Stakeholder Requirements Specification (StRS) describes how the organisation is pursuing new business or changing the current business in order to fit a new business environment, and how to utilise the system as a means to contribute to the business. The description includes, at the organisation level; the organisational environment, goals and objectives, the business model, and the information environment, and, at the business operation level; the business operation model, business operation modes, business operational quality, organisational formation, and concept of the proposed system. The information items of the StRS should be specified by the stakeholders (ISO/IEC/IEEE 29148 2011).

Statement (toteamus): A broad term referring to an expression by a *stakeholder* of a past, current or future state-of-affairs. (*)

Note: Depending on its *communicative intent*, a statement is interpreted as a fact, goal, requirement, claim, etc. Therefore, all *model elements* in a *system model* can also be understood as statements.

Structure (rakenne): A *system element* that mainly consists of passive *components* whose functioning does not depend on an external input such as actuation, mechanical movement or supply of power. (*)

Note: Depending on the viewpoint, a structure can have some active parts also. Therefore, a structure can be considered as a system, and it can also have *functions*.

Note: As stated by IAEA (2007) structures, are the passive elements: buildings, vessels, shielding, etc.

Note: IAEA 1099 (2000) defines an “active component” as a component whose functioning depends on an external input such as actuation, mechanical movement or supply of power. Its counterpart is a “passive component”.

Note: In its common meaning the word “structure” refers to the composition of an entity, in other words, to what the parts of the entity are and how they are related. This interpretation can be found also in this report in the context of describing “system structure” as a counterpart of “system behaviour”.

Sub-something (osa-, ali-): The prefix “sub” is used in this document informally to indicate that an entity is part of a larger entity of the same kind, for example “subsystem” or “subfunction”. (*)

System (järjestelmä): Combination of interacting *system elements* organised to achieve one or more stated purposes (ISO/IEC 15288 2015). In the context of this working report system elements are considered to be real-world (physical) entities like machines, structures, people, software or data.

Note: IAEA (2007): A system comprises several components, assembled in such a way as to perform a specific (active) function. A component is a discrete element of a system. Examples of components are wires, transistors, integrated circuits, motors, relays, solenoids, pipes, fittings, pumps, tanks and valves.

System element: A general-purpose term denoting a part of a system. (*)

System function (järjestelmätoiminto): See *function*.

System model (järjestelmän malli): Structured set of *model elements* describing a *system-of-interest* and its *operational environment* for the purposes of system development, operation and maintenance. (*)

System-of-interest (kohdejärjestelmä): *System* whose life cycle is under consideration (modified from ISO/IEC/IEEE 29148 2011).

System requirement (järjestelmävaatimus): A *requirement* that concerns a *system* or a *system element*, is intended for the developers as a starting point and is therefore expressed in technical terms.

Note: System requirements are usually derived from *stakeholder requirements* and other sources like standards and regulations.

System requirements specification (järjestelmävaatimukset): Structured collection of the requirements (functions, performance, design constraints, and attributes) of the *system* and its

operational environment and external interfaces. The System Requirements Specification (SyRS) identifies the technical specifications for the selected system-of-interest and usability for the envisaged human-system interaction. It defines the high-level system requirements from the domain perspective, along with background information about the overall objectives for the system, its target environment and a statement of the constraints, assumptions and non-functional requirements. It may include conceptual models designed to illustrate the system context, usage scenarios, the principal domain entities, data, information and workflows. SyRS has traditionally been viewed as a document that communicates the requirements of the acquirer to the technical community who will specify and build the system (ISO/IEC/IEEE 29148 2011).

Note: As understood from GAMP 5 (2008, App. D1), the pharmaceutical industry seems to use the term “user requirements specification” in this meaning.

Task (tehtävä): An activity having a limited and well-bounded scope can be called a task, for example a “design task” or “control task”. Some tasks can be described in terms of sequential “subtasks” and individual *actions*, while some others are rather related to an objective, such as product quality. (*)

Note: As a part of an activity, task is not allocated to specific resources. Often tasks are thought to be carried out by humans, but they can also be performed by technical systems.

Traceability (jäljitettävyys): Property of design information (the *system model*) that makes it possible, e.g. by linking model elements, to find out how *model elements* have been derived from external information sources (stakeholders and standards), from previous versions or from other model elements. (*)

Tracing (jäljittäminen): The activity of producing and maintaining traceability information as an integral part of the *system model*. (*)

User (käyttäjä): *Stakeholder* that benefits from a system during its utilisation (modified from ISO/IEC/IEEE 29148 2011).

User requirement (käyttäjävaatimus): *Stakeholder requirement* that represents the interests of one or more system *users*. (*)

User requirements specification (käyttäjävaatimukset): See *Stakeholder requirements specification*.

Validation (kelpuutus, validointi): *Activity* that uses objective evidence to confirm that the requirements which define an intended use or application have been met. Validation is often said to answer the question "Are you building the right system?" This often involves acceptance of fitness for purpose with end users and other stakeholders. Whenever all requirements have been met, a *validated* status is achieved. (*)

Note: IAEA (2007): The process of determining whether a product or service is adequate to perform its intended function satisfactorily.

Verification (todentaminen, verifiointi): *Activity* that uses objective evidence to confirm that stated requirements and specifications imposed at the start of a life-cycle phase have been met. Verification is often said to answer the question "Are you building the system right?" Whenever requirements have been met, a *verified* status is achieved. (*)

Note: IAEA (2007): The process of determining whether the quality or performance of a product or service is as stated, as intended or as required.

Appendix C: Artefact types

In the following, the artefact types illustrated in the SEAModel_{NPP} diagrams in Part II are explained in detail. Note that the descriptions are not concept definitions, but they supply information about the purpose and usage of the artefact types.

<p>Action: An action is an atomic piece of a work task, i.e. it is not decomposed into smaller behavioural elements. An action can be a system action or (human) actor action depending on the behaviour modelling method, system task or system use case modelling. System functions are often identified from the actions.</p>
<p>Actor action: A special case of Action. Actor action is an abstract artefact type; the sequence of actor actions of a system use case is stored in the Action artefacts.</p>
<p>Area: A special case of Space, such as a tank yard.</p>
<p>Audio file: Audio file is a special case of File. An audio file can be of any type. The most important attribute is the link to the actual audio file.</p>
<p>Behaviour: The Behaviour artefact is used to provide different perspectives to system functional architecture. It can be used to provide views to categorised sets of functionality or to completely different functionalities of machines with dual or more usage purposes.</p>
<p>Black box artefact: Contrary to Model element, a black box artefact is an engineering artefact with no or poor knowledge about their internal structure, or which do not have a well-defined structure. Such artefacts are conventional documents and foreign models.</p>
<p>Building element: A special case of Structural element, such as a wall or door.</p>
<p>Chemical environment: A special case of operational environment describing the chemicals, such as salt sprays and acids, the system-of-interest is expected to be in contact with.</p>
<p>Climatic environment: A special case of operational environment describing the temperature range, temperature change rate and moisture of the operational environment of the system-of-interest.</p>
<p>Component: A non-atomic system element, in some cases denoted as 'part'. Is often distinguished from a subsystem by the fact that a component has a part number whereas subsystem does not. However, a component can have several specialisations, such as information item, structural element, software component, joint element, device and person.</p>

<p>Composite flow: A special case of a flow where the flow consists of several primitive flows. An example is a quadrature encoder the output of which consists of the two pulse train channels, channel A and channel B, with 90 degrees phase shift to each other; the sign of the phase shift reports the direction of the rotation whereas the pulses report the rate of travel; hence the actual flow (the composite flow), rotation travel with its sign (direction), is derived from the two channels.</p>
<p>Constraint: A special case of a requirement. In one sense, it is not a requirement, because it only states facts e.g. about the system's environment (such as the dimensions of the space where the system will be installed); on the other hand, its effect in the design is similar to that of a requirement.</p>
<p>Context description: The Context description artefact provides description e.g. about the following issues requested by ISO 12100:2010: ergonomic principles, energy sources, space limits, life limit, service intervals, other time limits, housekeeping policy, material properties, other limits, external systems interaction and experience of use.</p>
<p>Context model: A special case of Model for modelling the abstract and concrete life cycle environment of the system-of-interest. Context model includes artefacts such as Stakeholder, Human actor, Life cycle phase, Past incident, Constraint, Environment, External system (element) and Glossary item.</p>
<p>Continuous action: A special case of Action. Action that continues over an indefinite time, such as monitoring of a temperature.</p>
<p>Data file: A special case of File. A data file can be of any type and with any data content. The most important attribute is the link to the actual data file.</p>
<p>Device: A special case of Component, such as a sensor, actuator or controller.</p>
<p>Diagram. A special case of File. The Diagram artefact can accommodate diagrams, pictures and photographs of any kind. The most important attribute is the link to the actual diagram file.</p>
<p>Document: The Document artefact defines any type of document or package of documents (such as the actual document and its attachments). The most important attribute is the link to the actual document file.</p>
<p>Drawing: A special case of File. The Drawing artefact can accommodate diagrams, pictures and photographs of any kind. The most important attribute is the link to the actual drawing file.</p>
<p>Electrical port type: The electrical case of the Port type artefact.</p>
<p>Electromagnetic environment: A special case of operational environment describing the electromagnetic characteristics, such as electric fields or possibility of electrostatic discharges, of the operational environment. Also conductive electrical and electromagnetic disturbances, such as noise and voltage pulses and variations, can be described by this artefact.</p>
<p>Engineering artefact: Engineering artefact is the generalisation of all the other artefact types. An Engineering artefact is an abstract artefact type, i.e. there is no special data item in the data repository called engineering artefact.</p>

<p>Engineering process: Systems engineering process description, e.g. according to ISO/IEC/IEEE 15288. Examples of engineering processes are: Stakeholder requirements definition process and Architectural design process.</p>
<p>Environment: Description of the mechanical, climatic, chemical, ergonomic external system, etc. environment, especially in regard to their effect in the system of interest. Domain knowledge can be described here, too. (Domain knowledge model covers models about such real world entities and logic around the system that are relevant to the development of the system-of-interest.)</p>
<p>Event: Event describes any type of event relevant to the system behavioural modelling, such as reaching a temperature limit and pressing a control button. It can also be used to store information about past and potential accidents and incidents. Potential accidents and incidents can be derived from Hazards (potential accident or incident is a presentation of an event-based Hazard).</p>
<p>External port: A special case of Port; a port of an External system element that interacts with the system-of-interest.</p>
<p>External system: A special case of System; a system that interacts with the system-of-interest.</p>
<p>External system element: A special case of System element; a system element that does not belong to the system-of-interest, but that interacts with the system-of-interest.</p>
<p>Failure mode: An identified fault or failure mode of a particular system element.</p>
<p>Failure mode type: A general fault or failure mode type.</p>
<p>Fauna: A special case of operational environment describing the fauna (and its effects) the system-of-interest is expected to be in contact with.</p>
<p>File: A file is the typical resource provided practically by all computer systems. It is expected that the project data repository provides a file system besides the possible database storage.</p>
<p>Flora: A special case of operational environment describing the flora (and its effects) the system-of-interest is expected to be in contact with.</p>
<p>Flow: Specifies the flow between ports. A flow can be electrical (signal), hydraulic (fluid), mechanical (momentum), optical (light), etc. Flows specify the ports (i.e. they inform, what type of flows the ports are able to transmit or receive).</p>
<p>FMECA hazard: A special case of Hazard. A hazard identified by the Failure Modes, Effects and Criticality Analysis method.</p>
<p>Foreign context model: A special case of Foreign model for modelling the abstract and concrete life cycle environment of the system-of-interest. E.g. a SysML model or a CAD model.</p>
<p>Foreign domain knowledge model: A special case of Foreign model for modelling the domain knowledge of the system context.</p>

Foreign environment model: A special case of Foreign model for modelling the environment of the system.
Foreign human model: A special case of Foreign model for modelling humans interacting with the system or who are part of the system.
Foreign model: Any kind of model, SysML model, virtual model, etc. that is not stored as a structured set of artefacts and their relations, but as a 'black-box' model file.
Foreign project model: A special case of Foreign model for the purpose of modelling the project work, management, resources, schedules and financial matters. E.g. a project management tool file.
Foreign system model: A special case of Foreign model for modelling the requirements, behaviour, architecture and other properties of the system-of-interest. E.g. a SysML model or a CAD model.
FTA hazard: A special case of Hazard. A hazard identified by the Fault Tree Analysis method.
Glossary item: Definitions, terms and abbreviations are presented in the Glossary item artefact.
Hazard: The Hazard artefact records the description of the hazards identified during the analysis sessions, the existing protective measures and the corrective actions recommendations.
HAZOP hazard: A special case of Hazard. A hazard identified by the Hazard and operability study method.
Human actor: Any human actor that interacts with the system, whether an assembly man, operator, maintenance man, cleaner, etc., or a bystander who is situated in the hazard zone of the system.
Hydraulic environment: A special case of operational environment describing the fluids the system-of-interest is expected to be in contact with. If the system uses hydraulic power for its operation, such power source are not described in this artefact, but in System element or External system element artefacts.
Hydraulic port type: The hydraulic case of the Port type artefact.
Impact analysis report: A project artefact that is created after modifications and consequent impact analysis.
Individual: Records information about the supplied system individual.
Individual property: Besides the information provided by the Individual artefact and its attributes, a set of individual parameters (colour, existence of options etc.) can be assigned to a product individual. Such information can include static (e.g. serial number) and dynamic information (e.g. operation hours), and configurable information (e.g. hydraulic control ramp parameters).
Information item: A special case of Component; a piece of information, such as a message on a fieldbus.

Information package: A special case of View for the purpose of communicating between the stakeholders, such as licensee and licensor.
Instantaneous action: A special case of Action. A 'zero' length action, such as event or pushing on/off button.
Issue: Any kind of issue prompting actions by the development organisation, e.g. to change specifications or to do redesign. Issue is managed by the modification procedure of the organisation.
Joint: The logical connection between ports. It does not specify the actual physical implementation of the connection. E.g. in case of an electrical connection, it represents the joint galvanic point that connects two or more electrical pins, but it does not specify the wires and cables used to implement the galvanic connection.
Joint element: A special case of a system element. Joint element is finally realised as a mechanical link, optical guide, cable etc.
Life cycle phase: The life cycle model is recorded in the Life cycle phase artefact, e.g. Concept, Development, Production, Utilisation, Support and Retirement according to ISO/IEC TR 24748-1:2010.
Magnetic environment: A special case of operational environment describing the magnetic fields the system-of-interest is expected to experience during operation.
Mechanical environment: A special case of operational environment describing the mechanical characteristics such as shock and vibration of the operational environment.
Mechanical port type: The mechanical case of the Port type artefact.
Message: Specifies a communications message.
Message hazard: A special case of Hazard. A hazard identified by message safety analysis.
Model: A structured set of modelling elements (i.e. a set of engineering artefacts) and their relationships representing an aspect of the system for the purposes of the system development, operation and maintenance.
Model element: Most of the engineering artefacts, such as Requirement, System function, System element are regarded as model elements. A model element is, contrary to Black box artefact, an engineering artefact with known structure and relations with other engineering artefacts according to the artefact model diagrams.
Model file: A special case of File. A model file can be of any type including the file types of commercial and open source modelling tools. The most important attribute is the link to the actual model file or files.
Modification authorisation: Modification authorisation initiates the implementation of the approved modification requests. It may update the original modification request, and hence it has to include description of the planned modification.
Modification log item: Modification log items keep record of the implemented modifications.

Modification request: Needs for modifications of the system-of-interest or of any of its elements is issued formally in Modification request artefacts that need to go through an approval work flow.
Network: A communication network segment with dedicated physical layer and data link layer parameters.
Network hazard: A special case of Hazard. A hazard identified by network safety analysis.
Networked signal flow: A signal flow that goes through at least one network segment.
Networked system parameter: A system parameter that can be accessed through a communication network.
Node: Node is a port to a communications network.
Non-composite flow: A flow that does not constitute of several primitive flows. A normal flow.
Normal channel: A normal functional channel (contrary to test channel) from input (such as sensors) to output (such as hydraulic actuators) through logic (such as programmable logic controller).
OHA hazard: Hazard identified by the Operating Hazard Analysis method.
Operating mode: Description of different types of control or operating modes that can include e.g. the following modes: automatic, manual; remote, local; diagnostics; 'limp home'.
Operating position: Descriptions of the positions at which the system is controlled and operated.
Operational environment: The operational environment artefact can be used to describe the operating environment as a single description or categorised to several environment categories, such as: Spatial environment, Seismological environment, Climatic environment, Magnetic environment, Electromagnetic environment, Mechanical environment, Hydraulic environment, Pneumatic environment, Chemical environment, Optical environment, Particle environment, Terrain, Flora and Fauna. Operational model can be attached with a domain model by a Foreign model artefact. Each of these can be attached with Property artefacts to provide detailed properties of the operational environment.
Operational state: Placeholder for the descriptions of the states of the system, such as power down state, power up state, operating state and emergency stop state.
Optical environment: A special case of operational environment describing the visibility in the operational environment.
Optical port type: The optical case of the Port type artefact.
Particle environment: A special case of operational environment describing the dust, gravel, etc. conditions of the system-of-interest.
Past incident: A record of accidents and incidents history, including near misses, of this type of systems or similar system types. Past incident is a special case of Event.

Person: A human being allocating a role (see the Role artefact type).
PHA hazard: A special case of Hazard. A hazard identified by the Preliminary Hazard Analysis method.
Photograph: Photograph is a special case of File. A photograph can be of any type. The most important attribute is the link to the actual photograph file.
Physical architecture: Physical architecture provides a means to present the system's architecture. The physical architecture artefact can represent a partial view of the whole system structure or different viewpoints of the whole system structure, such as development time view and manufacturing time view.
Pneumatic environment: A special case of operational environment describing wind, air blow and air pressure conditions the system-of-interest is expected to operate in. If the system uses pneumatic power for its operation, such power source are not described in this artefact, but in System element or External system element artefacts.
Pneumatic port type: The pneumatic case of the Port type artefact.
Port: The interfaces are modelled by the Port artefacts. Port is a logical interface entity that is specified in detail by the flows it can carry.
Port type: The port type specifies (with its properties) the interfaces of a product type. A port type may consist of sub-ports. A typical case is a connector with several pins.
Potential incident: A record of potential accidents and incidents derived from event-based Hazards. Potential incident is a special case of Event.
Presentation file: Presentation file is a special case of File. A presentation file can be of any type including the file types of commercial and open source presentation tools. The most important attribute is the link to the actual presentation file.
Product individual: Stores information about the product type instances, i.e. product individuals. Product individual artefact is a specialisation of the Individual artefact with no additional attributes.
Product property: Provides a way to present specification parameters, normally found in datasheets and technical manuals, in a structured way. Such properties include physical properties, such as weight, dimensions and power consumption, but also more abstract parameters, such as safety integrity level or functional capabilities, can be presented as product property.
Product type: Contains the overall identification and description of the library component or sub-system, or of the published system-of-interest.
Product type library: Collects all the data of the product types use to integrate the system-of-interest or of all the systems of the organisation, but it can also collect product type data of product types that are potential for application, although not currently used in any of the organisation's system.
Project: Description of the project that is founded to engineer the system-of-interest. Can be attached with a Foreign project model.

<p>Project activity: An engineering process is composed of project activities, which then compose of project tasks. Example project activities are: Capture stakeholder requirements, Define the architecture.</p>
<p>Project artefact: Special case of Engineering artefacts dealing with the project management issues.</p>
<p>Project data repository: A database or an XML storage for all systems engineering artefacts.</p>
<p>Project model: A special case of Model for the purpose of modelling the project work, management, resources, schedules and financial matters. Project model collects thus engineering artefacts, especially project artefacts, such as Project, Engineering process, Project activity, Project task, Issue, Modification request, Impact analysis report, Modification authorisation, Modification log item, Resource and Role.</p>
<p>Project task: A project activity consists of project task. Examples of project task are: Identify stakeholder and Define the functional architecture of the system.</p>
<p>Property: The Property artefacts present any kind of properties, included required properties, specified properties and realised properties. Behaviour and Structure are in principal properties of the system, but in this model they are not modelled as special cases of property. However, behaviour and structure properties can be described by Product property artefacts; Product property is a special case of Property.</p>
<p>Relationship: An association artefact, i.e. it defines a relation type between two engineering artefacts. Most of these relationships are depicted in the artefacts model diagrams, but in principle, any types of relations can be added. The implementation of a relationship can be the same as that of an engineering artefact, e.g. a database table (i.e. a many-to-many relationship table) with appropriate attributes, such as suspect flag (for impact analysis), relationship type and rationale for the relation.</p>
<p>Requirement: The Requirement artefact defines a requirement and its attributes, such as type and source. A requirement can be a stakeholder requirement, system requirement or constraint.</p>
<p>Resource: A person, tool, room or any other resource needed to carry out the project.</p>
<p>Reusable fragment: A special case of a file. A reusable fragment is not a complete model or document as such, but is embedded into one or more documents or foreign models to compose a document or model.</p>
<p>Rich document file: A special case of File. A document file the support formatting of text and inclusion of tables and drawings and other means of information presentation.</p>
<p>Risk assessment: Works as an assignment to carry out a risk assessment task. It, however, also contains a short description of the results of the risk assessment (the actual results are reported in comprehensive analysis reports).</p>
<p>Risk estimation: Stores the risk estimation parameters and the resulting risk level.</p>
<p>Risk estimation IEC 61508: Risk estimation parameters and risk level according to IEC 61508-5:2010.</p>

<p>Risk evaluation: Stores the judgment and conclusions, on the basis of the risk analysis, of whether the risk reduction objectives have been achieved.</p>
<p>Role: A special case of Component; any person role relevant to system-of-interest. The persons that carry out engineering processes, project activities and project task are tagged by the Role artefact. It is better to use roles instead of person names when assigning project work to persons. This is due to the fact that persons may change. Hence a separate list of person is needed.</p>
<p>Room: A special case of Space, such as an instrument room.</p>
<p>Safety function. A special case of System function specified to provide a functional safety measure, such as safety related stop, prevention of unexpected start-up and hold-to-run function.</p>
<p>Safety requirement: A special case of Requirement. Safety requirements can be distinguished from normal requirements by a True/False flag or by a Requirement type attribute.</p>
<p>Scenario: Scenario is an example flow of actions of a system use case. i.e. it may involve only some of the human actors of the system use case, and it goes through only single path of the possible sequence of actions of the system use case. Scenarios can be used e.g. in software testing and in safety analyses. A Scenario artefact includes the example flow as a story or in a more structured way by well-defined attributes.</p>
<p>Seismological environment: A special case of operational environment describing the expected or recorded seismic characteristics of the operational environment.</p>
<p>Session: The session meeting minutes are recorded in the Session artefact.</p>
<p>Software component: A special case of Component; a reusable, encapsulated, piece of software, a software component.</p>
<p>Space: Space describes an area (2D) or volume (e.g. a room) (3D) where the system-of-interest or its system elements are located. The dimensions or other properties can be presented by attached Property artefacts or directly in the Space artefact.</p>
<p>Spatial environment: A special case of operational environment describing the space where the system-of-interest will operate. Spatial environment can be further elaborated by Structural element and Space artefacts.</p>
<p>Spreadsheet file: A special case of File. A spreadsheet can be of any type including the file types of commercial and open source spreadsheet tools. The most important attribute is the link to the actual spreadsheet file.</p>
<p>Stakeholder: Stakeholder artefacts define the stakeholders that may state requirements for the system (i.e. that have interest in the system). Stakeholders can include e.g. system users, domain experts, principals, investors, board of directors, corporate management, authorities, laws, standards, customers, maintenance staff, training staff, system engineer, buyers of the system and marketing and sales.</p>
<p>Stakeholder requirement: A special case of Requirement: requirement set by a stakeholder.</p>

<p>Structural element: A special case of Component; a hardware element, such as beam, wall, door etc. The relevant structural elements are introduced by this artefact type.</p>
<p>Subsystem: A special case of System and System element; a subsystem is a system from its own point of view and a system element from the parent system point of view.</p>
<p>Support: A special case of Structural element, such as a pipe support.</p>
<p>System: The central node in the artefacts data model. It only includes a small number of attributes, mainly title and a short description of the system, i.e. the system identification. System-of-interest, Subsystem and External system are special cases of System.</p>
<p>System action: A special case of Action. System action is an abstract artefact type; the sequence of system actions of a system task is stored in the Action artefacts.</p>
<p>System activity: System activities are used to describe the intended behaviour of goal-oriented agents, such as humans or technical systems. System activity is an abstract artefact type and has two specialisations, System task and System use case.</p>
<p>System element: The logical representation of the physical component or sub-system that finally is selected or designed and manufactured to implement the system element. System elements together constitute the system-of-interest. System elements include, besides the physical components and subsystems, more abstract elements, such as information items and SW elements.</p>
<p>System element individual: Provides information about a system element individual that may have different contents for different system-of-interest instances the system element belongs to. The System element individual artefact is a specialisation of the Individual artefact with no additional attributes.</p>
<p>System function: A system level function (contrary to SW function), e.g. boom movement.</p>
<p>System model: A special case of Model for modelling the requirements, behaviour, architecture and other properties of the system-of-interest.</p>
<p>System-of-interest. The System-of-interest artefact defines the system under development and during all the life cycle phases. It is a special case of System. It is an abstract artefact type, i.e. there is no special placeholder for system-of-interest, but the system-of-interest is implemented as a System artefact.</p>
<p>System process: A special case of a system use case or system task. Describes the sequence of system use cases or system tasks in an explicit way (contrary to the implicit way in which the sequence of system use cases or system tasks can be concluded from the post conditions of one use case or system task and preconditions of another use case or system task).</p>
<p>System property: A required or specified property of a system (the realised properties are stored in Product property artefacts and the properties of the instances of the Product are stored in Individual property artefact). System properties include system parameters to configure e.g. a system function, system element or port.</p>

<p>System requirements specification: System requirements specification (SyRS) is an instance of the Document artefact type. Besides the actual lists of requirements, SyRS normally provides a description of the system-of-interest in its first chapters.</p>
<p>System task: The System task artefact is the system realisation view of the behaviour. It defines the flow of system functions. It is especially useful in cases where human actors are not involved, but it is also used to identify system functions that cannot be identified from system use cases or use case acts.</p>
<p>System use case: A system level use case (contrary to SW use cases). Use cases can be specified according to SysML.</p>
<p>Temporal action: A special case of Action. An action carried out in finite time; i.e. is not instantaneous ('zero length') neither continuous ('infinite').</p>
<p>Terrain: A special case of operational environment describing the shape of ground of the operational environment.</p>
<p>V&V case specification: V&V case specification specifies a test, analysis, inspection, calculation, design document review, demonstration, calculation, simulation, comparison, sampling, measurement etc. to verify or validate that the artefacts under V&V comply with the expectations. V&V case specification includes, among others, description of the V&V case and its detailed steps, a list of proposed or obliged tools, a description of the expected environment and infrastructure for the V&V case, e.g. required space and temperature, vibration etc. parameters. It also includes a description of the expected results. Issues such as schedule and persons or roles whom this V&V case is assigned to are not included; this is because the V&V case may be re-used in different phases of the project. Such issues are defined in the V&V plans.</p>
<p>V&V claim: The result of verification or validation is recorded here. The result of verification or validation. The main contents are the pass/no-pass verdict and the argumentation of the verdict.</p>
<p>V&V execution interpreter: An instruction document or a program that provides means for interpreting the verification results, such as a program that plots a graph of the virtual measurement data.</p>
<p>V&V execution parameter: The parameters relating to the execution of the verification (or validation), such as the number of simulation runs.</p>
<p>V&V execution report: Records the results of the test, analysis, demonstration, review etc. case executions.</p>
<p>V&V model parameter: The parameters that configure the model item under verification or validation for the verification or validation execution.</p>
<p>V&V plan: Collects a set of test, analysis, demonstration, review etc. cases to form a specific sequence of tests for a specific purpose, such as for Factory Acceptance Test (FAT).</p>
<p>V&V report: V&V report is an instance of the Document artefact type. It reports the results of verification and validation activities done according to the V&V plans.</p>

V&V requirement: A special case of Requirement. In many cases, the requirements specification or safety standards set requirements as to how the design shall be verified or validated.

Video file: Video file is a special case of File. A video file can be of any type. The most important attribute is the link to the actual video file.

View: View provides the possibility to pick a set of engineering artefacts and their relationships for specific project process concerns.

Wiki page: A special case of a document for the purpose of editing and browsing the document directly via the web browser interface. Not necessarily edited using the Wiki syntax.

Title	Conceptual model for safety requirements specification and management in nuclear power plants
Author(s)	Teemu Tommila & Jarmo Alanen
Abstract	<p>Clearly stated requirements, systematic configuration management and traceability are a key prerequisite for the safety of industrial plants. In spite of the long research tradition and training, in particular in software engineering, poor requirements are still a major source of safety problems. Requirements engineering is a challenge also in nuclear power plant automation. The characteristics of requirements engineering, such as multi-disciplinary collaboration, uncertainties and abstract concepts, are difficult for engineers who prefer to think in terms of technical solutions. The working practices and tools for describing requirements are often vague. Even the standards and guidelines developed for the nuclear domain fail to provide a clear and consistent vocabulary for describing power plants. Well-defined terminology would, however, be needed for communicating the requirements between various stakeholders and engineering disciplines.</p> <p>The goal of this report is to foster mutual understanding among industrial professionals by providing clear terminology. Furthermore, the concepts form a basis for design guidelines and computer tools. Accordingly, the report is divided into two parts, the first one discussing general modelling principles and the second one suggesting a more practical data model for tool development.</p> <p>One of the starting points is that requirements cannot be discussed in isolation from other engineering activities and system descriptions. Even the boundary between requirements and design solutions is not always clear. Therefore, this report is not limited to requirements but discusses the principles of modelling complex socio-technical systems in a broader sense. The second starting point is that the number of requirements and dependencies requires computer tools. Computer tools, in turn, need consistent data models. This is why this report has taken influences also from international standardisation of product data modelling. It works towards this vision in a semi-formal, database-oriented way by defining concepts that might be used in future computer tools to describe power plant systems and their requirements.</p>
ISBN, ISSN, URN	ISBN 978-951-38-8365-2 (URL: http://www.vttresearch.com/impact/publications) ISSN-L 2242-1211 ISSN 2242-122X (Online) http://urn.fi/URN:ISBN:978-951-38-8365-2
Date	November 2015
Language	English, Finnish abstract
Pages	120 p. + app. 26 p.
Name of the project	Safety requirements specification and management in nuclear power plants (SAREMAN)
Commissioned by	Ministry of Employment and the Economy (SAFIR2014 programme)
Keywords	Nuclear power, Systems Engineering, Requirements Engineering, Instrumentation and Control Systems
Publisher	VTT Technical Research Centre of Finland Ltd P.O. Box 1000, FI-02044 VTT, Finland, Tel. 020 722 111

Nimeke	Tietomalli ydinvoimalaitosten turvallisuusvaatimusten määrittelyyn ja hallintaan
Tekijä(t)	Teemu Tommila & Jarmo Alanen
Tiivistelmä	<p>Selkeät vaatimukset, suunnittelutiedon konfiguraation hallinta sekä jäljitettävyyden ovat teollisuuslaitosten turvallisuuden edellytyksiä. Laajasta tutkimuksesta ja koulutuksesta huolimatta huonosti määritellyt vaatimukset ovat edelleen yksi syy teollisuusautomaation turvallisuusongelmiin. Vaatimusten käsittely on haaste myös ydinvoima-automaatiossa. Sen monitekninen luonne, epävarmuudet ja abstraktisuus ovat hankalia insinööreille, jotka ovat kiinnostuneempia alansa teknisistä ratkaisuista. Vaatimusten esitystavat ja työvälineet ovat usein kehittymättömiä. Myöskään standardit ja ohjeistot eivät tarjoa johdonmukaista käsitteistöä ydinvoimalaitosten kuvaamiseen. Hyvä sanasto olisi kuitenkin välttämätön investointiprojektin eri osapuolten ja tekniikan alojen välisessä yhteistyössä.</p> <p>Tämän raportin tavoite on tukea eri alojen yhteisiä ajatusmalleja selkeän terminologian avulla. Lisäksi on tarkoitus, että määritellyt käsitteet tarjoavat pohjan suunnitteluohjeistojen ja tietokonetyökalujen kehittämiseksi. Tämän jaottelun mukaisesti raportti on jaettu kahteen osaan, joista ensimmäinen tarkastelee järjestelmien kuvaamisen yleisiä periaatteita ja jälkimmäinen ehdottaa konkreettisempia tietomalleja työkalukehityksen pohjaksi.</p> <p>Raportin yksi lähtökohta on, että vaatimusten käsittelyä ei voida erottaa muista suunnitteluaktiviteeteista ja järjestelmäkuvauksista. Jopa vaatimuksen ja suunnitteluratkaisun ero on usein häilyvä. Sen vuoksi tämä raportti ei rajaudu vain vaatimuksiin, vaan tarkastelee monimutkaisia sosioteknisiä järjestelmiä laajemmin. Toinen lähtökohta on, että vaatimusten suuri määrä ja erilaiset riippuvuudet edellyttävät tietokonetyökalujen käyttöä. Työkalut taas vaativat pohjaksi yksiselitteisiä tietomalleja. Siksi tässä raportissa on hyödynnetty myös tuotetiedon hallinnan kansainvälisiä standardeja. Tarkoitus on edetä kohti puoliformaaleja tietorakenteita ja tietokantapohjaista suunnittelutapaa ydinvoimalaitosten järjestelmien ja niiden vaatimusten kuvaamisessa.</p>
ISBN, ISSN, URN	ISBN 978-951-38-8365-2 (URL: http://www.vtt.fi/julkaisu) ISSN-L 2242-1211 ISSN 2242-122X (Verkkojulkaisu) http://urn.fi/URN:ISBN:978-951-38-8365-2
Julkaisuaika	Marraskuu 2015
Kieli	Englanti, suomenkielinen tiivistelmä
Sivumäärä	120 s. + liitt. 26 s.
Projektin nimi	Turvallisuusvaatimusten määrittely ja hallinta ydinvoimalaitoksilla (SAREMAN)
Rahoittajat	Työ- ja elinkeinoministeriö (SAFIR2014-ohjelma)
Avainsanat	
Julkaisija	Teknologian tutkimuskeskus VTT Oy PL 1000, 02044 VTT, puh. 020 722 111



Conceptual model for safety requirements specification and management in nuclear power plants

ISBN 978-951-38-8365-2 (URL: <http://www.vttresearch.com/impact/publications>)
ISSN-L 2242-1211
ISSN 2242-122X (Online)
<http://urn.fi/URN:ISBN:978-951-38-8365-2>

