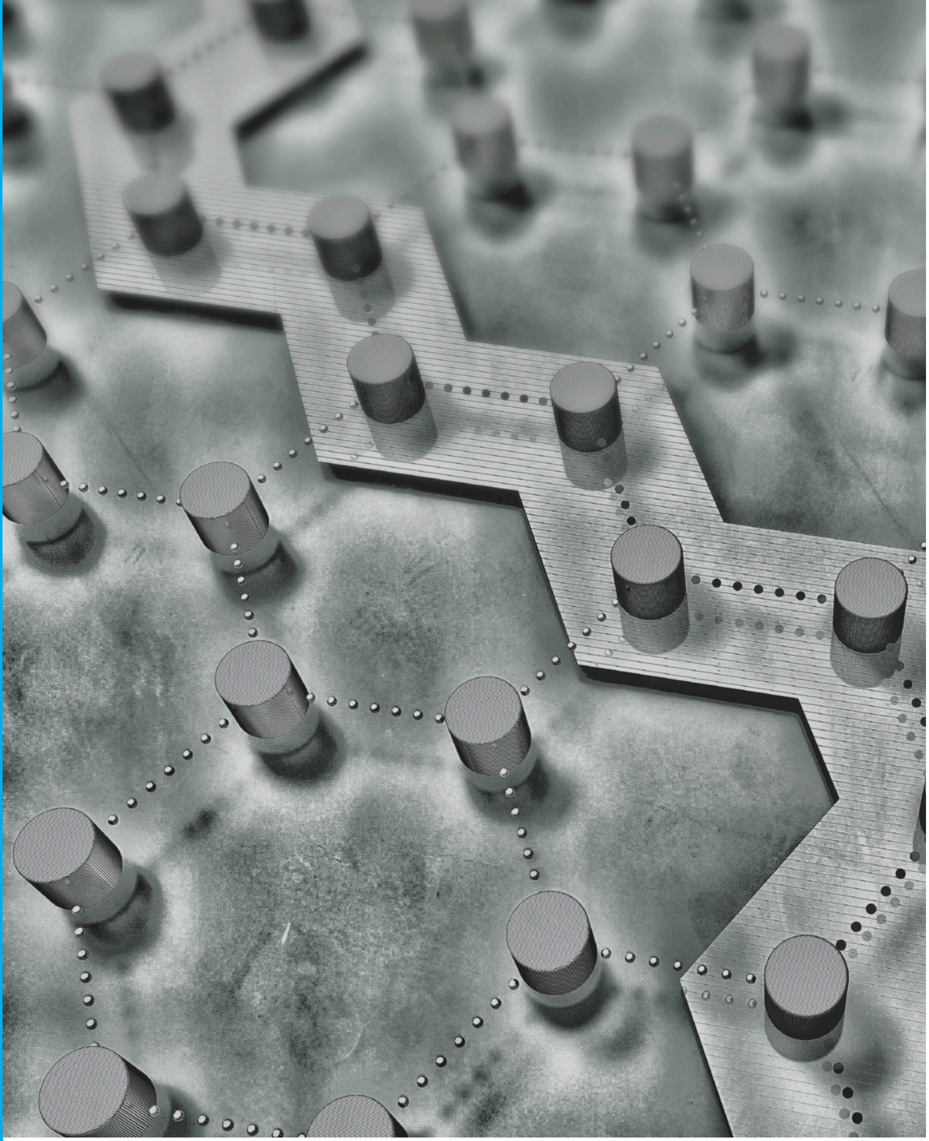


VTT Technology • Science • Visions • Research Highlights
236

Requirements traceability in simulation driven development

Jarmo Alanen | Pekka Isto | Petri Tikka | Teemu Tommila

VTT TECHNOLOGY 236

Requirements traceability in simulation driven development

Jarmo Alanen, Pekka Isto & Teemu Tommila

VTT

Petri Tikka

Tampere University of Technology student

ISBN 978-951-38-8359-1 (URL: http://www.vttresearch.com/impact/publications)

VTT Technology 236

ISSN-L 2242-1211
ISSN 2242-122X (Online)
http://urn.fi/URN:ISBN:978-951-38-8359-1

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

Cover image: iStockphoto

Preface

This report is the main result of VTT subproject Task 3 (‘Requirement and simulation driven design’) of the SIMPRO project (‘Computational methods in mechanical engineering product development’) carried out during the years 2012–2015 and financed by the following organisations: Tekes – the Finnish Funding Agency for Innovation, Aalto University, Tampere University of Technology, Lappeenranta University of Technology, University of Jyväskylä, Wärtsilä Finland Oy, Patria Land Systems Oy, KONE Oyj, MeVEA Oy, FS Dynamics Finland Oy Ab, EDR & Medeso Oy, Dassault Systèmes Oy, Techila Technologies Oy and VTT Technical Research Centre of Finland Ltd. The project was coordinated by VTT.

We thank VTT subproject Task 4 leader Ali Muhammad for the discussions to identify the simulation engineering artefacts.

Contents

List of abbreviations

In the following, the abbreviations used in this report are listed.

ALM	Application Life cycle Management
CAD	Computer Aided Design
CAN	Controller Area Network
DXL	DOORS Extension Language
FMECA	Failure Mode, Effects and Criticality Analysis
FTA	Fault Tree Analysis
GUI	Graphical User Interface
HAZOP	Hazard and Operability Study
MEWP	Mobile Elevating Work Platform
OHA	Operating Hazard Analysis
PHA	Preliminary Hazard Analysis
PL	Performance Level
PLC	Programmable Logic Controller
PLM	Product Life cycle Management
SDO	Service Data Object (a CANopen term)
SEAModel	Systems Engineering Artefacts Model
SIL	Safety Integrity Level
SME	Small and Medium-sized Enterprise
SRP/CS	Safety-related part of a control system
SysML	Systems Modeling Language

TIM	Traceability Information Model
UCSA	Use Case Safety Analysis (a VTT defined method based on OHA)
UML	Unified Modeling Language
URL	Uniform Resource Locator
V&V	Verification and Validation
XML	eXtensible Markup Language
XSD	XML Schema Definition

1. Introduction

The key success factors in development of complex systems are among the following:

1. systematic processes and life cycle model (such as ISO/IEC/IEEE 15288 [2008] and its daughter standards)
2. a systematic model for the engineering artefacts and their relations (such as specifications, CAD-models, pieces of information, and so forth)
3. an effective organisation model
   - well-defined roles and responsibilities (like systems engineer, requirements engineer, and so forth)
   - well-defined collaboration model (to facilitate consistent view in all involved organisations of the goal, data and state of the development)
4. well-planned use of project management and systems engineering tools
   - a good selection of engineering tools (model based tools advocated)
   - a flexible tool integration model (to allow integration of various tools used by the collaboration partners)
5. a tool to orchestrate all of the above (such as a PLM tool).

This report addresses the second success factor, systematic model for the engineering artefacts. The particular focus of the research the results of which are reported in this report is on the traceability of the engineering artefacts in simulation oriented systems development.

Engineering artefacts include, among others, requirements specifications, system functions specifications, system architecture descriptions and verification and validation artefacts, and simulation related artefacts being a part of the verification and validation artefacts. In complex systems, arrangements for traceability and impact analysis play an important role in managing the iterative systems development. To provide the traceability of engineering artefacts, the following factors need to be provided by the organisation carrying out the systems engineering of the system:

- Traceability Information Model (TIM)

- a tool to store and trace the engineering artefacts according to the TIM. The tool shall provide an integration model to facilitate integration of different kinds of engineering tools.

Another important factor is the motivation of the engineering personnel to create and store engineering artefacts systematically according to the information model. This is more of a psychological factor, but resistance can be alleviated by engineering tools with good usability. This holds true especially in case of a traceability management tool: Traceability management is a tedious and time consuming task, which leads to neglects and failures in creating and maintaining artefacts traces.

Yet another relevant factor is the cost of the tools: Development of complex system requires involvement of several organisations, including small and medium-sized industrial enterprises that cannot afford to buy and maintain tools that cost tens of thousands per year.

2. Objectives of the work

In this work, our objective is to provide an easily implementable data model for engineering artefacts (i.e. work products). We call the model Systems Engineering Artefacts Model (SEAModel). The artefacts model defines the core engineering artefact types and their relations. The artefacts model is easy to convert to a traceability information model due to the fact that most of the artefacts relations can be considered as trace relations.

Furthermore, our goal is to provide a demonstration of the usage of SEAModel in simulation oriented case example.

We do not study the existing data models, such as ISO 10303-233 (2012), a.k.a. AP233, due to the fact that this work is a continuation of our earlier publication (Alanen et al. 2011), which discusses ISO 10303-233 and other systems engineering data models.

Our focus is in machinery with mechanics and programmable electronic control systems.

3. Systems engineering artefacts model

3.1 Introduction

The main motivation for the systems engineering artefacts model, i.e. SEAModel, is the need to provide traceability between different artefacts, especially from requirements to design and implementation, from implementation to test execution, and from test execution to verification and validation reporting. Another motivation is to provide a model that can easily be implemented onto a relational database, onto a Product Lifecycle Management (PLM) tool or onto an Application Lifecycle Management (ALM) tool or whichever tool that provides management of structured and relational data.

SEAModel consists – at the time of writing – of the following packages (see Figure 1).

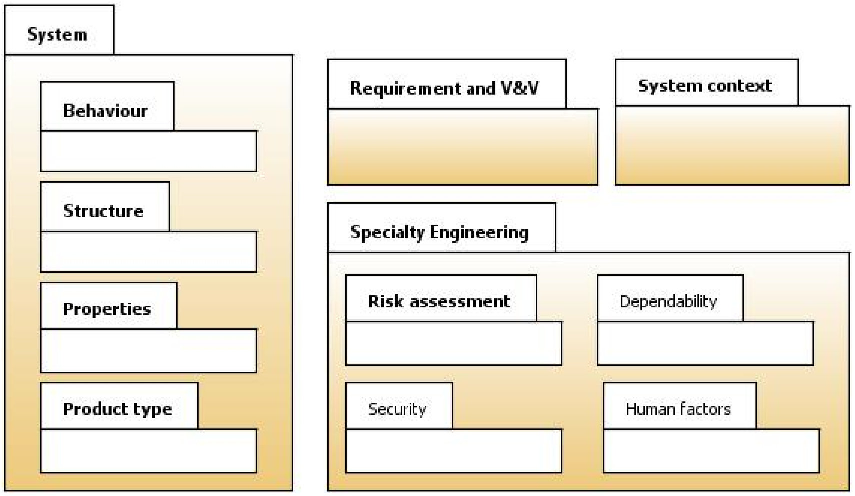


Figure 1. SEAModel packages.

For some of the artefact types, it is not easy or possible to define, which package, 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 1 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 1 import elements from each other.

The System package includes nested packages for behaviour, structure, properties[1] and product type. The specialty engineering package includes nested packages for risk assessment (safety), security, dependability and human factors engineering; currently only the Risk assessment package is defined in more detail. The Speciality Engineering package can, of course, later include models from other fields, like environmental sustainability.

Due to limited resources, in this work, we did not model project artefacts, like project processes, activities and task, or modification requests, impact analysis reports and the like.

The packages are described in more detail starting from Section 3.3. The notation for the models follows the SysML block definition diagram notation. A short guide for the notation is provided below (Figure 2).

[1] Behaviour and structure are also properties of a system, but instead of using a phrase ‘other properties’ or ‘physical and other properties’, we simply use the phrase ‘properties’.

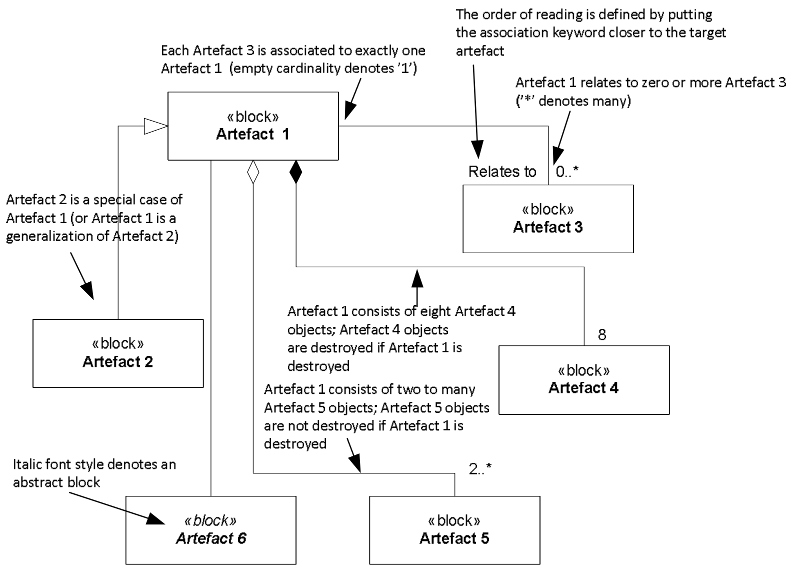

The order of reading is defined by putting the association keyword closer to the target artefact
Each Artefact 3 is associated to exactly one Artefact 1 (empty cardinality denotes '1')
Artefact 1 relates to zero or more Artefact 3 ('*' denotes many)
«block» Artefact 1
Relates to
0..*
«block» Artefact 3
Artefact 2 is a special case of Artefact 1 (or Artefact 1 is a generalization of Artefact 2)
«block» Artefact 2
Artefact 1 consists of eight Artefact 4 objects; Artefact 4 objects are destroyed if Artefact 1 is destroyed
8
«block» Artefact 4
Artefact 1 consists of two to many Artefact 5 objects; Artefact 5 objects are not destroyed if Artefact 1 is destroyed
Italic font style denotes an abstract block
2..*
«block» Artefact 6
«block» Artefact 5

Figure 2. SysML block definition diagram notation guide.

Each block in the block definition diagrams introduced in Sections 3.3 to 3.6 represent an artefact type that can be implemented as a database table or similar structured set of data elements in a data repository tool. The specialised artefact type (like Artefact 2 in Figure 2) can be implemented in two different ways, as a separate database table (or similar) or by using an attribute in a single table to denote the specialisation type. The abstract artefact types (like Artefact 6 in Figure 2), however, need not to be implemented; they are only provided to help understand the models.

Before we present the actual SEAModel packages starting from Section 3.3, the structure of the anticipated data repository to store the actual model elements is presented in the following section (Section 3.2). The notation is the same (see Figure 2) as for the actual SEAModel packages.

3.2 Data repository model

SEAModel classifies the engineering artefacts that populate a project data repository. In other words, we don't speak about a particular system model 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 here the structure of a data repository that manages artefacts and their relationships according to SEAModel.

Figure 3 shows the structure of the engineering data repository. 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 number[2] 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. However, documents can't be avoided. So, model elements and black box artefacts are the two main classes of engineering artefacts. Correspondingly, the 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 data repository model. The main types of black box artefacts are documents (like international standards) and foreign models (like 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[3] 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 system and another one for the system context. Models can be used to encapsulate related model elements and their relationships into logical units. However, a model can share elements with other models. Note here that the model elements can be textual, like 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.

Views are used to define, which relevant parts of the models and black box artefacts are combined and shown to the users for specific purposes like risk assessment and formal approvals. A single document or model can exist in several views. Version management of the shared artefacts needs to be carefully planned to ensure correct versions of the model elements for each of the models or views sharing the model elements; models or views may want to include different versions of the same model element.

[2] 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 and other features is not elaborated in this report. It is assumed that the implementation platform provides basic version control of the engineering artefacts.

[3] Only some of them are depicted in Figure 3, namely Requirement, System element and System function; the sections starting from Section 3.3 present the rest of the model elements.

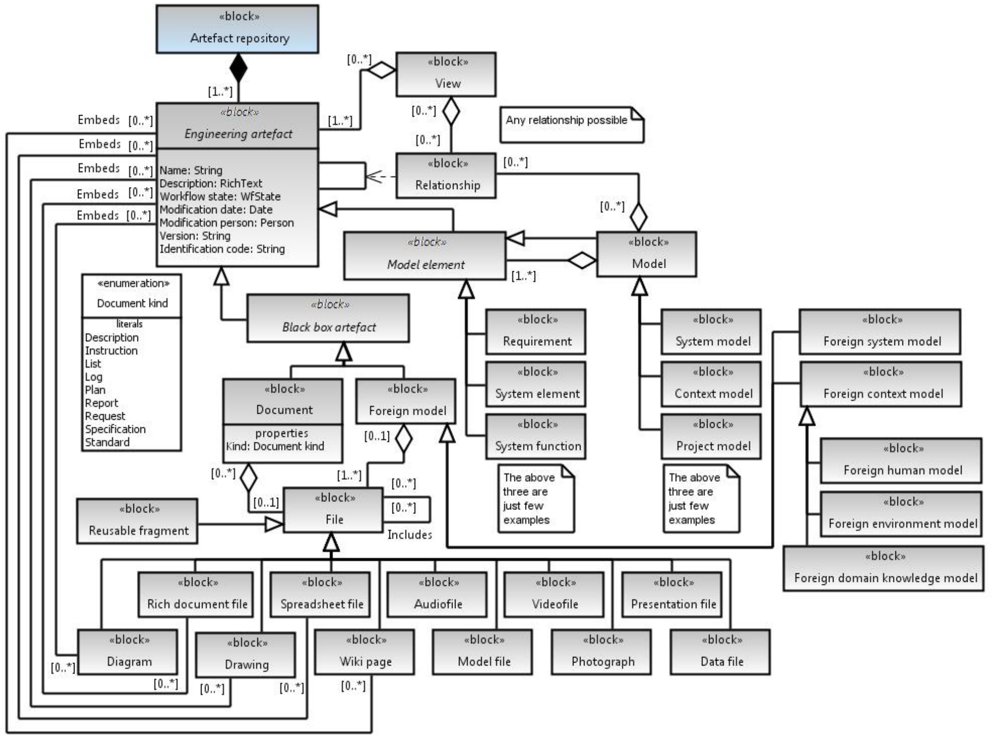


Figure 3. Data repository model.

The black box artefact is elaborated further in Figure 3. 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.

As can be seen in Figure 3, any engineering artefact can be linked to a Document or Foreign Model artefact by applying a relationship between the two engineering artefacts; for example, a requirement can be linked to a requirements specification document. For conciseness, such links are not depicted in subsequent models starting from Section 3.3, except in some rare cases, like the link between a requirement and requirements specification document that is depicted in the Requirements and V&V model in Section 3.5.1.

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 3. These document kinds (except document kind Standard) are specified by IEC 61508-1 (2010), excluding document kind Diagram due to the fact that Diagram represents content type, not document kind[4]. 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 3. The Word processing file, Wiki page, Diagram, Drawing and Spreadsheet artefact types possess a special property to include artefacts (e.g. requirement objects, system use cases and other engineering artefacts defined in the subsequent models) within the free-flow text. This facilitates automatic or semi-automatic document generation[5].

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 3 includes some examples of such foreign model specialisations.

The package structure depicted in Figure 1 does not represent the structure of the data repository. Instead, the package structure is created to help the reader understand the SEAModel contents.

In the subsequent sections starting from Section 3.3, the SEAModel packages are described in more detail. The artefact types (i.e. the ‘blocks’) presented in the subsequent sections (and above in Figure 3) are described in Appendix A.

[4] In other words, specifications, instructions and other document kinds can have content in the form of diagram, text, and so forth.

[5] 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.

3.3 System package

A system is characterised by its behaviour, structure and other properties. The system integrates product types from other vendors or the system developer itself, and finally the ready system is launched as a product type to be used by its acquirer. Hence, as depicted in Figure 1, the System package consists of the System artefacts model and four sub-packages, Behaviour, Structure, Properties and Product type. These five models are described as block definition diagrams in the following sub-sections (Sections 3.3.1 to 3.3.5).

To better understand the sub-sequent models, it is necessary to understand the hierarchical structure of the systems. The system hierarchy model according to ISO/IEC/IEEE 15288 (2008) is illustrated in Figure 4. The system structure is modelled by two artefact types: System (system-of-interest being its special case) and System element, in which a system element can be a subsystem or an atomic element, i.e. a component.

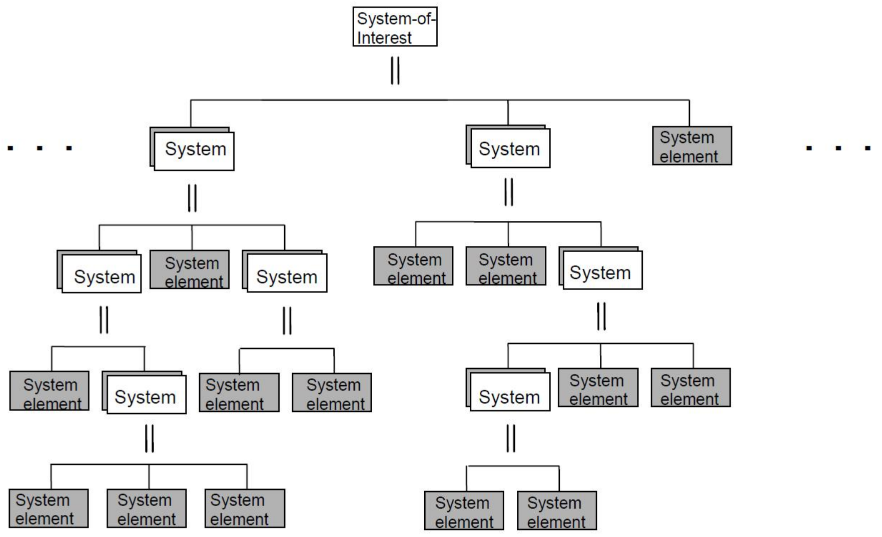


Figure 4. System-of-interest structure model (ISO/IEC/IEEE 15288 2008).

The model in Figure 4 is partially redrawn in Figure 5 to better illustrate the fact that a system only consists of system elements, and that a system element can be a subsystem or an atomic element, and that a subsystem does not have a special modelling element, but is a system (in fact a system-of-interest) from the point of view of the developer of the subsystem.

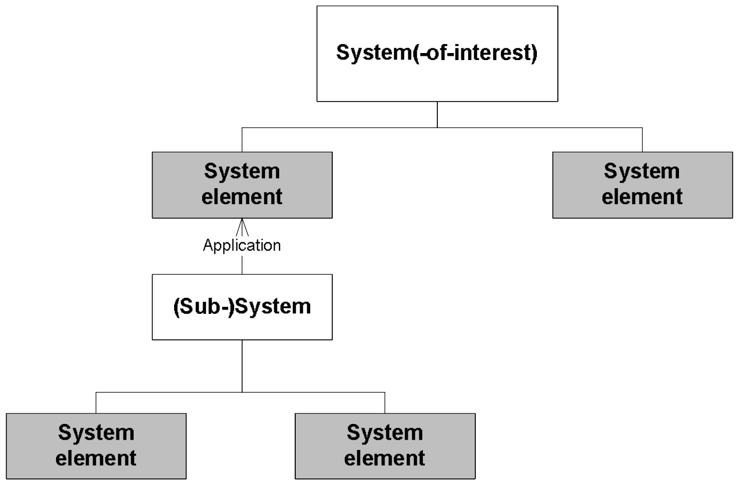


Figure 5. ISO/IEC/IEEE 15288:2008 partially redrawn.

In case of SEAModel, we enhance this model by distinguishing a development time system and a published system. For the purpose of published system, a new modelling element is introduced: Product type. The Product type modelling element stores the information necessary to apply the particular product type by an acquirer of the system to his or her specific purposes. It does not store all the development time information, which can partly be confidential, but the development time information is stored in the System (or its special case: System-of-interest) modelling element and its relating artefacts. In other words, the revised structure model provides three aspects of a system: development time information in the **System** model element, the published information in the **Product type** model element and the role of the published system as a sub-system or component in the **System element** model element[6]. Figure 6 illustrates the revised system-of-interest structure model with the distinguished model element for published systems.

[6] Or one can think the System element artefact as a logical presentation of the physical architecture element, and the Product type artefact as the representation of the physical implementation of the physical architecture element.

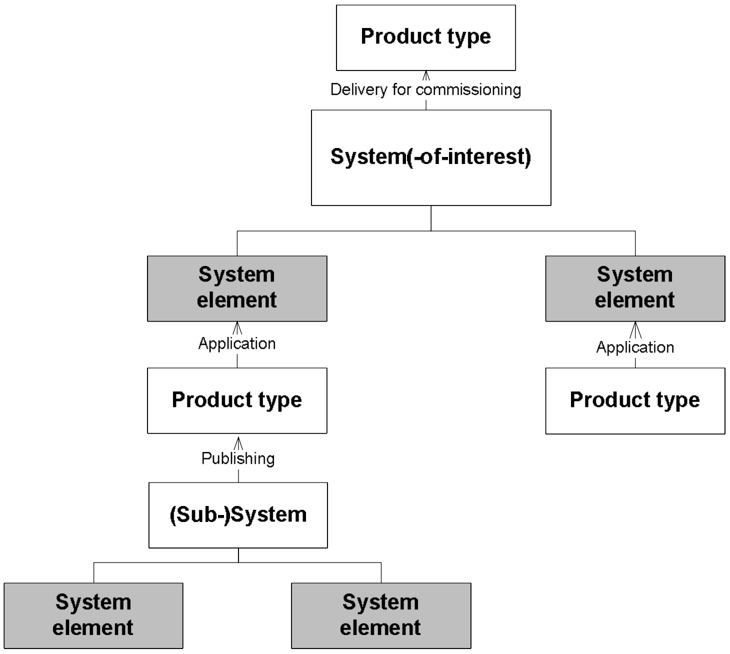


Figure 6. ISO/IEC/IEEE 15288 (2008) structure model customised by adding a new model element: Product type.

There is yet another aspect of a system: a system individual. When a published system is incorporated into the design, it gets its application specific role in the System element model element, but when several instances of a system-of-interest and its constituent sub-elements are produced, each produced system or applied system element has got its instance specific data, such as serial number and operating hours. Hence yet another modelling element is introduced: **Individual**. The supplier of the system and the acquirer of the system (and maybe some other stakeholders) may collect and store different kinds of information about the system and system element individuals. The completed system-of-interest structure model with the Individual model element is depicted in Figure 7.

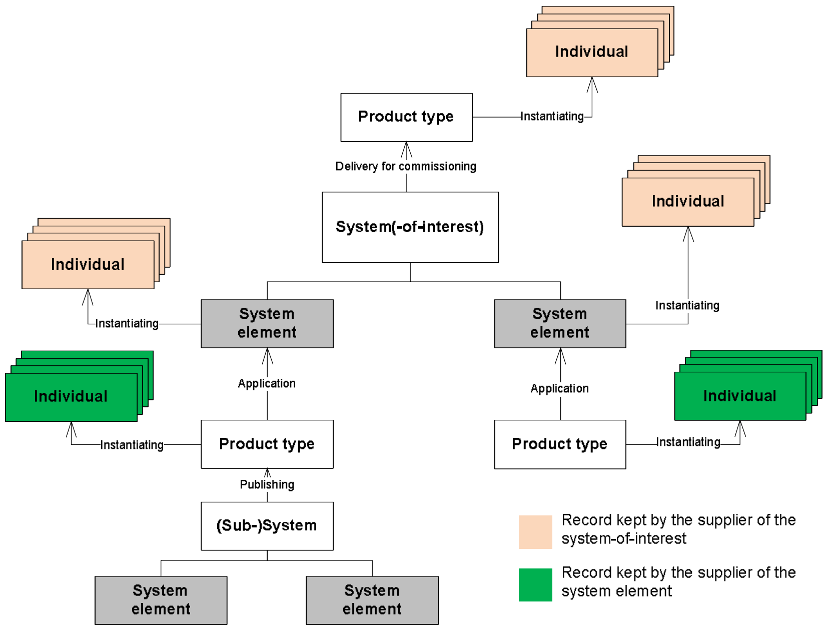


Figure 7. System-of-interest model with model elements for the system element individual and product type individual.

SEAModel is applied to each system, whether a system-of-interest or a subsystem developed in-house. For off-the-shelf sub-systems and components, SEAModel 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 SEAModel is not used by the system element developer, it is difficult to arrange seamless traceability of requirements from the main system to the farthest system element, and traceability of verification artefacts from the system element to the main system.

3.3.1 System artefacts model

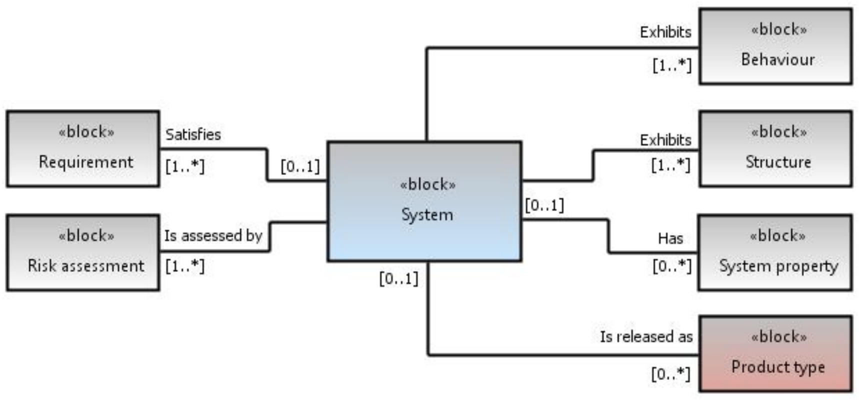


Figure 8. System artefacts model diagram.

The System artefacts model (Figure 8) is the main and top level model to define the artefacts relating to the system under development. The parts of the system model (besides the System artefact type) are Requirement, Structure, Behaviour, Risk assessment, System property and Product type 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.

3.3.2 Structure package

The Structure sub-package provides models for the physical architecture of the system. The Structure package includes two models, the System structure artefacts model (Section 3.3.2.1) and a draft version of the Network artefacts model (Section 3.3.2.2) to define artefacts and their relations of communications networks; currently, only the CANopen network type is covered.

3.3.2.1 System structure artefacts model

The System structure artefacts model is based on the AP233 system structure concept model presented in Figure 9.

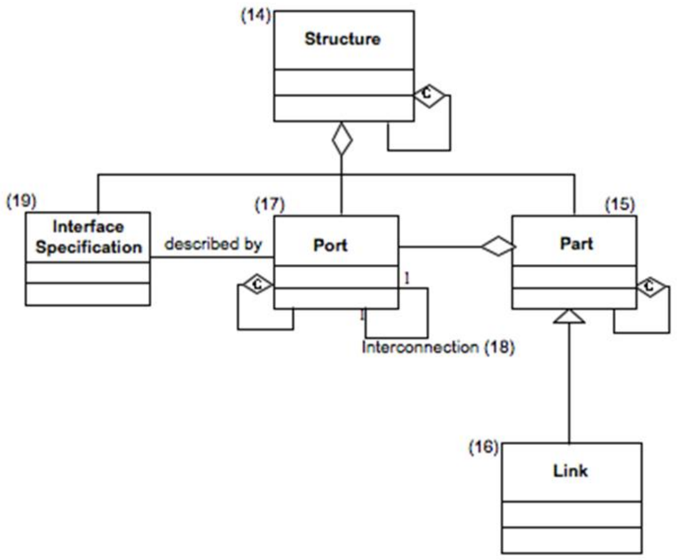


Figure 9. System structure concept model of AP233 (ISO/DIS 10303-233 2009).

The main idea in SEAModel is that the physical structure of a system is defined by its system elements (Parts in the AP233 model above) and their interfaces. The Structure artefacts model is depicted by two diagrams, system decomposition artefacts diagram (Figure 10) and system interfaces artefacts diagram (Figure 11).

The system decomposition diagram in Figure 10 formalises the System-of-interest model outlined in Section 3.2 in Figure 7 with the addition of the concept of External system element.

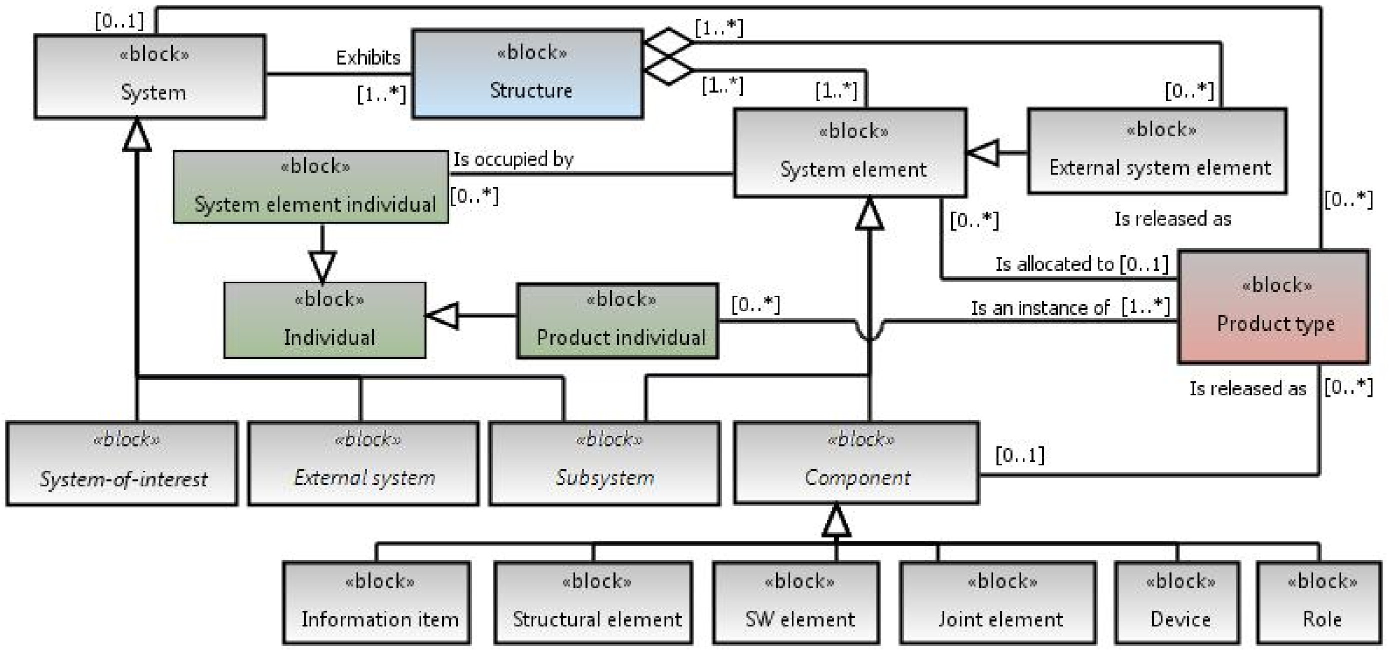


Figure 10. System decomposition artefacts diagram.

The hierarchical decomposition diagram depicts the physical structure of the system: Structure consists of system elements that can be atomic, i.e. components, or non-atomic, i.e. subsystems. Subsystem and 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. a programmable control system or mechanical system. In other words, the model does not exclude hydraulic systems and the like, although the focus in this report is in programmable electronic control systems and mechanical system. This applies to all of the models presented in this chapter. However, some special artefact types might be needed to be added to the models presented in this report if comprehensive coverage of hydraulic and other specific 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 11).

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

The artefact types (i.e. the blocks in Figure 10), the title of which is written in Italics typeface, are abstract, i.e. it is assumed that in the implementation platform of SEAModel, such artefact types do not have any dedicated representation. 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 representation in the data repository implementation.

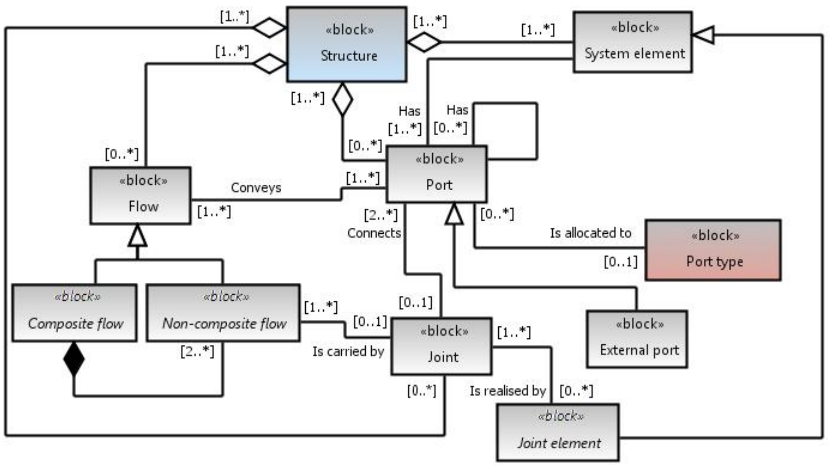


Figure 11. System interfaces artefacts diagram.

The System interfaces diagram is supplied in Figure 11. 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 Structure to Port seems unnecessary, because a Structure 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 Structure artefact if necessary. This association is in fact useful in the case of the behaviour model (see Section 3.3.3); a system function can be allocated to a structure 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 structure 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 structure 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 the 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 10 nor in Figure 11; see Section 3.3.4 instead.

A structure can be described and illustrated in one or more documents, models and diagrams, like 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 Section 3.2).

3.3.2.2 Network artefacts model (CANopen case)

The Network artefacts model in Figure 12 defines artefacts and their relations of communications networks.

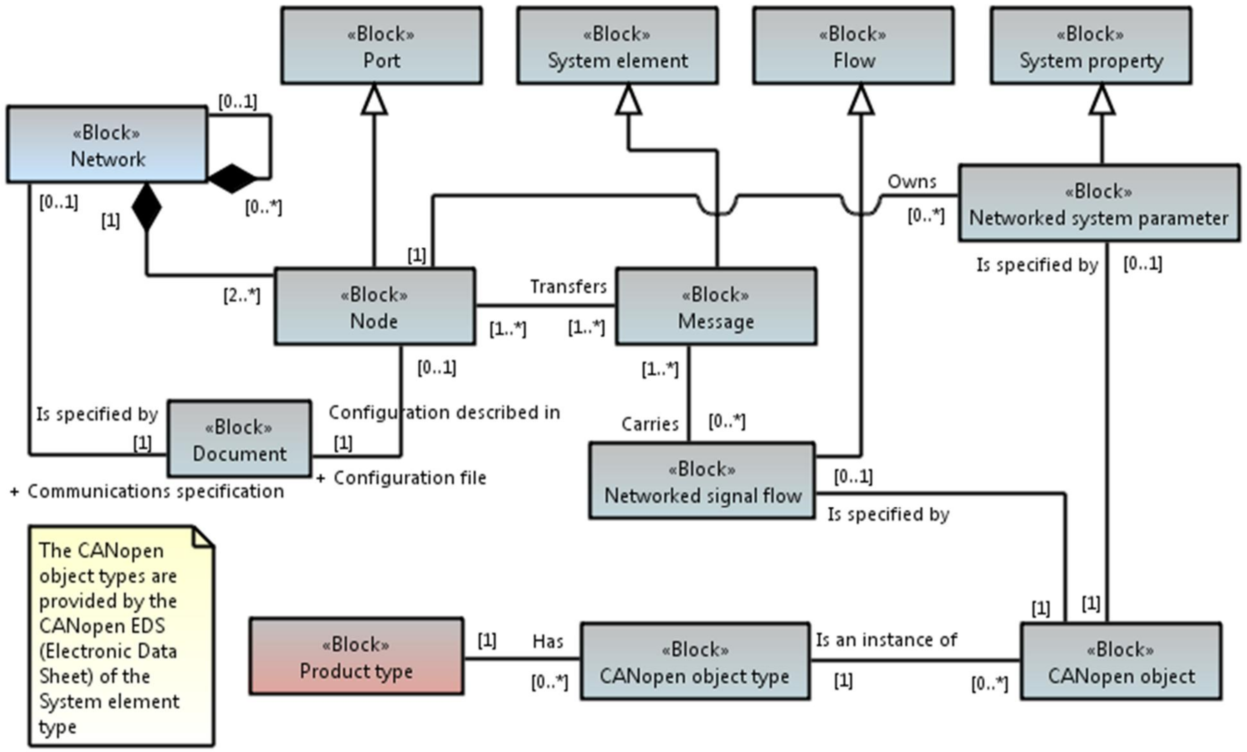


Figure 12. Network artefacts model diagram (draft).

The model in Figure 12 only supports CANopen networks. However, the main artefacts, Network, Node, Message, Networked signal flow and Network system parameter are supposed to be applicable to other communication protocols as well.

Network consists of nodes and sub-networks. A node is a special case of Port (see description of Ports in Section 3.3.2). A network is specified by a protocol specification and by a message specification[7] that is occupied by Message artefacts. Messages are owned by nodes. A message can carry one or more Networked signal flows. A networked signal is a special case of Flow (see description of Flows in Section 3.3.2). A networked signal flow is mapped to a CANopen object (in the case of CANopen networks). A CANopen object is an instance of CANopen object type owned by Product type.

Furthermore, some of the system parameters reside on remote nodes. Hence to make them accessible, Networked system parameters are defined as a special case of System properties. In the case of CANopen, access to such parameters is accomplished through its SDO-service.

Note that this model is not complete and has neither been tested nor demonstrated. The model above was created to ensure that the Structure artefacts model can be linked with the Network artefacts model.

3.3.3 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 13. It encompasses the description of the functionality of the system. Its core artefacts are the System use case, System task and System function artefacts. Interface model for the behaviour artefacts is not provided in this report.

[7] The protocol specification and message specification together constitute the communications specification. The communications specifications should also present the network architecture.

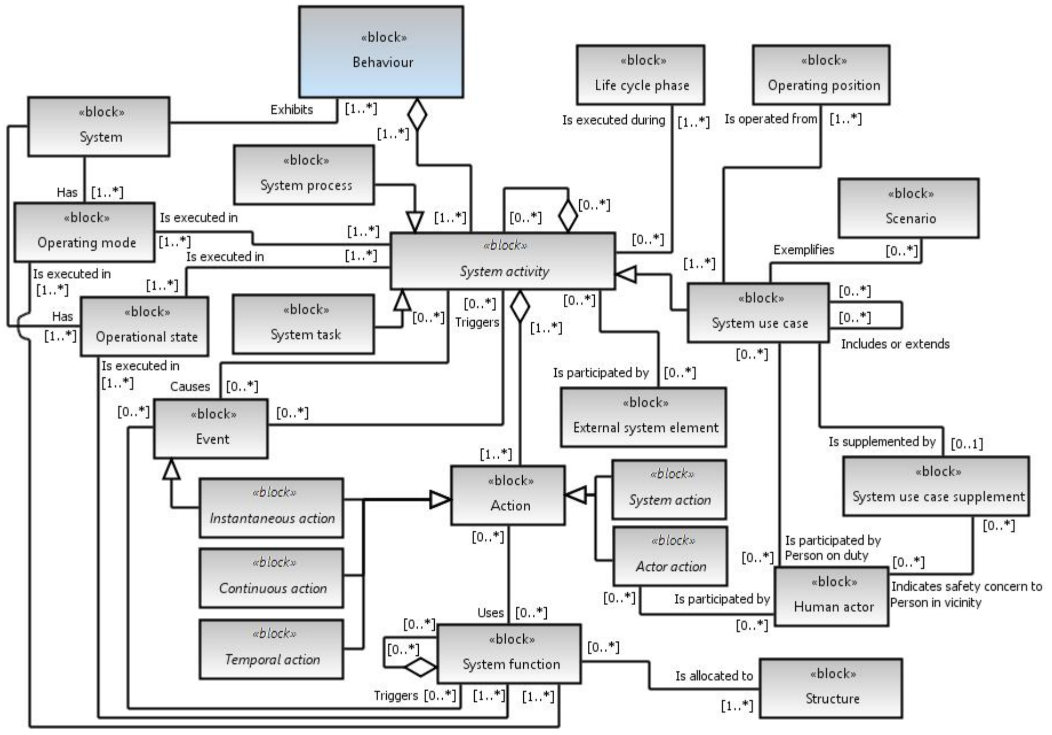


Figure 13. Behaviour artefacts model diagram.

The Behaviour artefacts model in Figure 13 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 according to ISO 12100 [2010]). However, the main artefact type in the model is the System activity artefact. System activity is specialised by System process, System task and System use case[8]. 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 representation 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.

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 in which human actors are not involved (like 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 Figure 18 in which the System use case artefact type is pointed out as an example artefact to be traced).

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 Ac-

[8] Action could also be defined to be a specialisation of Activity, but we do not do that here because activities are non-atomic and action is atomic.

tion artefacts. The reason for separating the Action artefact from the System use case artefact is that during the Use Case Safety Analysis (UCSA) (Alanen & Tiusanen 2010) 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 13, but can be seen (at least implicitly) in Figure 18 and Figure 15 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, like 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 Section 3.2).

3.3.3.1 Safety function artefacts model

The Safety function artefacts model (Figure 14) completes the behaviour model of a system.

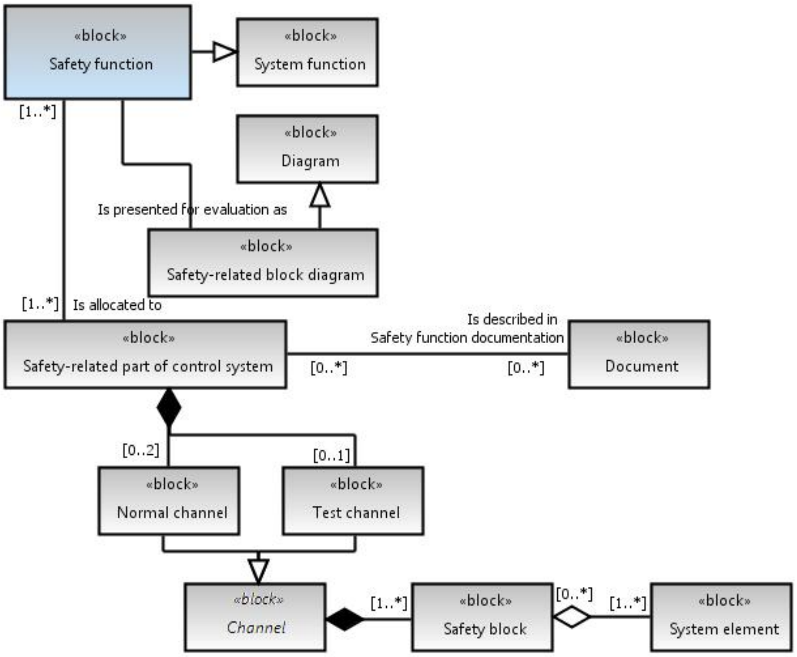


Figure 14. Safety function artefacts model diagram.

The attributes of the Safety function artefact are selected such that the requirements for a safety function specification according to IEC 62061 (2005) are fulfilled[9]. A safety function is a special case of a system function; it is not a subfunction of a system function.

The model includes the artefact types needed to carry out the Performance Level (PL) evaluation according to ISO 13849-1:2006. (Neither the IEC 62061 nor the IEC 61508-1 [2010] safety integrity levels [SIL] are currently supported by the model, although the System function specification is done according to IEC 62061.) Safety function must be represented for the PL evaluation in a manner that cannot be fulfilled by the Structure artefacts model presented in Section 3.3.2. Hence a special set of artefact types is attached to Safety function. A safety-related block diagram needs to be drawn to define the logical structure of the safety function. The diagram illustrates, which safety blocks (i.e., unities of system elements) are logically connected in series and which in parallel in the fault tolerance sense. ISO 13849-1 (2006) (in its Appendix B) gives guidance on creating such diagrams. Such a diagram is in theory drawn for each safety-related part of a

[9] The reason for adopting the IEC 62061 function specification format while otherwise referencing ISO 13849-1 is that ISO 13849-1 does not provide such a systematic function specification template as IEC 62061 does. The specification template is not presented in this report.

control system (SRP/CS)[10], but SEAModel requires a block diagram that combines all the related SRP/CS:s of the safety function instead of a set of single SRP/CS diagrams due to the fact that it is more common to present a combined diagram that has its context in the safety function, not in individual SPR/CS:s. The black box artefacts model presented in Section 3.2 allows linking of Document artefacts to an SRP/CS artefact. Such a document can be a safety manual or a technical manual of an off-the-shelf safety device, such as a safety PLC.

A safety-related part of a control system[11] (SRP/CS) can be a one channel system or a two channel system. Such channels are denoted Normal channels in the model. Test channels may also be defined (only in Category 2 solutions according to ISO 13849-1). The Normal channels and Test channels are special cases of the Channel artefact. Each channel consists of blocks, and blocks consist of system elements.

[10] Very often a safety function is considered to consist of three SRP/CSs: input, logic and output. PL is evaluated for each of them and the combined PL is calculated according to the rules of ISO 13849-1.

[11] Note that the SISTEMA tool by IFA (Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung) calls these ‘subsystems’. As ISO 13849-1 does not use the term ‘subsystem’, we simply call them SRP/CSs and, in fact, SISTEMA treats them as SRP/CSs according to ISO 13849-1 even though it uses diverse terminology.

What actually is a safety function?

*ISO 13849-1 defines the concept of safety function as: "**function of the machine whose failure can result in an immediate increase of the risk(s)**", but in its body text the concept of safety function is not consistent with the definition. In fact, the body text seems to define the safety function as IEC 62061 defines the concept of a safety related control function: "**control function implemented by a SRECS [Safety Related Electrical Control System] with a specified integrity level that is intended to maintain the safe condition of the machine or prevent an immediate increase of the risk(s)**". (Note that the definition for 'safety function' in IEC 62061 is the same as in ISO 13849-1.) We would simply like to say: **a function that is added for the safety reasons**.*

The concept of a safety function can thus be somewhat obscure to a machine engineer, and it may be difficult to identify and specify a safety function. For example, a boom movement is a normal operational function. When limiting its speed to a safe level, the speed-limiting facility can be called a safety function, but it may be difficult to point it out and show where it is because it may simply be a line of application software and a parameter embedded in the application software. Let us think of another safety function called 'prevention of unexpected movement': the boom movement is stopped when the joystick is released to its central position but, for safety reasons, an enabling switch (a 'dead-man's switch') and a hydraulic enabling valve are added. Now the required performance level for the safety function 'prevention of unexpected movement' could be, for example, PL d, which is typically achieved by Category 3 (i.e. two channels) architecture according to ISO 13849-1. What are the two channels? The first one is the normal centre position stop and the second one is the enabling switch – enable valve – channel. Now this leads to the fact that half of the safety function is allocated onto the normal channel and the rest to the additional safety channel. In both of the examples, it is difficult to separate the safety function from the operational function and hence the electrical control system that executes the normal system functions easily becomes a safety-related electrical control system as a whole. In some industry sectors, the safety functions are strictly separated from normal operational functions.

3.3.4 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 15.

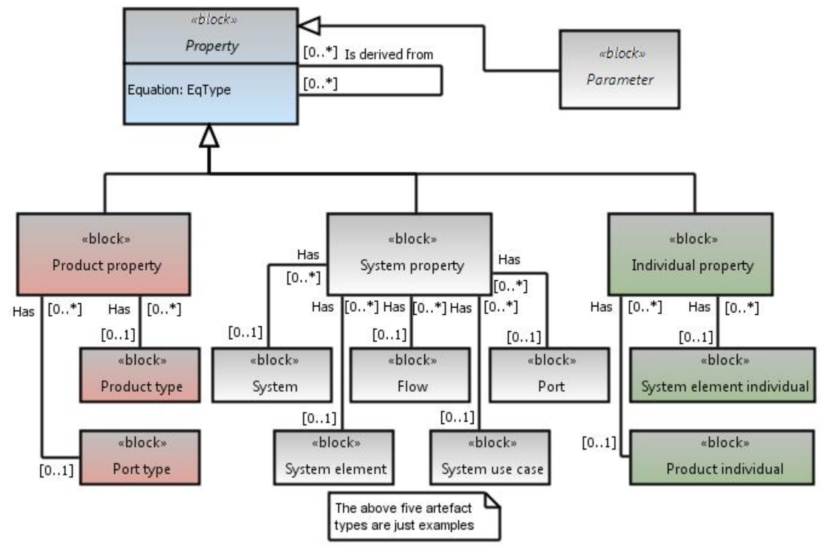


Figure 15. Properties artefacts model diagram.

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 (like 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 called Equation.

A property may need further elaboration. This is achieved by documents, models and diagrams, like a parametric diagram according to SysML. The documents, foreign models and diagrams are linked to the corresponding artefacts as black box artefacts (see Section 3.2).

3.3.5 Product type package

The Product type sub-package only contains one model, the Product type arte-facts model depicted in Figure 16. It covers the description of a released system or component to be applied by a systems integrator or acquirers of a system.

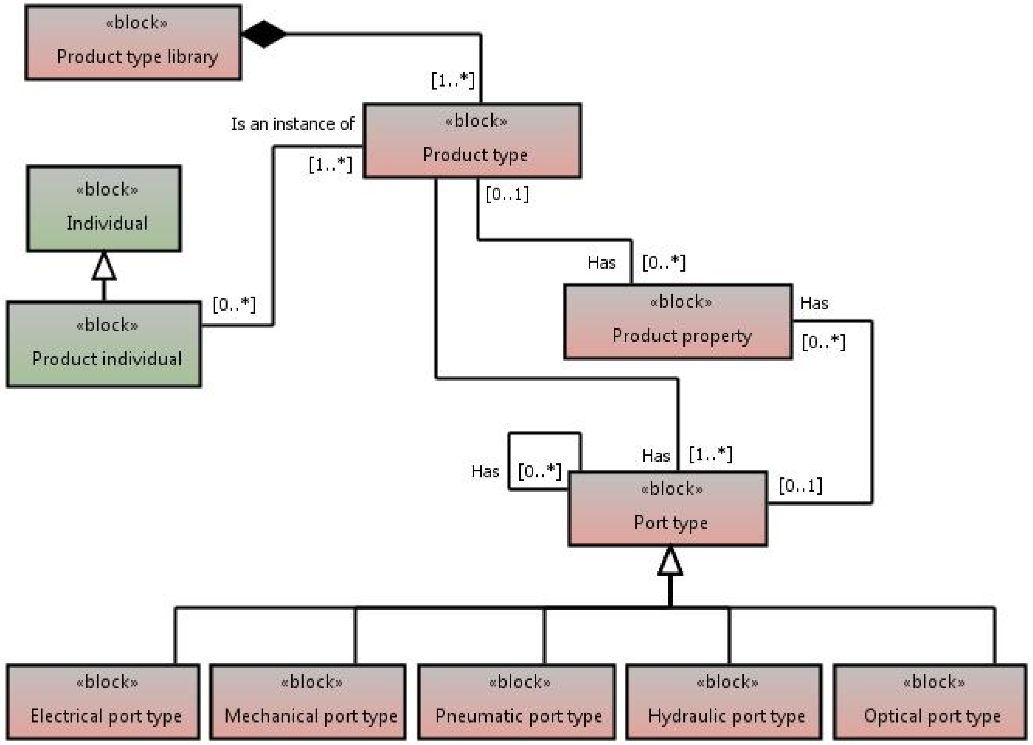


Figure 16. Product type artefacts model diagram.

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 11). In other words, a product type is the physical implementation[12] 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 Section 3.2. 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, and so forth. 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 facilitates automatic embedding of the specification parameters in the application specific documents or drawings generated from the data 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 like serial number and maintenance and warranty information.

3.4 System context package

The System context package only includes the System context artefacts model in Figure 17. It introduces artefact types for description of the context in which the system will be used. The context includes both concrete (like environment and human actors) and abstract issues (like past incidents and glossary).

The model in Figure 17 introduces Constraint artefact to encompass constraints caused by the system context to the system-of-interest. An example is a case in which 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.

[12] '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.

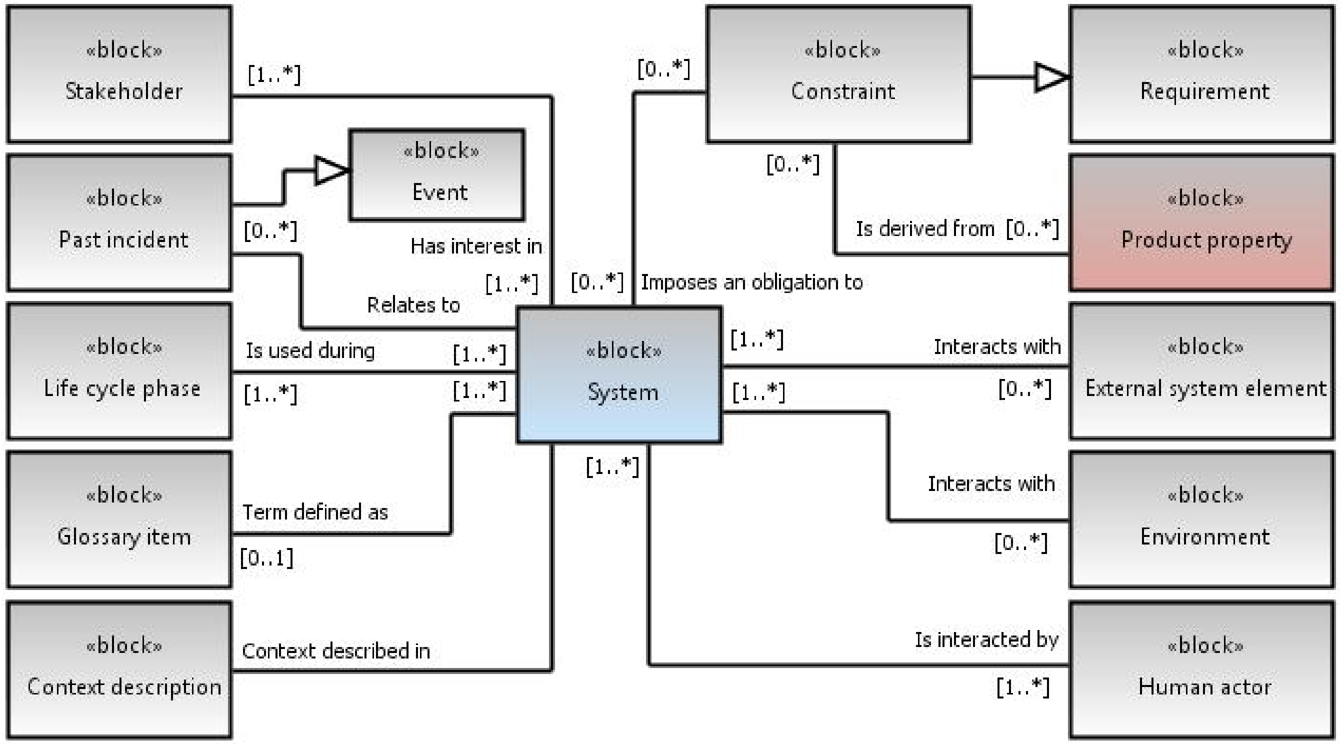


Figure 17. System context artefacts model.

3.5 Requirement and V&V package

The Requirement package consists of one artefacts model, the Requirements and V&V artefacts model, which is presented in Section 3.5.1.

3.5.1 Requirements and V&V artefacts model

The core model for traceability from requirements to design and implementation, from implementation to test execution, and from test execution to verification or validation reporting is presented in Figure 18[13]. We call this the 'core loop of systems engineering'. This model is the starting point for SEAModel implementation.

[13] 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 5.

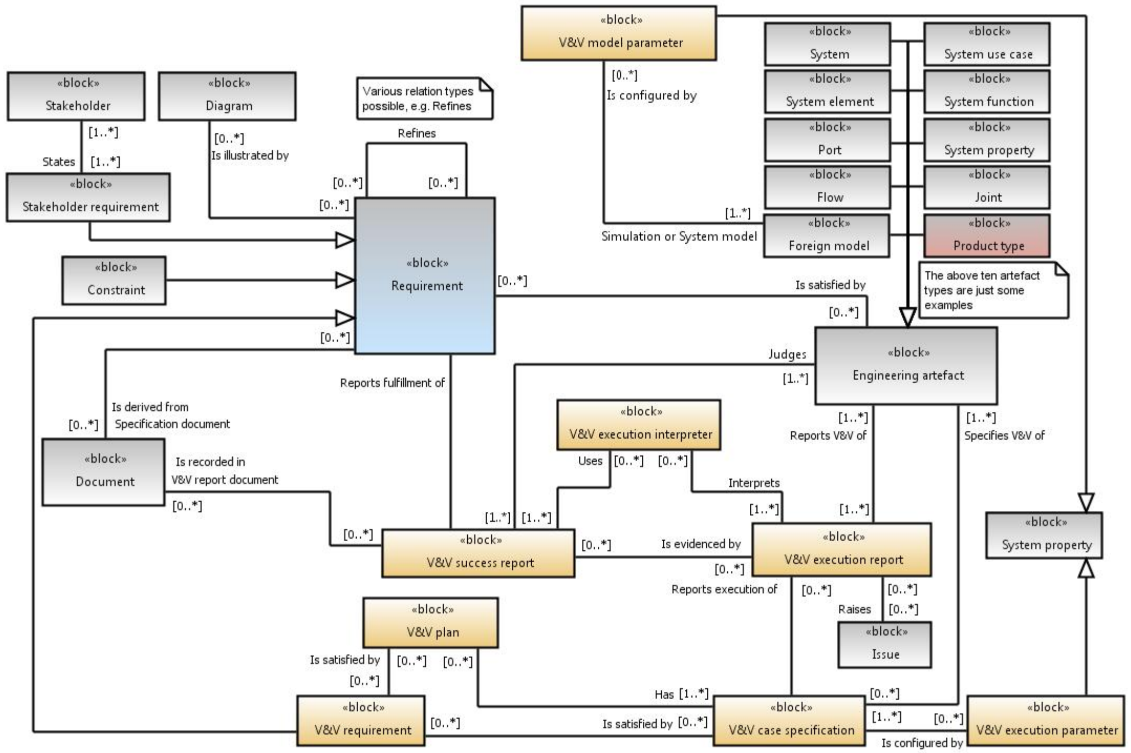

«block» V&V model parameter
[0..*] Is configured by
«block» System
«block» System use case
«block» System element
«block» System function
«block» Port
«block» System property
«block» Flow
«block» Joint
«block» Foreign model
«block» Product type
[1..*] Simulation or System model
The above ten artefact types are just some examples
«block» Stakeholder
«block» Diagram
Various relation types possible, e.g. Refines
Refines
[1..*] States [1..*]
[0..*] Is illustrated by
«block» Stakeholder requirement
[0..*] [0..*] [0..*]
«block» Requirement
«block» Constraint
[0..*] Is satisfied by [0..*]
[0..*]
Judges [1..*]
«block» Engineering artefact
Reports fulfillment of
Is derived from Specification document [0..*]
«block» V&V execution interpreter
[1..*] Reports V&V of
[1..*] Specifies V&V of
«block» Document
Is recorded in V&V report document [0..*]
Uses [0..*] [0..*]
Interprets
[1..*] [1..*] [1..*] [1..*]
[0..*]
«block» V&V success report
[0..*] Is evidenced by [0..*]
«block» V&V execution report
«block» System property
Reports execution of [0..*]
[0..*] Raises [0..*]
«block» V&V plan
«block» Issue
Is satisfied by [0..*] [0..*] [0..*]
Has [1..*]
[0..*]
«block» V&V requirement
[0..*] Is satisfied by [0..*]
«block» V&V case specification
[1..*] [0..*] Is configured by
«block» V&V execution parameter

Figure 18. Requirements and V&V artefacts model diagram.

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 level system requirements are derived from the stakeholder requirements[14]. A requirement can have child requirements that refine the parent requirements. In this case, the relationship name is 'Refine', but other relationships can also be created. Diagrams, documents and models can be linked to the requirements where necessary as described in Section 3.2.

The engineering artefacts[15] 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 success report. A V&V case specifies tests, analyses, simulations or any other methods to evidence fulfilling the requirements for the design or project processes[16]. A V&V execution report must be traced to the artefact under verification or validation to prompt for a test (or analysis or any other method) re-execution if the artefact under test is updated. The V&V execution interpreter provides a means to relate a V&V success report with instructions or tools to interpret the V&V execution results, like 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 success report artefact. It may be necessary to execute more than one test or analysis case for the evidence of a successful result of the validation. Hence the V&V success report 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 call for corrective actions. The issue is managed according to the modification procedure of the project.

V&V plan collects a set of V&V case specification to form a specific sequence of tests for a specific purpose, such as for Factory Acceptance Test (FAT).

The validation model is designed to work with the ISO 13849-2 (2012) validation procedure. For example, when validating the category of a safety function,

[14] 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).

[15] 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 18 depicts only some of such.

[16] Project process artefacts are not modelled in this report. Nevertheless, artefact types like Project process, Project activity and Project task can easily be incorporated into SEA-Model.

according to ISO 13849-2, observance of the basic safety principles must be inspected. The basic safety principles of electrical systems are listed in Appendix D of ISO 13849-2. Let us consider the following basic safety principle: *Proper selection, combination, arrangements, assembly and installation of components/system*. The particular issue is written into the V&V case specification artefact, e.g. as: *Inspect that components/systems are properly selected, combined, arranged, assembled and installed*. In ISO 13849-2, there are several such tasks, not only the basic and well-tried safety principles that can be presented as test or analysis cases. The fault modes to be considered can also be presented as test or analysis cases, but SEAModel provides dedicated artefact types for fault mode types and for the actual fault modes (see Figure 19).

3.5.2 Modelling and simulation artefacts in SEAModel

In this section, a short discussion is provided on management of modelling and simulation artefacts by SEAModel.

The design and simulation models are typically created by CAD tools, SysML tools, FEM analysis tools, simulation tools and the like. Such models are stored as foreign models in the SEAModel data repository, i.e. they are not stored as structured model elements, but as separate files.

The Foreign model artefact can be linked to several artefacts, such as System, Requirement, System use case, System task, System function, System element and Product type. It is not defined here, which type of models can be incorporated into a Foreign model artefact, but it is assumed that non-structured text and ad-hoc (informal) diagrammatic representations are not considered here as models. For those purposes, two artefact types are defined, Document and Diagram.

In general, the simulation artefacts constitute of the following:

- **simulation requirements**: the requirements, which state that simulation has to be carried out; more detailed requirements about the simulation execution; the requirements can include rationale for the simulations
- **simulation plans**: the plans for the simulation process, what is the purpose (the rationale) of the simulations, what are the items under simulation, simulation strategy, list of simulation cases, simulation schedule, simulation personnel
- **simulation case specifications:** the rationale for the simulation case, the exact specifications of the simulation steps, list of tools to perform the simulation case, environment of the simulation, expected results
- **simulation model:** an executable, purpose driven, view of the item (like a CAD model) under verification; note that a special simulation model is not needed if the actual model supports simulation of the characteristics under verification
- **simulation model parameters:** the parameters that configure the simulation model element under verification for the simulation; normally, the parameters of the design under analysis are changed first, and thereupon the simulation model parameters are updated accordingly, but in cases in

which the actual model does not support simulation, like a physical prototype, the simulation model parameters are changed during the simulation, and the parameters of the physical model may be updated according to the simulation results

- **simulation execution parameters:** the parameters relating to the execution of the simulation, like the number of simulation runs
- **simulation results**: the simulation results can include virtual measurement data, animation videos, pictures or sound files
- **simulation results interpreter**: an instruction document or a program that provides the means for interpreting the simulation results, like a program that plots a graph of the virtual measurement data.

Besides the artefacts above, the common artefacts like requirements and design artefacts are relevant during the simulation process. The design artefacts are the ones that are verified or validated by the simulation cases, and the requirements are the ones against which the justification of correct design, based on the simulation results, is done.

The following table outlines, how the simulation artefacts above are mapped onto SEAModel. The relevant model in SEAModel is the requirement artefacts model presented in Section 3.5.1.

Table 1. Mapping of simulation related artefacts onto SEAModel.

Simulation artefact	Corresponding artefact type in SEAModel
Simulation requirement	V&V requirement
Simulation plan	V&V plan
Simulation case specification	V&V case specification
Simulation model	Foreign model (in most cases)
Simulation model parameters	V&V model parameter
Simulation execution parameters	V&V execution parameter
Simulation results	V&V execution report
Simulation results interpreter	V&V execution interpreter

3.6 Specialty engineering package

In principle, 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 designed (see Section 3.6.1).

3.6.1 Risk Assessment package

The Risk assessment sub-package only contains the risk assessment artefacts model presented in Figure 19. It is based on the ISO 12100 (2010) risk assessment model depicted in Figure 20.

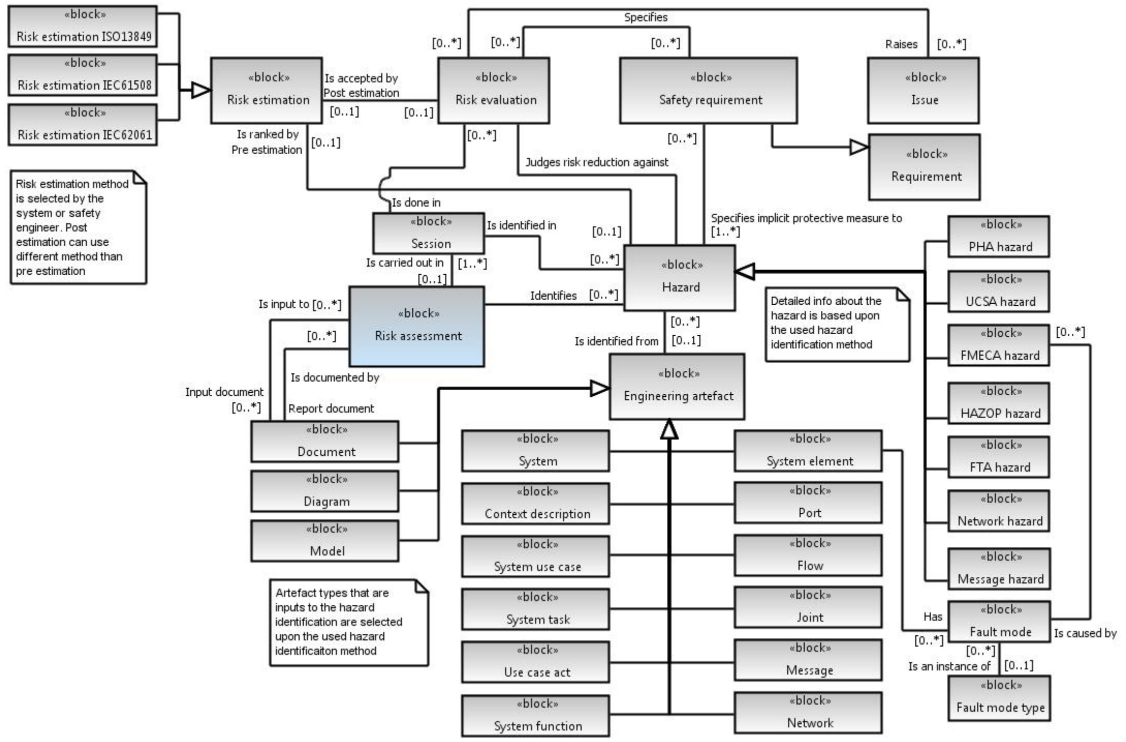


Figure 19. Risk assessment artefacts model diagram.

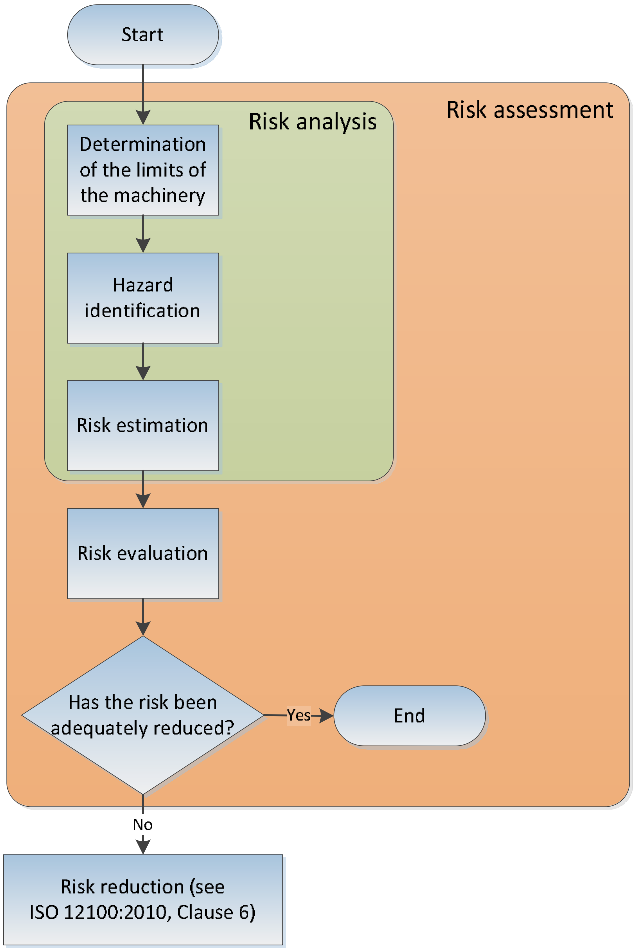

Start
Risk assessment
Risk analysis
Determination of the limits of the machinery
Hazard identification
Risk estimation
Risk evaluation
Has the risk been adequately reduced?
Yes
End
No
Risk reduction (see ISO 12100:2010, Clause 6)

Figure 20. 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), Use Case Safety Analysis (UCSA), Failure Mode Effects and Criticality Analysis (FMECA), Fault Tree Analysis (FTA), Hazard and operability study (HAZOP), message safety analysis and network safety analysis. FMECA is supported by two additional artefact types, Fault mode[17] and Fault mode type.

A risk assessment is performed in several analysis sessions the minutes of which are recorded in the Session artefacts. During the PHA-, UCSA- and other 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 design artefacts:

- System-of-interest (typically for PHA)
- System context related artefacts (typically for PHA)
- System use case (typically for PHA and UCSA [the supplemented use case])
- System task (typically for PHA and HAZOP)
- Action (typically for UCSA 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
- UCSA 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 the risk estimation methods of IEC 61508, IEC 62061 and ISO 13849-1 are supported. 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 risks caused by the identified hazard. The existing protective measures must be evidenced in the form of existing safety requirements

[17] Often an alternative phrase, Failure mode, is used.

and linked to the particular hazard. For example, 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 3.5.1.

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 the machine with a collision avoidance system, but the risk evaluation team may find it too expensive to be implemented, and thus 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.

4. Platforms to provide traceability management

Practical implementations of SEAModel require a software tool with the following features:

- structured artefact repository in which relations between the artefacts can be created and maintained
- artefacts traceability with impact analysis
- Version control of artefacts and of a set of artefacts (with “baselines” feature)
- modification control
- automatic or semi-automatic document generation
- document management (or seamless integration to an existing document management system)
- integration possibility with systems engineering tools like requirements management tools, CAD-tools, SW programming tools and the like.
- concurrent engineering capabilities
- collaboration features including wiki-pages, task lists, discussion boards, announcements and the like.
- metrics of various systems engineering issues, e.g. how many of the system requirements are covered by the design artefacts.

Besides the above features, the following issues need to be considered when selecting the tool:

- cost
- responsiveness (especially important when playing with the traceability features)
- usability, user experience.

In SME:s, cost of the licenses becomes one of the most dominant factors in selecting the systems engineering platform. Another critical factor is the possibility to integrate software tools that are already applied in the company. Such tools include requirements management tools, modelling tools, CAD-tools and simulation tools. And yet another important factor is how well the systems engineering platform product (like a PLM tool) fits into the company context including IT infrastructure and competence of the personnel. One of the biggest hurdles is, however, psychological resistance of engineers to move from document based engineering to a way of working in which information is input via graphical model elements or by using forms. Engineers do not feel relaxed if they forfeit their freedom and are forced to work systematically. Hence excellent usability and user experience of the systems engineering platform is a prerequisite for overcoming the hurdle of psychological resistance.

4.1 Product and application life cycle management tools

Typical tools to support at least the core set of the features above are among PLM (Product Lifecycle Management) tools and ALM (Application Lifecycle Management) tools. The main difference between these tool is that PLM tools are CAD-oriented, whereas ALM tools are software development oriented. Examples of PLM tools are ARAS PLM, Dassault Enovia (CATIA V6), Eurostep Share-A-Space, PTC Windchill and Siemens Teamcenter. Examples of ALM tools are IBM RELM[18] (Rational Engineering Lifecycle Management) with other Rational tools, Polarion ALM and Microsoft TFS (Team Foundation Server).

Some of the PLM tools may use fixed underlying data model thus hindering full implementation of SEAModel, but in most cases the data model according to SEAModel can be implemented. There also are a lot of variations as to how traceability with impact analysis is implemented; in some cases the traceability and impact analysis is displayed with sophisticated graphical diagrams, whereas in some other cases tailoring is needed to provide decent user interface for the traceability and impact analysis.

4.2 ModelBus and Traceino

ModelBus is a platform intended for integrating various systems engineering tools to enable interoperability of tools and automation of engineering processes. It is the server side for many of the systems engineering related products being developed at the System Quality Center of Fraunhofer FOKUS. On the client side, Metrino provides model validation and quality assurance, Traceino provides traceability between different models and other artefacts, and Requino provides requirements engineering. Furthermore, ModelBus TeamProvider feature is available for using ModelBus services from Eclipse.

The adapter architecture of ModelBus is specified and supported to allow integration of third party tools by implementing plug-ins for the tool that bridge between the tools' internal data representation and that of ModelBus (see Figure 21). The ModelBus web site describes adapters for IBM Rational DOORS and Rational Rhapsody, Eclipse and Papyrus, Sparx Enterprise Architect, Microsoft Office, and Matlab Simulink. It is notable that no CAD tool adapters are available but one

[18] RELM is actually a Systems Engineering lifecycle management tool.

would have to be custom developed should ModelBus be selected as the integration platform for engineering mechatronic products.

The ModelBus server, TeamProvider and Papyrus adapters are freely available for downloading at the web site. The server is available for Linux, Windows, and OS X, and various releases of Eclipse. The client side software is installed within Eclipse from update sites maintained by the ModelBus team.

Since ModelBus and the client tools would provide significant parts of an integrated platform for systems engineering, we tested the software hoping to build the demonstration described in Chapter 5 on it. The ModelBus DOORS adapter requires Open Services for Lifecycle Collaboration interface of DOORS NG while we use DOORS 9.5; therefore, the DOORS integration could not be tested. We did not get a reply querying about the availability of Matlab Simulink adapter. ProR was used for requirements engineering, Papyrus for SysML modelling, and Traceino for traceability between requirements and SysML model. We tested ModelBus server versions 1.9.8 and 1.9.9 and versions 1.0.0 and 1.1 of Traceino on Eclipse Juno and Eclipse Luna releases, respectively.

While installing of the core ModelBus server is straight forward since it runs simply as a Windows application, installing the client side software within Eclipse required more effort. The Eclipse releases are not frozen but get regular updates for the various components and the updates may break the dependencies of other features unless the features are also updated to support the latest Eclipse components. This is a common issue with Eclipse features which are not actively maintained, and investigating and solving the compatibility issues require some proficiency in administering Eclipse installations. However, we had working installations after some effort.

ModelBus server can be configured and administered either with a web browser based interface or with Eclipse. The administrator can configure users, groups, and access rights to projects in the server. As such, ModelBus provides version control functionality similar to Subversion and technically ModelBus is implemented using Subversion repository as the internal database. ModelBus also provides functionality for locking models for exclusive use by one user at a time.

Traceino supports traceability by allowing the creation of typed links between elements of two or more models. Traceino is not a single application but a framework and a set of views and wizards that can be integrated with third party modeling tools. An implementation of the framework for Eclipse is freely available for download, the availability for other tools in not known.

Traceino framework does not come with a predefined traceability model but the user is free to define and create the traceability meta model with the functionality provided by Eclipse Modeling Framework. While this freedom to customize the meta model to ones requirements is appreciated by experienced users, it comes with the cost of requiring deep understanding of the model hierarchy of the Eclipse Modeling Framework. Once the meta model has been created and committed to ModelBus repository, Traceino can “discover” models of a particular meta model in the repository.

The combination of ModelBus and Traceino was tested by recreating the tutorial example from the Traceino user guide. The example is very similar to what would be needed for the demonstration: A trace type is defined to link a ReqIF SpecObject to an UML Class, and additional meta modelling properties such as cardinalities, link direction, and constraints are defined in the traceability meta model. The completed tracebility meta model must be committed to ModelBus server to be available to other tools connected to ModelBus. Traceability links from classes to requirements can be created and modified in Eclipse by opening the Eclipse views introduced by the Traceino feature, clicking appropriate GUI buttons and model elements. Since SysML is implemented in Eclipse as a specialization of UML, the meta model created would be applicable also for SysML model of the demonstration.

Both the tested ModelBus servers used as a version controlled repository and the Traceino frameworks for Eclipse experienced software instability. The ModelBus server would throw an exception and require restarting. Sometimes the server crash left the underlying database in an inconsistent state rendering it unusable. It became quickly apparent that the underlying Subversion database must be backed up frequently to avoid losing work. Another difficulty was that ModelBus apparently uses the Subversion database for internal operations. At times, ModelBus server did one of the internal operations while the model was also being edited in Eclipse. This caused a conflict when attempting to commit the edited model back to ModelBus server. Eclipse would require manual resolution of the conflict at XML file level, which is of course highly undesirable from the user's point of view.

The ModelBus manager web application enables management of the traceability links directly at the ModelBus server. The manager provides a centralized point of view of the models stored at the server and might be a tool for managing links between models created with tools other than those based on Eclipse. The manager is freely available for downloading but could not be tested since installation failed due to lack of instructions.

While ModelBus concept is very attractive from the point of view of model based systems engineering and the approach would fit very well with the goals of the demonstration, the implementation level at the time of the evaluation was not mature enough to support research work let alone commercial work. High level of ICT proficiency would be needed to manage the installations and work around the issues in the available software. Such skills would likely not be easily available in a typical SME doing design and engineering of mechatronic products. Also, adapter development would be necessary for tracing to CAD models.

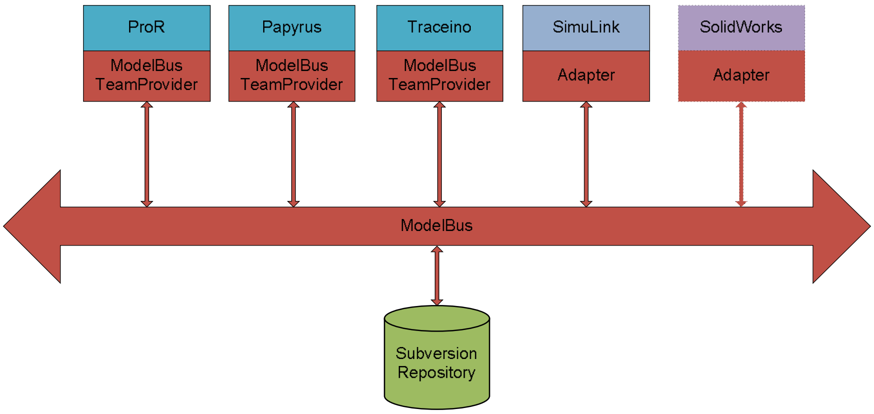


Figure 21. ModelBus and Traceino architecture. SolidWorks adapter is not available at the time of the evaluation.

4.3 Simantics

Simantics is an integration platform for modelling and simulation software tools. Its database architecture is an ontology based graph storage with a very fine data granularity. Compared to a relational database, Simantics database does not contain records with several fields, but each field is a data item of its own (a resource in the Simantics terminology or a node in the graph theory terminology). The idea is to chop the models from a modelling or simulation tool to atomic pieces of data to facilitate sharing of the data between the tools. To integrate a modelling or simulation tool (or any other tool), adaptation work is needed. The form of adaptation depends on the tool under integration and whether on-line or off-line integration is selected.

Because of the fine grained data integration facility for the modelling and simulation tools, Simulink is an interesting platform candidate for SEAModel implementation. The Simantics architecture allows for arranging traceability between e.g. requirements and model elements. At the time of the SIMPRO project, however, such a feature was not available. Furthermore, version control feature was under work during our research. Hence we decided to restrict our evaluation to integrate a requirement management tool with Simantics. In fact, the general requirements management file format ReqIF was selected as the data source for the integration to serve a number of requirements management tools, not only a single tool. ReqIF data could be imported to Simantics after a moderate integration software programming; the ReqIF import mechanism was made generic to accept any kinds of XML files that are specified by XSD schemas. However, to provide full requirements roundtrip, new features (such as version control) for Simantics are required (Mätäsniemi & Alanen 2015). A special research report on the ReqIF trial was written (Mätäsniemi & Alanen 2015).

4.4 Others

Some of the needed features can be achieved by a requirement management software or with database oriented content management systems (CMS) like MS SharePoint. Both of these provide a relational database such that data items (i.e. engineering artefacts) can be linked to another to provide traceability. CMS tools, however, do not intrinsically provide impact analysis, i.e. suspect flagging in case one of the engineering artefacts at the one end of a trace link is edited; professional requirements management tools provide this feature. The advantage of CMS tools lies in the fact that such are often already in place in a company, and hence the investments to licenses are minimal if not zero. But the amount of work to tailor the CMS application to support systems engineering is remarkable.

In both cases, i.e. with requirements management tools and CMS, there is practically no ready provision for mechanical CAD and simulation tool integration. With a normal URL-linking, CAD and simulation files can be linked to other artefacts such as requirements and test case specifications. With programming, such links can be enhanced to provide impact analysis. We demonstrated such method between SolidWorks CAD tool and Rational DOORS requirements management tool (see Section 5.3.5). The trick was to create surrogate objects at DOORS of each of the CAD model files; when a CAD model element is updated (and hence its file is updated) the programmed script compares the date and time of the surrogate object and model file, and copies the new timestamp to the surrogate object if the timestamps are different. The corresponding surrogate object is thus touched causing the suspect flag of a trace link to be raised, if e.g. a requirement is linked to the surrogate object that represents the model file of the changed CAD model element.

In principle, any relational database can be used to create a systems engineering data repository according to SEAModel. We have demonstrated such with a MySQL database (with MS Access user interface). The demonstration is reported in Alanen et al. (2011). Such a raw database implementation requires a lot of work to reach the level of CMS tools in regards to collaboration features and document management support. Hence a pure relational database as a starting point is not recommended.

5. Traceability demonstration

5.1 Traceability information model

Traceability makes it possible to find out how and why certain design solutions have been derived. It requires some additional 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. Such models can easily be derived from the artefacts models in Chapter 3. 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.

Below in Figure 22, an example traceability information model derived from the Requirement and V&V artefacts model (see Section 3.5.1) is provided.

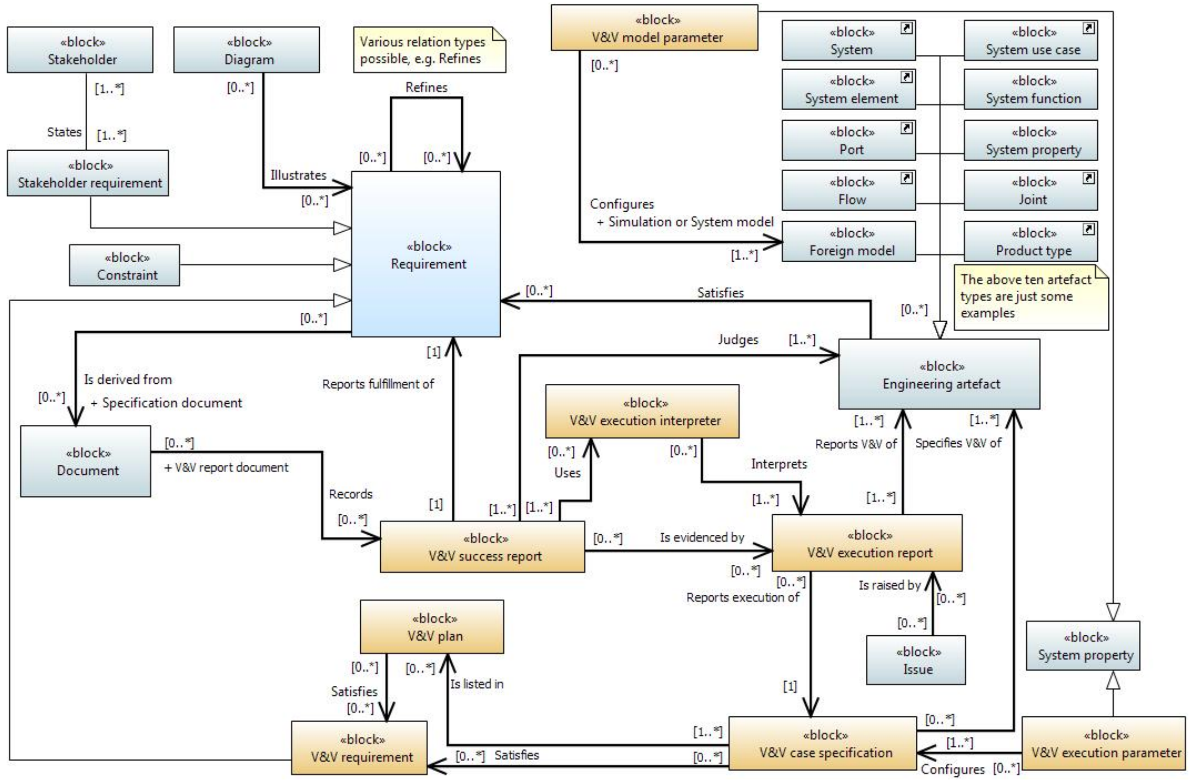


Figure 22. Traceability information model for requirements and V&V artefacts.

The relationship names in Figure 18 are updated to their reciprocal versions in Figure 22 in some cases to provide reading order from the source to target direction. The arrowed trace links point 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 target. As can be seen in Figure 22 compared to Figure 18, all the other relations except the relation between Stakeholder and Stakeholder requirement[19] manifest themselves as trace links.

In a similar fashion, all the other concept diagrams in Chapter 3 can be converted to traceability diagrams. In the following demonstration case (see Section 5.2), however, mainly the TIM in Figure 22 is applied.

The TIM is applied to each system, whether a system-of-interest or a subsystem developed in-house. For off-the-shelf subsystems 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 subsystem or part manufacturer to be stored into the *Product type* artefact. In the 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.

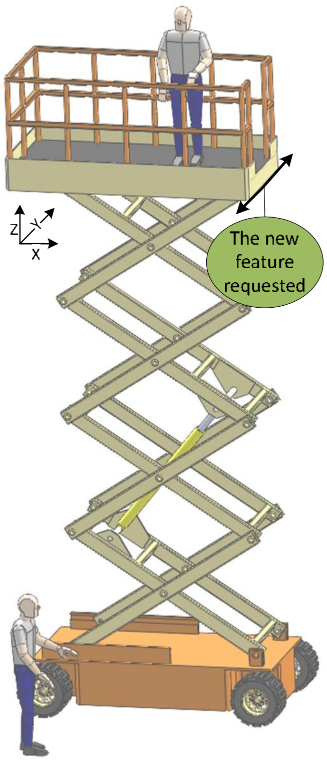
To implement the traceability information model, the data repository platform has to support traceability management including impact analysis.

[19] 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. However, updating the stakeholder requirements in that case is not initiated by the trace analysis (i.e. impact analysis), but by more powerful project procedures.

5.2 Definition of the demonstration

To demonstrate traceability according to the models in Chapter 3 and especially the TIM in Section 5.1, the following systems engineering work flow scenario was defined:

Overall description of the scenario



A company is manufacturing mobile elevating work platforms (MEWP) of scissors type. One of its customers requests a new feature to be able to move the platform of the machine horizontally (sideways) in order to be able to reach closer to a wall in case the chassis of the machine cannot be parked close to the wall. The systems engineer and the safety engineer of the MEWP company analyses the request and create system requirements (including safety requirements) accordingly. The systems engineer sketches the architecture of the updated mechanical design of the platform. A mechanical engineer of the MEWP company creates a CAD model to implement the architecture design. The CAD model is simulated according to a simulation case specification set by the systems engineer or the mechanical engineer. To do that, a simulation model is created by the mechanical designer or a test engineer out of the CAD model. The results of the simulation execution are compared to the requirements, and judgment about the conformity of the CAD model with the requirements is issued. (See Figure 35, Figure 36 and Figure 37 in Appendix B.)

Actors (see Figure 38 in Appendix B)

- Customer
- Systems Engineer
- Safety Engineer
- Mechanical Engineer
- Test Engineer

Tools (see Figure 38 in Appendix B)

- Requirements Management Tool
- SysML Tool
- CAD Tool
- Simulation Tool
- V&V Tool

- Integration and Traceability Platform (ITP)

Engineering artefacts (see Figure 39 in Appendix B)

- Stakeholder requirements
- System requirements
- PHA Hazards
- Risk evaluations
- Use cases
- Architecture model
- CAD model
- Verification requirements
- Simulation case specification
- Simulation model
- Simulation results
- Requirements trace
- Verification report.

Detailed scenario steps

1. Requirements acquisition (see Figure 40 in Appendix B)
    1.1. The Systems Engineer captures the Stakeholder requirements from the Customer and stores them to the Requirements Management Tool.
    1.2. The Systems Engineer analyses the Stakeholder requirements. The Safety Engineer performs a Preliminary Hazard Analysis (PHA) and reports the PHA Hazards and their Risk evaluations. The Systems Engineer creates corresponding System requirements (including the safety requirements) and stores them to the Requirements Management Tool. The Use cases are created based upon the functional requirements (that are among the System requirements).
2. Architecture design (see Figure 41 in Appendix B)
    2.1. The Systems Engineer designs the Architecture model of the system using the SysML Tool and traces with the ITP the Architecture model elements to the System requirements where relevant.
    2.2. The Systems Engineer issues the System requirements and the Architecture model.
3. Detailed design (see Figure 42 in Appendix B)
    3.1. The Mechanical Engineer creates, using the CAD tool, the CAD model according to the Architecture model and the System requirements and traces with the ITP the CAD model elements to the Architecture model elements and to the relevant System requirements.
    3.2. The Mechanical Engineer issues the CAD model.
4. Verification (See Figure 43 in Appendix B)
    4.1. Verification specification (See Figure 44 in Appendix B)
        4.1.1. The Systems Engineer creates the Verification requirements to verify the CAD model, and traces with the ITP the Verification re-

quirements to the corresponding System requirements and Architecture model elements.

4.1.2. The Systems Engineer issues the Verification requirements.

4.1.3. Simulation case specification (See Figure 45 in Appendix B)

4.1.3.1. The Mechanical Engineer or Systems Engineer creates The Simulation case specification according to the Verification requirements and stores them to the V&V tool and traces with the ITP the Simulation case specification to the Verification requirements and to the CAD model elements under simulation.

4.1.3.2. The Mechanical Engineer or Systems Engineer issues the Simulation case specification.

4.2. Verification execution (See Figure 46 in Appendix B)

4.2.1. Create and run simulation (See Figure 47 in Appendix B)

4.2.1.1. The Mechanical Engineer or Test Engineer creates the Simulation model of the CAD model, using the Simulation Tool, such that the simulation according to the Simulation case specification can be executed, and traces with the ITP the Simulation model to the CAD model and to the Simulation case specification, which the Simulation model was created for.

4.2.1.2. The Mechanical Engineer or Test Engineer issues the Simulation model.

4.2.1.3. The Mechanical Engineer or Test Engineer executes the simulation, using the Simulation tool, and records the Simulation results to the V&V tool, and traces with the ITP the Simulation results to the Simulation case specification and to the Simulation model under execution.

4.2.1.4. The Mechanical Engineer or Test Engineer issues the Simulation results.

4.2.2. Judge conformity (See Figure 48 in Appendix B)

4.2.2.1. The Systems Engineer or Mechanical Engineer evaluates the Simulation results and judges conformity of the CAD model to the relevant System requirements, and records it as 'pass' or 'no pass' into the Verification report in the V&V tool, and traces with the ITP the Verification report to the CAD model and to the relevant System requirements and to the Simulation results that evidence the verdict.

4.2.2.2. The Systems Engineer or Mechanical Engineer issues the Verification report.

The workflow scenario is depicted in a semi-formal way in Appendix B.

5.3 Implementation of the demonstration

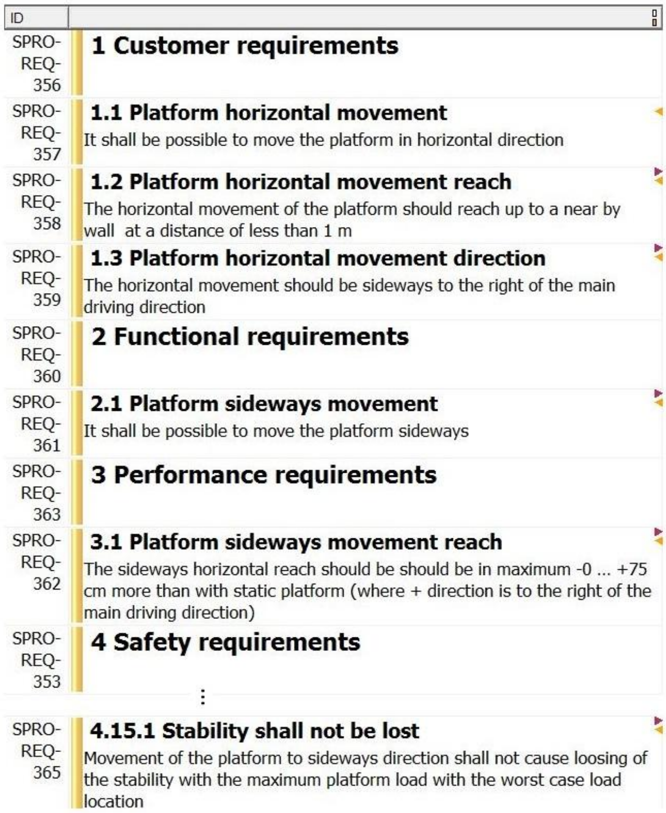
The work activities presented in Section 5.2 and in Appendix B are presented in more detail in the following sub-sections. Further details about the implementation of the demonstration can be found in (Tikka 2015).

5.3.1 Requirements in DOORS

The first step of the scenario concerns requirements acquisition. IBM Rational DOORS is a requirements management platform for requirements specification and requirements engineering collaboration. DOORS stores stakeholder requirements such as the new customer feature request about the platform horizontal movement, but also the safety requirements that are resulted by the PHA. The customer request is recorded as a stakeholder requirement of which a corresponding system requirement is created after analysis. The safety requirements are recorded among the other system requirements.

The created system requirements can be categorised for instance as functional, performance, maintenance or safety requirements depending on their nature.

In addition to the mentioned customer requirement, a set of requirements for the case study are derived from the Machinery Directive 2006/42/EY, from the harmonised C-type standard EN 280 and from some other relevant standards for machinery safety. An excerpt of the DOORS requirements module is provided in Figure 23.



ID	
SPRO-REQ-356	**1 Customer requirements**
SPRO-REQ-357	**1.1 Platform horizontal movement** It shall be possible to move the platform in horizontal direction
SPRO-REQ-358	**1.2 Platform horizontal movement reach** The horizontal movement of the platform should reach up to a near by wall at a distance of less than 1 m
SPRO-REQ-359	**1.3 Platform horizontal movement direction** The horizontal movement should be sideways to the right of the main driving direction
SPRO-REQ-360	**2 Functional requirements**
SPRO-REQ-361	**2.1 Platform sideways movement** It shall be possible to move the platform sideways
SPRO-REQ-363	**3 Performance requirements**
SPRO-REQ-362	**3.1 Platform sideways movement reach** The sideways horizontal reach should be should be in maximum -0 ... +75 cm more than with static platform (where + direction is to the right of the main driving direction)
SPRO-REQ-353	**4 Safety requirements**
	⋮
SPRO-REQ-365	**4.15.1 Stability shall not be lost** Movement of the platform to sideways direction shall not cause loosing of the stability with the maximum platform load with the worst case load location

Figure 23. An excerpt of the DOORS requirements module.

The original customer request is expanded to three customer requirements that explain the feature request in detail: the platform shall move in horizontal direction, the movement of the platform should reach a distance less than 1 m and the movement direction should be to sideways the right of the main driving direction. A matching functional requirement for the requested feature is: “It shall be possible to move the platform sideways”, which emphasises the possibility to move the platform. A technical request for the performance of the MEWP is the platform sideways movement reach, which complements the previous functional requirement. An important aspect for the design and manufacturing of the MEWP is safety. The new feature should not jeopardise the structural stability while the platform is moving horizontally. Therefore, the stability of the MEWP is simulated to inspect, whether the stated safety requirement is compromised and possible stabilisers needs to be added to the chassis. The simulation model is introduced and studied later in Section 5.3.4.

5.3.2 Initial physical architecture – the logical model

The new physical architecture of the MEWP platform subsystem, based upon the customer requirements and the consequent system requirements, is sketched by creating a logical model of the physical architecture. The model is designed with the modelling tool Papyrus, which is accessed through Eclipse Luna. Papyrus is used because it offers support for modelling languages such as UML and SysML, and is free of charge. The logical model introduces abstractions of the physical system elements such that the required functions can be allocated onto them, but the actual physical implementation is left for the CAD designer. The top level diagram of the logical model is provided in Figure 24.

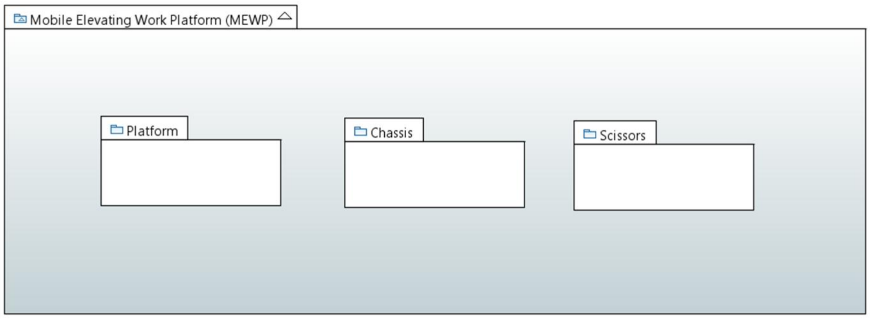


Figure 24. The top level diagram of the logical model.

The main focus is on the Platform subsystem, which the new customer feature is allocated to. The other two subsystems are Chassis and Scissors. The logical architecture of the Platform is captured in Figure 25 by a block definition diagram.

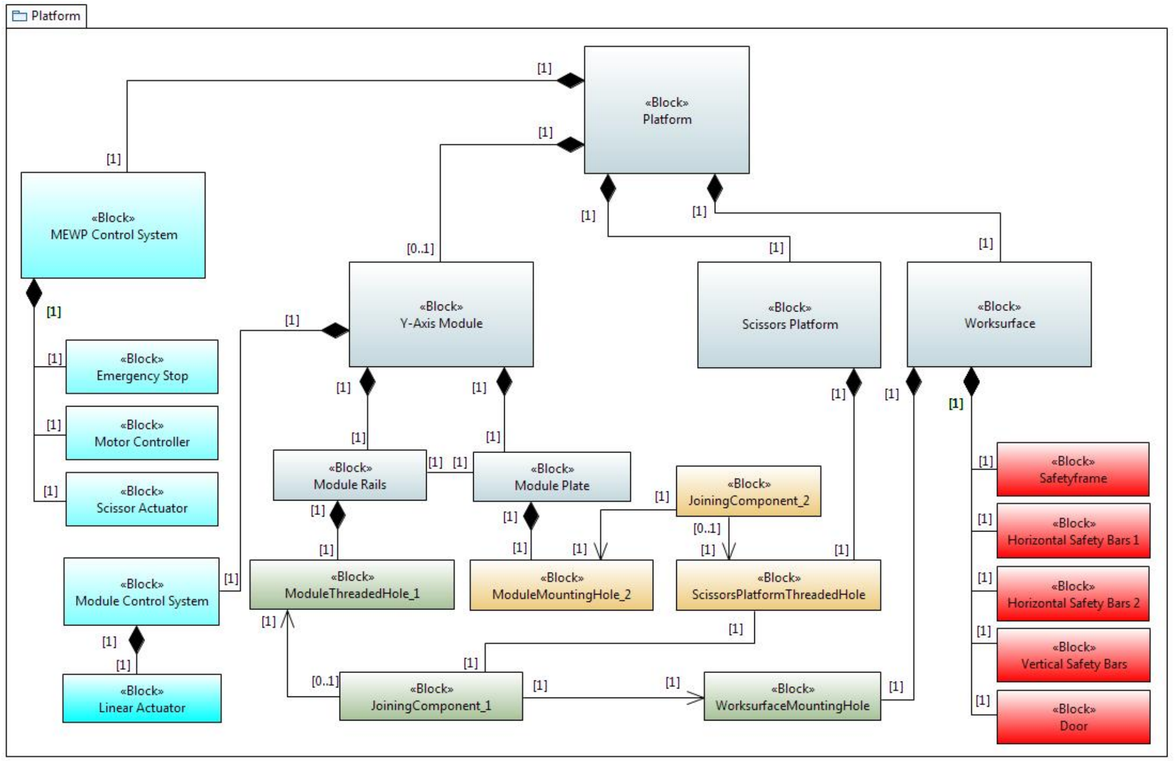

Platform
«Block» Platform
«Block» MEWP Control System
«Block» Emergency Stop
«Block» Motor Controller
«Block» Scissor Actuator
«Block» Module Control System
«Block» Linear Actuator
«Block» Y-Axis Module
«Block» Module Rails
«Block» Module Plate
«Block» ModuleThreadedHole_1
«Block» ModuleMountingHole_2
«Block» JoiningComponent_2
«Block» JoiningComponent_1
«Block» Scissors Platform
«Block» ScissorsPlatformThreadedHole
«Block» WorksurfaceMountingHole
«Block» Worksurface
«Block» Safetyframe
«Block» Horizontal Safety Bars 1
«Block» Horizontal Safety Bars 2
«Block» Vertical Safety Bars
«Block» Door
[1]
[0..1]

Figure 25. Decomposition of the updated platform mechanics – The logical model of the platform architecture.

The Platform block in Figure 25 is the top-level block and provides the context for the new feature. Platform is composed of four main components, Y-Axis Module, Scissors Platform, Work surface and MEWP Control System. The y-axis (i.e. horizontal) movement feature of the platform is made possible by a modular solution: The feature can be added according to a customer request to the structure of the MEWP by including the y-axis module between the sub platform (Scissors Platform), where the scissors are attached to, and the work surface. The y-axis module includes a module plate that moves on the frame of the module rails. Actuation of the module is carried out by a linear actuator, but the more specific method is out of the scope of this report. Because the module is optional, multiplicity at the part end of the composite association is 0...1. Also the associations regarding the attachment of the module are marked as 0...1. Joining of the module is achieved with joining components that will be mounted to the mounting holes and threaded holes in the work surface and scissors platform. If the module is not used, the scissors platform is directly connected to the work surface.

5.3.3 CAD model by SolidWorks

After the logical architecture is defined to satisfy the stakeholder requirements, a physical implementation model is created. The physical model is created by a mechanical engineer according to the logical model and the system requirements provided by the systems engineer. The system requirements are acquired from the requirements management tool, which was chosen to be Rational DOORS (see Section 5.3.1). The goal of the CAD model is to satisfy with a physical representation the specifications and demands set by the customer and systems engineer. The model is produced with a CAD tool, which was chosen to be SolidWorks for the demonstration. SolidWorks was chosen because of its integration possibilities with Simulink modelling tool that was chosen for the simulation.

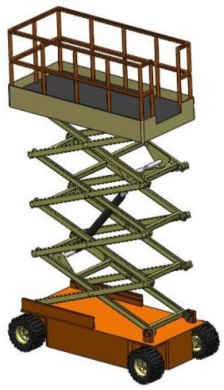
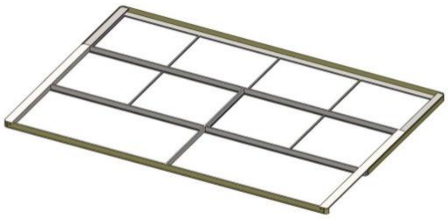



Figure 26. The original MEWP CAD model and the added y-axis module.

The original platform (see Figure 26) was modified into two different sub platform structures to enable the new y-axis (i.e. horizontal) movement add-on. The y-axis module is a frame component (see Figure 26) that can be connected between the two sub platforms. The updated version of the mechanical structure is depicted in Figure 27.

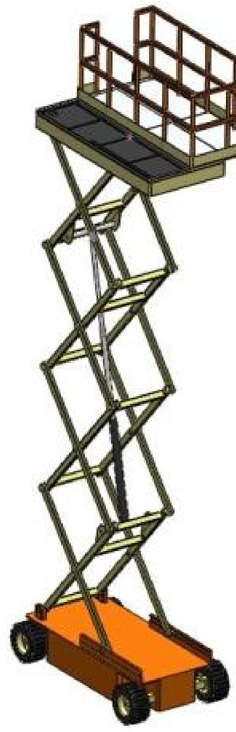


Figure 27. The updated CAD-model based upon the customer feature request.

5.3.4 Simulation by Simulink

The CAD model is verified according to the verification requirements that are derived by the systems engineer from the corresponding system requirements (including safety requirements) and architecture model elements. The verification requirements are classified as process requirements. In this demonstration, as loss of stability is identified as a potential hazard for the revised MEWP, a verification requirement to check the stability of the structure with a simulation tool is issued.

The issued verification requirements are used to create the simulation case specifications that define the rationale for the simulation case, the exact specifications of the simulation steps, list of tools to perform the simulation case, environment of the simulation and the expected results for the simulation. These specifications are set to test the model in the environment or situation, in which the observed hazard can manifest. The simulation case specification defines the height and distance of the platform, general position of the MEWP and workload. Furthermore, the standard EN 280 defines a set of wind conditions, manual forces and distributions of loads and forces that together create the conditions of minimum stability and unfavourable stresses.

The simulation model is developed from the CAD model by a mechanical or a test engineer; the model is done according to the simulation case specification. The simulation software used for the model is Simulink SimMechanics 2nd generation version. SimMechanics offers a plug-in option to facilitate direct export of SolidWorks CAD assemblies into SimMechanics. The plug-in enables one to use within SimMechanics the body specifications like inertial properties, constraints and coordinate systems defined in SolidWorks.

The stability loss -hazard is simulated with a model in which the observed tipping direction is the direction of the extended work platform. The simulation model consists at the top level of three main subsystems, Chassis, Scissors and Platform. Additionally, there are two subsystems for tools and personnel, which together make up for the maximum 900 kg workload set as a requirement. Figure 28 illustrates the top level of the simulation model with the before mentioned subsystems.

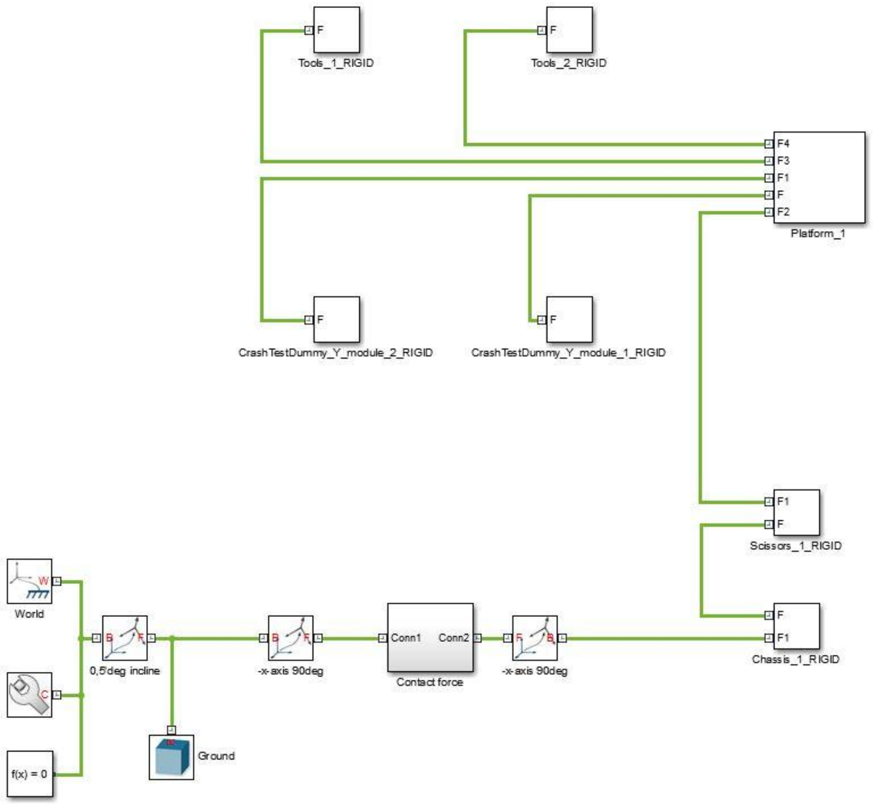


Figure 28. The simulation model for the scissors MEWP – The top level model.

The MEWP is positioned according to the EN 280 standard in the most unfavourable position with the maximum allowed inclination of the chassis. The machine is also under wind loads and side forces like manual force, which try to tip the structure. The tipping lines are determined to be at ¼ of the tyre ground contact width from the outside of the ground contact width. The wind forces are assumed to act horizontally, and they are applied at the centre of area of each structural component, person and equipment.

With the exception of Platform, all the subsystems are made rigid to measure only the stability of the MEWP while the platform is moving and to keep the model complexity as low as possible. Rigid components do not have joints or sensors, and they merely support the integrity of the ensemble.

After being issued, the simulation model is executed and visualised (see Figure 29), and the results are recorded to the verification and validation tool within the ITP.

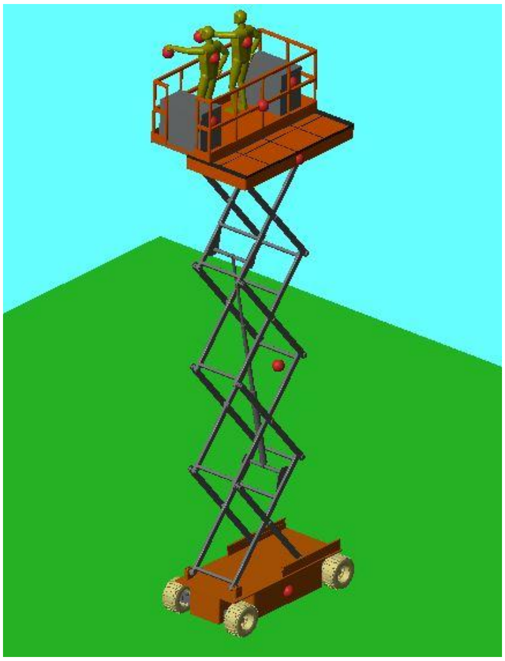


Figure 29. The visualisation of the simulation model depicted in the full horizontal extension.

5.3.5 Traceability by DOORS

The traceability was implemented using IBM Rational DOORS (version 9.5) as the traceability platform. DOORS accommodates the requirements, but also the logical architecture model of the system under demonstration (i.e. a SysML tool is only used to provide the diagram of the logical architecture). Furthermore, DOORS is used to trace the CAD model and simulation model artefacts by using the surrogate object method, in which the external model files are linked to the requirements (and to the logical architecture model elements) via surrogate objects (see Figure 30). A surrogate object represents in the DOORS environment the actual artefact of a foreign tool: The requirement objects are traced to the surrogate objects, and the surrogate objects are linked by a URL to the external file the surrogate objects represents. The impact analysis provided by DOORS does not cover changes in external files. Hence a DOORS script in DXL language was written that checks the date and time of the external files linked to the surrogate objects, and updates a custom attribute (DateAndTime) if the date and time of the external file has been changed. This provides the necessary feature that when an externally linked file is touched, the surrogate object is also touched. Executing the script assists thus in the impact analysis such that also changes in external files are covered by the analysis.

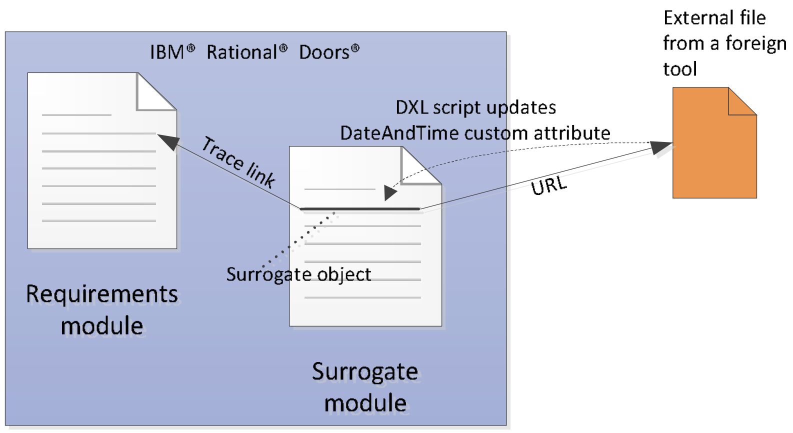
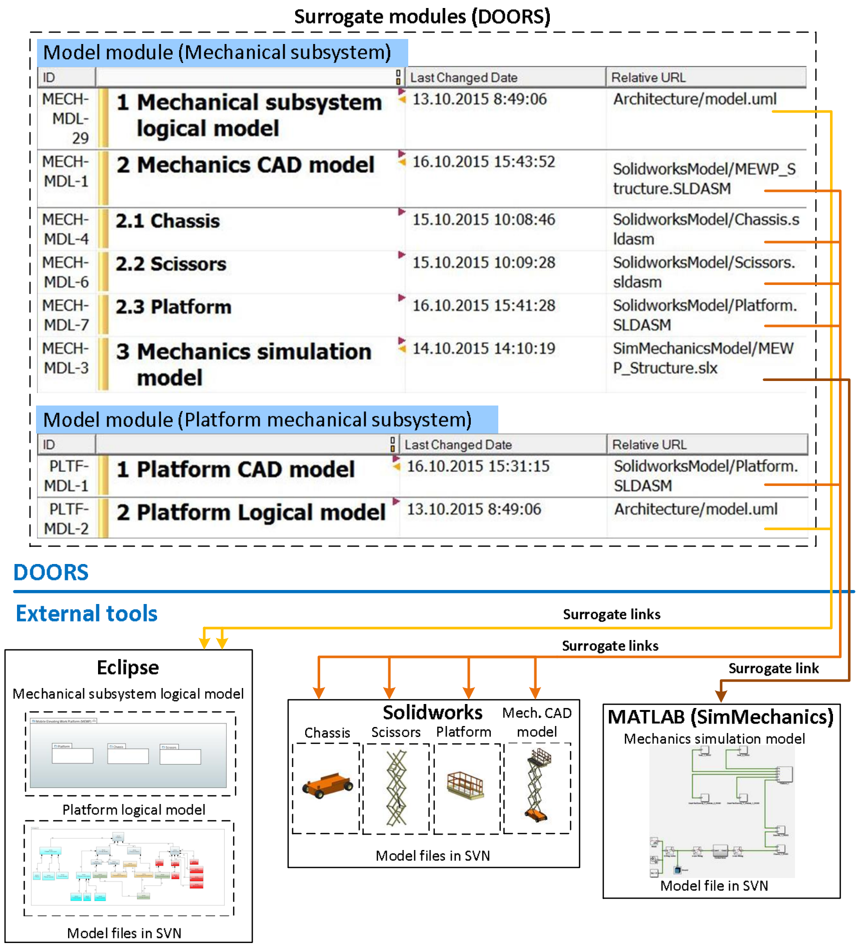


Figure 30. The surrogate object method.

The foreign tools, like CAD, SysML and simulation tools often work only in a local workspace, i.e. within a local hard drive. To make available the files produced by the tools for all the relevant parties, as well as for the DOORS tool, the files are committed to a version control system. In this case, Subversion (SVN) was selected. DOORS and SVN together makes thus the ITP (presented in Section 5.2) for the demonstration. The necessary files from the different software tools such as SolidWorks, Papyrus and Simulink are brought together in SVN to enable the tracing process within DOORS.

An example of the surrogate method usage is depicted in Figure 31.



Model module (Mechanical subsystem)

ID		Last Changed Date	Relative URL
MECH-MDL-29	**1 Mechanical subsystem logical model**	13.10.2015 8:49:06	Architecture/model.uml
MECH-MDL-1	**2 Mechanics CAD model**	16.10.2015 15:43:52	SolidworksModel/MEWP_Structure.SLDASM
MECH-MDL-4	**2.1 Chassis**	15.10.2015 10:08:46	SolidworksModel/Chassis.sldasm
MECH-MDL-6	**2.2 Scissors**	15.10.2015 10:09:28	SolidworksModel/Scissors.sldasm
MECH-MDL-7	**2.3 Platform**	16.10.2015 15:41:28	SolidworksModel/Platform.SLDASM
MECH-MDL-3	**3 Mechanics simulation model**	14.10.2015 14:10:19	SimMechanicsModel/MEWP_Structure.slx

Model module (Platform mechanical subsystem)

ID		Last Changed Date	Relative URL
PLTF-MDL-1	**1 Platform CAD model**	16.10.2015 15:31:15	SolidworksModel/Platform.SLDASM
PLTF-MDL-2	**2 Platform Logical model**	13.10.2015 8:49:06	Architecture/model.uml

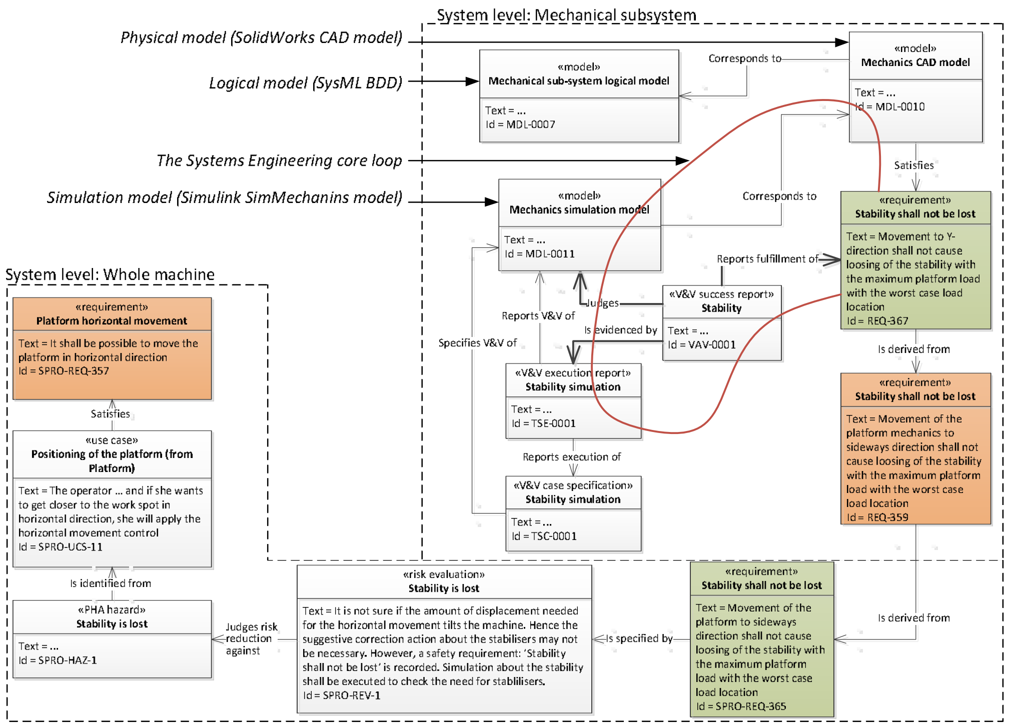
Figure 31. An example of the surrogate object method usage.

A set of artefacts was created during the work phases described in Sections 5.3.1 to 5.3.4. The artefacts are (see Figure 37 and Figure 39 in Appendix B):

- Stakeholder requirements
- System requirements
- PHA Hazards
- Risk evaluations
- Use cases
- Architecture model
- CAD model
- Verification requirements
- Simulation case specification
- Simulation model

- Simulation results
- Requirements trace
- Verification report.

Figure 32 depicts the artefacts that were traced in the demonstration. Note that all the artefacts are implemented as native DOORS objects, except the model artefacts, implementations of which are model files that are represented by surrogate objects in DOORS. There were many more artefacts, which are not presented in the simplified traceability diagram in Figure 32. The complete traceability diagram can be found elsewhere (Tikka 2015).


System level: Mechanical subsystem
System level: Whole machine
Physical model (SolidWorks CAD model)
Logical model (SysML BDD)
The Systems Engineering core loop
Simulation model (Simulink SimMechanins model)
Corresponds to
Corresponds to
Satisfies
Reports fulfillment of
Judges
Reports V&V of
Is evidenced by
Specifies V&V of
Reports execution of
Is derived from
Is derived from
Is specified by
Judges risk reduction against
Is identified from
Satisfies

«model»
Mechanics CAD model
Text = ...
Id = MDL-0010

«model»
Mechanical sub-system logical model
Text = ...
Id = MDL-0007

«model»
Mechanics simulation model
Text = ...
Id = MDL-0011

«requirement»
Stability shall not be lost
Text = Movement to Y-direction shall not cause loosing of the stability with the maximum platform load with the worst case load location
Id = REQ-367

«V&V success report»
Stability
Text = ...
Id = VAV-0001

«V&V execution report»
Stability simulation
Text = ...
Id = TSE-0001

«V&V case specification»
Stability simulation
Text = ...
Id = TSC-0001

«requirement»
Stability shall not be lost
Text = Movement of the platform mechanics to sideways direction shall not cause loosing of the stability with the maximum platform load with the worst case load location
Id = REQ-359

«requirement»
Platform horizontal movement
Text = It shall be possible to move the platform in horizontal direction
Id = SPRO-REQ-357

«use case»
Positioning of the platform (from Platform)
Text = The operator ... and if she wants to get closer to the work spot in horizontal direction, she will apply the horizontal movement control
Id = SPRO-UCS-11

«PHA hazard»
Stability is lost
Text = ...
Id = SPRO-HAZ-1

«risk evaluation»
Stability is lost
Text = It is not sure if the amount of displacement needed for the horizontal movement tilts the machine. Hence the suggestive correction action about the stabilisers may not be necessary. However, a safety requirement: 'Stability shall not be lost' is recorded. Simulation about the stability shall be executed to check the need for stablilisers.
Id = SPRO-REV-1

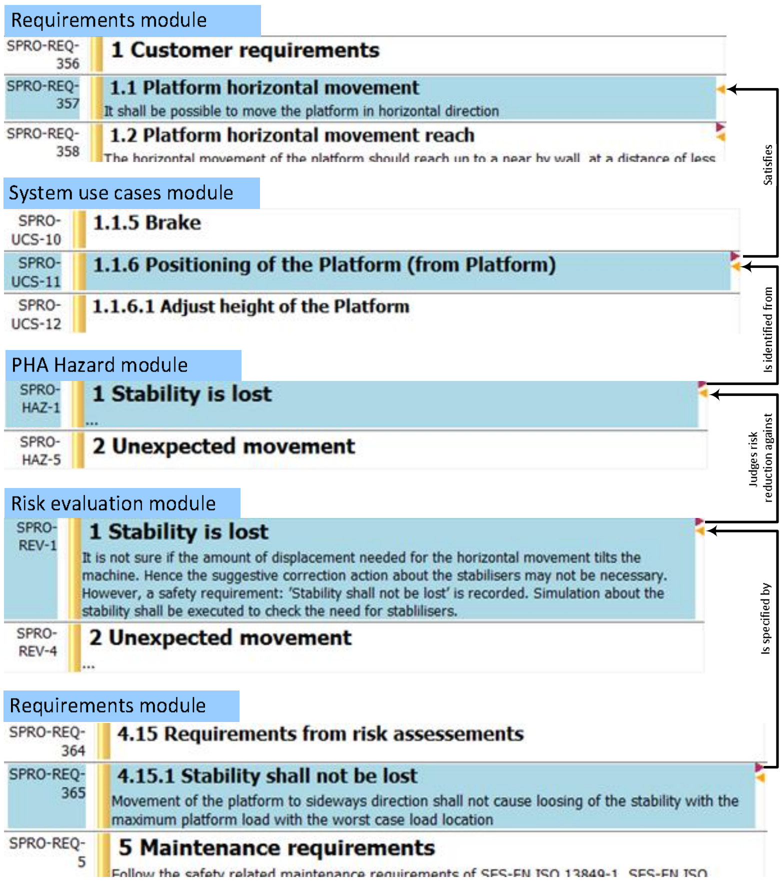
«requirement»
Stability shall not be lost
Text = Movement of the platform to sideways direction shall not cause loosing of the stability with the maximum platform load with the worst case load location
Id = SPRO-REQ-365

Figure 32. Simplified traceability diagram.

The red freehand arc in Figure 32 points out the Systems Engineering core loop, i.e. from a requirement to design artefacts, from design artefacts to the V&V execution report (in this case simulation results), from the V&V results to the V&V success reports and from the V&V success reports back to the original requirement that is claimed to be fulfilled (or not fulfilled) by the design artefacts.

All the traces in Figure 32 are implemented as DOORS trace links. Hence the impact analysis feature provided by DOORS works normally and covers the whole SE core loop.

The DOORS implementation of the core traces of Figure 32 are depicted in Figure 33 and Figure 34.



Requirements module

SPRO-REQ-356	**1 Customer requirements**
SPRO-REQ-357	**1.1 Platform horizontal movement** It shall be possible to move the platform in horizontal direction
SPRO-REQ-358	**1.2 Platform horizontal movement reach** The horizontal movement of the platform should reach up to a near by wall, at a distance of less

System use cases module

SPRO-UCS-10	**1.1.5 Brake**
SPRO-UCS-11	**1.1.6 Positioning of the Platform (from Platform)**
SPRO-UCS-12	**1.1.6.1 Adjust height of the Platform**

PHA Hazard module

SPRO-HAZ-1	**1 Stability is lost** ...
SPRO-HAZ-5	**2 Unexpected movement**

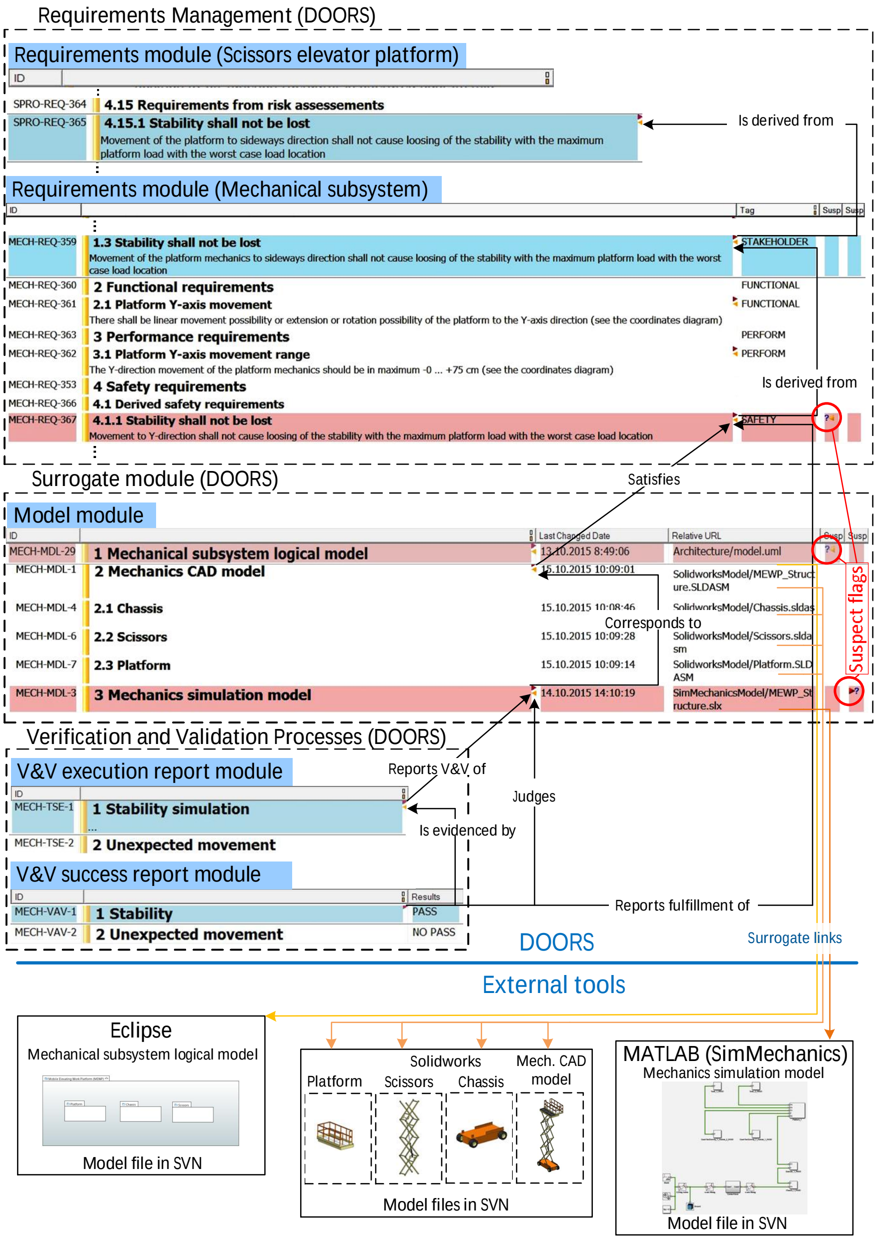
Risk evaluation module

SPRO-REV-1	**1 Stability is lost** It is not sure if the amount of displacement needed for the horizontal movement tilts the machine. Hence the suggestive correction action about the stabilisers may not be necessary. However, a safety requirement: 'Stability shall not be lost' is recorded. Simulation about the stability shall be executed to check the need for stablilisers.
SPRO-REV-4	**2 Unexpected movement** ...

Requirements module

SPRO-REQ-364	**4.15 Requirements from risk assessements**
SPRO-REQ-365	**4.15.1 Stability shall not be lost** Movement of the platform to sideways direction shall not cause loosing of the stability with the maximum platform load with the worst case load location
SPRO-REQ-5	**5 Maintenance requirements** Follow the safety related maintenance requirements of SFS-EN ISO 13849-1, SFS-EN ISO

Figure 33. DOORS implementation of the trace chain from the 'Stability is lost' requirements to the original requirement at the whole machine level (see Figure 32).



Requirements module (Scissors elevator platform)

ID	
SPRO-REQ-364	**4.15 Requirements from risk assessements**
SPRO-REQ-365	**4.15.1 Stability shall not be lost** Movement of the platform to sideways direction shall not cause loosing of the stability with the maximum platform load with the worst case load location

Requirements module (Mechanical subsystem)

ID		Tag	Susp	Susp
MECH-REQ-359	**1.3 Stability shall not be lost** Movement of the platform mechanics to sideways direction shall not cause loosing of the stability with the maximum platform load with the worst case load location	STAKEHOLDER		
MECH-REQ-360	**2 Functional requirements**	FUNCTIONAL		
MECH-REQ-361	**2.1 Platform Y-axis movement** There shall be linear movement possibility or extension or rotation possibility of the platform to the Y-axis direction (see the coordinates diagram)	FUNCTIONAL		
MECH-REQ-363	**3 Performance requirements**	PERFORM		
MECH-REQ-362	**3.1 Platform Y-axis movement range** The Y-direction movement of the platform mechanics should be in maximum -0 ... +75 cm (see the coordinates diagram)	PERFORM		
MECH-REQ-353	**4 Safety requirements**			
MECH-REQ-366	**4.1 Derived safety requirements**			
MECH-REQ-367	**4.1.1 Stability shall not be lost** Movement to Y-direction shall not cause loosing of the stability with the maximum platform load with the worst case load location	SAFETY	?	

Model module

ID		Last Changed Date	Relative URL	Susp	Susp
MECH-MDL-29	**1 Mechanical subsystem logical model**	13.10.2015 8:49:06	Architecture/model.uml	?	
MECH-MDL-1	**2 Mechanics CAD model**	15.10.2015 10:09:01	SolidworksModel/MEWP_Structure.SLDASM		
MECH-MDL-4	**2.1 Chassis**	15.10.2015 10:08:46	SolidworksModel/Chassis.sldas		
MECH-MDL-6	**2.2 Scissors**	15.10.2015 10:09:28	SolidworksModel/Scissors.sldasm		
MECH-MDL-7	**2.3 Platform**	15.10.2015 10:09:14	SolidworksModel/Platform.SLDASM		
MECH-MDL-3	**3 Mechanics simulation model**	14.10.2015 14:10:19	SimMechanicsModel/MEWP_Structure.slx		?

V&V execution report module

ID	
MECH-TSE-1	**1 Stability simulation** ...
MECH-TSE-2	**2 Unexpected movement**

V&V success report module

ID		Results
MECH-VAV-1	**1 Stability**	PASS
MECH-VAV-2	**2 Unexpected movement**	NO PASS

Figure 34. DOORS implementation of the Systems Engineering core loop at the mechanical subsystem level in Figure 32.

Figure 34 also depicts an impact analysis case: The Model module and the Requirements module of the mechanical subsystem display the suspect flags for the case in which the mechanics CAD model has been changed. The suspect flags indicate that it has to be checked whether the CAD model still satisfies the *Stability shall not be lost* -requirement (MECH-REQ-367) (and whether the CAD model still corresponds to the Mechanical subsystem logical model; this link is beyond the SE core loop); furthermore, it has to be checked that the mechanics simulation model still corresponds to the CAD model.

5.4 Results of the demonstration

The demonstration indicates that SEAModel can be implemented satisfactorily with a set of common mechanical engineering tools and a typical requirements management tool.

In this case, the integration and traceability platform (ITP) consisted of the Subversion version control software and DOORS requirements management tool with some additional scripting. Such an integration platform does not provide optimal granularity of traceability due to the fact that the models (SysML, CAD and simulation) can only be traced at the file level.

Furthermore, true integration of the tools data is not achieved. This means, for example, that when the systems engineer sketches the physical architecture of a mechanical subsystem with a SysML tool, the mechanical CAD engineer has to manually input the data from the SysML model to create the CAD model. The consequence is that there may be misprints and parameter errors in the created CAD model, and the changes in the SysML model do not automatically reflect to the CAD model. It may be, anyway, difficult to automatically reflect structural changes from a SysML model to its corresponding CAD model, but at least the model element names and parameter values should reflect from the father model to the child model. Nevertheless, with the implementation concept used in our demonstrations, changes in the father model mark the child model with a suspect flag prompting the engineer of the child model to check, what needs to be changed in the child model due to the changes in the father model.

The DXL script created in this work for DOORS to connect DOORS objects to files in Subversion to facilitate the file level traceability was found to be a very useful and an easy step forward from the current practices that do not apply any kind of tracing of modelling artefacts. It is very probable that similar scripts can be created to provide traceability connectivity between other than DOORS and Subversion software tools.

During the demonstration, there was a noticeable difficulty in deciding about which artefacts should be traced to a particular artefact, even though the TIM and SEAModel were available. The main reason for this was in the fact that the demonstration included artefacts from three system levels, the whole system level, the mechanical subsystem level and the platform level. So it was rather easy to get confused with the system levels. This observation about the complexity suggests that an optimal traceability tool shall be aware of the artefacts data model (like SEAModel) of the company and guide the user in creating the trace links.

6. Conclusions

With the Systems Engineering Artefacts Model (SEAModel) reported in this publication, requirements engineering can be extended to cover simulation artefacts. SEAModel supports well at least the basic needs for simulation artefacts tracing. SEAModel is implementable with different kinds of database oriented software platforms, and thus it fits even for small- and medium-sized companies. However, the implementation work varies greatly depending on the chosen platform. Optimal solution is quite difficult to find. Traceability of artefacts from a set of heterogeneous modelling tools is difficult to arrange with the optimal granularity and full visibility of data. However, file level granularity of model elements may provide the satisfactory solution for traceability, and is a step forward from neglected requirements traceability.

The greatest obstacle in requirements aware simulation engineering is still in the attitudes of the engineers. SEAModel tries to facilitate creation of data repository and traceability tools such that simulation engineers feel more comfortable with a well organised data repository for the artefacts they need as the input for their work and for the artefacts they and their tools produce. As awareness of the benefits and thus application of model-based systems engineering increases, such a structured data repository is a more natural approach over a conventional document based repository of systems engineering data.

The research work done in Task 3 of the VTT subproject pointed out the need to promote the Model-Based Systems Engineering approach in Finland to better manage requirements, design artefacts and V&V artefacts of complex systems. There is a lot of work to do to make the heterogeneous systems engineering tools to work on shared data instead of isolated snapshots of the same information on each tool. Simantics provides one interesting platform for the purpose. With the traceability information model according to SEAModel, Simantics would offer SME:s with an affordable solution for integrated systems development.

Another future research topic is the collaboration model to improve the communications and data sharing between contractors and their sub-contractors. This is especially important in the development of safety relevant systems to ensure consistent and actual set of safety requirements from contractors to their sub-contractors, and transfer of evidence of fulfilling the requirements from sub-contractors to the main contractors. SEAModel can be used as the basis for the data sharing model.

Working with the SEAModel is quite complex without a Systems Engineering background. Nevertheless, development of complex system causes complex processes and complex artefacts models, as Niklas Luhmann said: “Only complexity can reduce complexity.” The complexity can be alleviated by a good traceability tool that is aware of the artefacts model (such as SEAModel) used in the company.

References

Alanen, J. and Tiusanen, R. 2010. Operating Hazard Analysis Using Supplemented Use Case Descriptions. The 6th International Conference on safety of Industrial Automation Systems (SIAS 2010). Tampere, 14–15.6.2010. SIAS 2010 Proceedings. Helsinki: Suomen Automaation Tuki Oy (Finnish Automation Support Ltd). 6 p.

Alanen, J., Vidberg, I., Nikula, H., Papakonstantinou, N., Pirttioja, T. and Sierla, S. 2011. Engineering data model for machine automation systems. VTT Tiedotteita 2583. Espoo: VTT. 137 p. Available at: http://www.vtt.fi/inf/pdf/tiedotteet/2011/T2583.pdf

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.

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 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 62061. 2005. Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems. 2005-01-20 edn. Geneva: International Electrotechnical Commission. 205 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 12100. 2010. Safety of machinery – General principles for design – Risk assessment and risk reduction. Edition 1. Geneva: International Organisation for Standardization. 77 p.

ISO 13849-1. 2006. Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design. Edition 2. Geneva: International Organisation for Standardization. 85 p.

ISO 13849-2. 2012. Safety of machinery – Safety-related parts of control systems – Part 2: Validation. Edition 2. Geneva: International Organisation for Standardization. 79 p.

ISO/IEC/IEEE 15288. 2008. Systems and Software Engineering – System Life Cycle Processes. 2008-02-01 edn. International Organization for Standardization, International Electrotechnical & Institute of Electrical and Electronics Engineers. 84 p.

Mätäsniemi, T. and Alanen, J. 2015. Implementation of ReqIF for the Simantics platform. VTT Research report VTT-R-04722-15. 22 p.

MSR Consortium. 2002. MSRSYS – ECU control system description [Homepage of 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 18.11.2014].

Tikka, P. 2015. Requirements Traceability in Simulation Driven Mechanical Engineering. Master Thesis. Tampere: Tampere University of Technology (to be published).

Appendix A: Descriptions of the artefact types

In the following, descriptions of the artefact types presented in the artefacts models in Chapter 3 are provided. Note that the descriptions are not concept definitions, but they supply information about the purpose and usage of the artefact types.

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.
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.
Artefact repository: A database or a XML storage for all systems engineering artefacts.
Audio file: Audio file is a special case of File. An audio file can be of any format (WAV, MP3, and so forth). The most important attribute is the link to the actual audio file.
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.
Black box artefact: Contrary to Model element, a black box artefact is an engineering artefact with no or limited knowledge about their internal structure, or which do not have a well-defined structure. Such artefacts are conventional documents and foreign models.
CANopen object. CANopen object is an information item (an instance of a CANopen object type) carried by a communications message.
CANopen object type. CANopen object type according to CiA DS301 and DSP 311.

Channel. A logical entity that encompasses the signal processing flow from input to output through logic.

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.

Composite flow. A special case of a flow in which 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.

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 (like 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.

Context description. The Context description artefact provides description of the system-of-interest’s general context including 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.

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 like Stakeholder, Human actor, Life cycle phase, Past incident, Constraint, Environment, External system (element) and Glossary item.

Continuous action: A special case of Action. Action that continues over an indefinite time, such as monitoring of a temperature.

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.

Device. A special case of Component, such as a sensor, actuator or controller.
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.
Document. The Document artefact defines any type of document or package of documents (like the actual document and its attachments). The most important attribute is the link to the actual document file.
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.
Electrical port type: The electrical case of the Port type artefact.
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.
Environment. Description of the mechanical, climatic, chemical, ergonomic, external system and other 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.)
Event: Event describes any type of event relevant to the system behavioural modelling, like 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).
External port: A special case of Port; a port of an External system element that interacts with the system-of-interest.
External system: A special case of System; a system that interacts with the system-of-interest.

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.
Fault mode: An identified fault or failure mode of a particular system element.
Fault mode type: A general fault or failure mode type.
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.
Flow: Specifies the flow between ports. A flow can be electrical (signal), hydraulic (fluid), mechanical (momentum), optical (light), and so forth. Flows specify the ports (i.e. they inform, what type of flows the ports are able to transmit or receive).
FMECA hazard: A special case of Hazard. A hazard identified by the Failure Modes, Effects and Criticality Analysis method.
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.
Foreign domain knowledge model: A special case of Foreign model for modelling the domain knowledge of the system context.
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, and the like that are not stored as a structured set of artefacts and their relations, but as a 'black box' model 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 and the like, or a bystander who is situated in the hazard zone of the system.
Hydraulic port type: The hydraulic case of the Port type artefact.
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, and so forth) 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, like a message on a fieldbus.
Instantaneous action: A special case of Action. A 'zero' length action, like 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 and the like.
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.
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, like 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.

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 (like sensors) to output (like hydraulic actuators) through logic (like programmable logic controller).
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 state: Placeholder for the descriptions of the states of the system, like power down state, power up state, operating state and emergency stop state.
Optical port type: The optical case of the Port type artefact.
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.
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 format. The most important attribute is the link to the actual photograph file.
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.
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, like weight, dimensions and power consumption, but also more abstract parameters, like 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 used 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 have potential for application, alt-

hough not currently used in any of the organisation's system.
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.
Property: The Property artefacts represent 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.
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, like suspect flag (for impact analysis), relationship type and rationale for the relation.
Requirement: The Requirement artefact defines a requirement and its attributes, like type and source. A requirement can be a stakeholder requirement, system requirement or constraint.
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.
Rich document file: A special case of File. A document file that supports formatting of text and inclusion of tables and drawings and other means of information presentation.
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).
Risk estimation: Stores the risk estimation parameters and the resulting risk

level.
Risk estimation IEC 61508: Risk estimation parameters and risk level according to IEC 61508-5:2010.
Risk estimation IEC 62061: Risk estimation parameters and risk level according to IEC 62061:2006.
Risk estimation ISO 13849: Risk estimation parameters and risk level according to ISO 13849-1:2006.
Risk evaluation: Stores the judgment and conclusions, on the basis of the risk analysis, of whether the risk reduction objectives have been achieved.
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 (not presented in this report).
Safety block: A set of system elements that constitute a specific portion of the overall safety related part of the control system.
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.
Safety-related block diagram: A special case of diagram that illustrates the logical structure of the safety related part of the control system according to Annex B of ISO 13849-1:2008.
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.
Safety-related part of control system: “Part of a control system that responds to safety-related input signals and generates safety-related output signals” (ISO

13849-1:2008). It can be electrical, electronic, programmable electronic, mechanical, pneumatic, hydraulic, and so forth.
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 action flow as a story or in a more structured way by well-defined attributes.
Session: The session meeting minutes are recorded in the Session artefact. If wished, this artefact type can be divided into two types, Minutes and Session.
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.
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.
Stakeholder requirement: A special case of Requirement: requirement set by a stakeholder.
Structural element: A special case of Component; a hardware element, like rod, wall and the like.
Structure: Structure provides a means to present the system's physical architecture. The structure artefact can represent a partial view of the whole system structure or different viewpoints of the whole system structure, like development time view and manufacturing time view.
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.

SW element: A special case of Component; a reusable, encapsulated, piece of software, a software component.
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.
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.
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.
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, like information items and SW elements.
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.
System function: A system level function (contrary to SW function), e.g. boom movement.
System model: A special case of Model for modelling the requirements, behaviour, architecture and other properties of the system-of-interest.
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 representation for system-of-interest, but the system-of-interest is implemented as a System artefact.

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 pre-conditions of another use case or system task).
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.
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 in which 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.
System use case: A system level use case (contrary to SW use cases). Use cases can be specified according to SysML.
System use case supplement: A safety related supplement that adds attributes such as 'actor qualification', 'list of operator instructions', 'expected misuse', 'activation frequency', 'preliminary accident scenario' to the system use case.
Temporal action: A special case of Action. An action carried out in finite time; i.e. is not instantaneous ('zero length') neither continuous ('infinite').
Test channel: A special case of Channel that performs monitoring of the normal channel.
UCSA hazard: A special case of Hazard. A hazard identified by the Use Case Safety Analysis method.
V&V case specification: Specifies a test, analysis, inspection, calculation, design document review, demonstration, calculation, simulation, comparison, sampling, measurement and the like 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 and other parameters. It

also includes a description of the expected results. Issues such as schedule and persons or roles whom this V&V case is as-signed 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.
V&V execution interpreter: An instruction document or a program that provides means for interpreting the verification results, like a program that plots a graph of the virtual measurement data.
V&V execution parameter: The parameters relating to the execution of the verification (or validation), like the number of simulation runs.
V&V execution report: Records the results of the test, analysis, demonstration, review and other case executions.
V&V model parameter: The parameters that configure the model item under verification or validation for the verification or validation execution.
V&V plan: Collects a set of test, analysis, demonstration, review and other cases to form a specific sequence of tests for a specific purpose, such as for Factory Acceptance Test (FAT).
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.
V&V success report: The result of verification or validation. The main contents are the pass/no-pass verdict and the argumentation of the verdict. This artefact type can also be called V&V claim.
Video file: Video file is a special case of File. A video file can be of any format. 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.

Appendix B: Semiformal presentation of the demonstration workflow

In the following figures, the demonstration workflow introduced in Section 5.2 is presented in a semiformal way using the UML and SysML use case and activity diagram notations. See figures from Figure 35 to Figure 48.

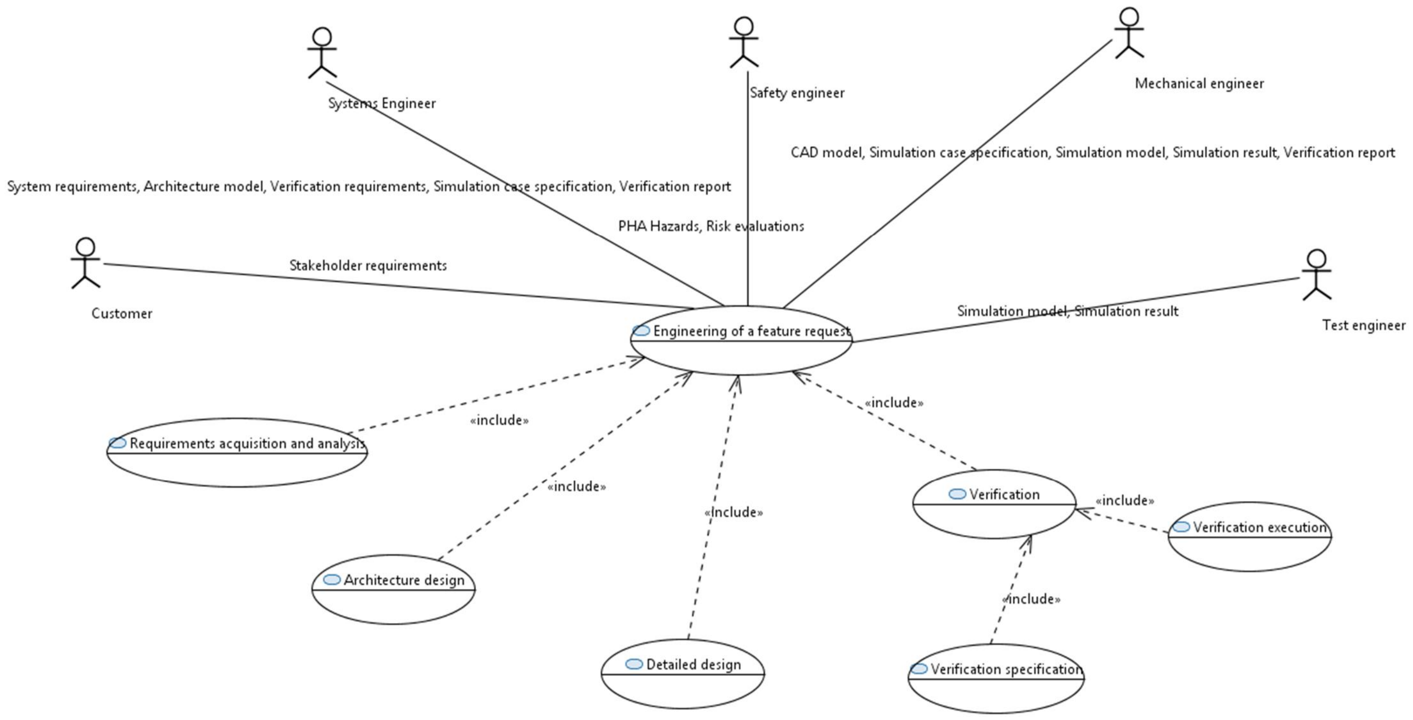

Systems Engineer
Safety engineer
Mechanical engineer
CAD model, Simulation case specification, Simulation model, Simulation result, Verification report
System requirements, Architecture model, Verification requirements, Simulation case specification, Verification report
PHA Hazards, Risk evaluations
Stakeholder requirements
Customer
Simulation model, Simulation result
Test engineer
Engineering of a feature request
«include»
Requirements acquisition and analysis
«include»
«include»
«include»
Verification
«include»
Verification execution
Architecture design
«include»
Detailed design
Verification specification

Figure 35. The main use case.

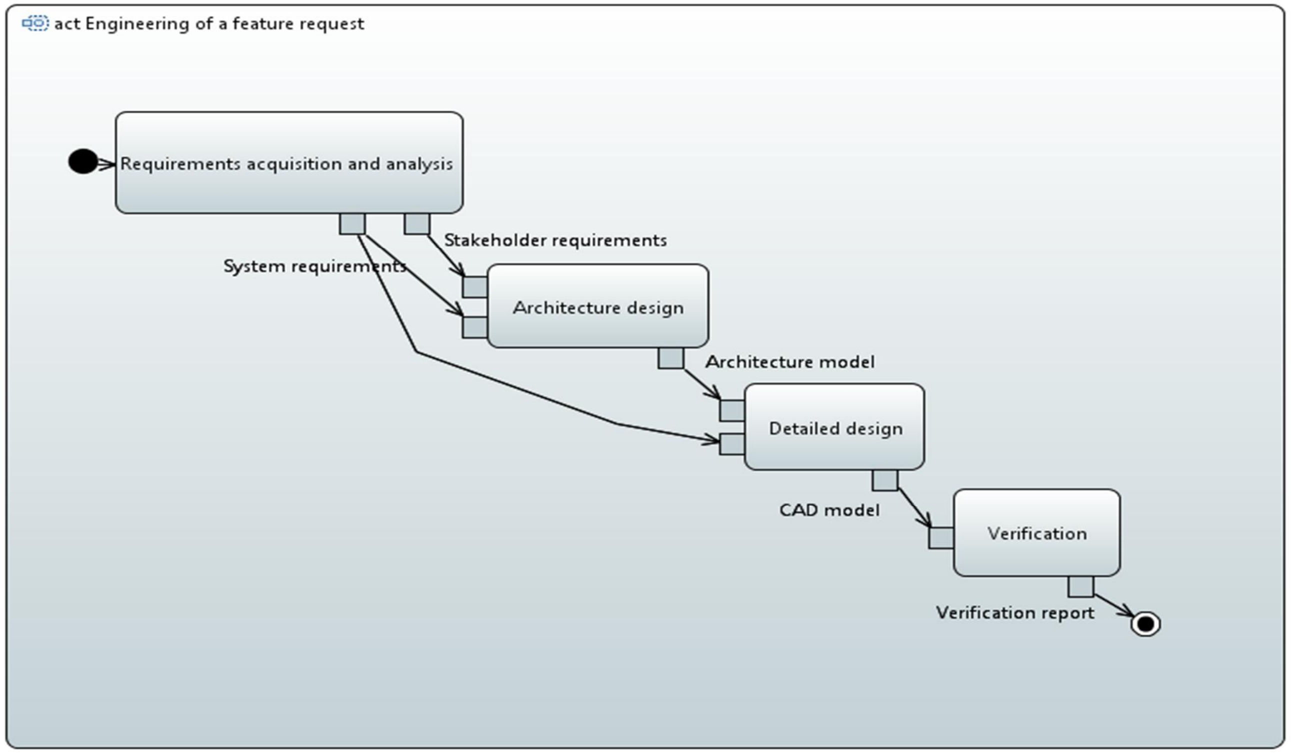


Figure 36. The overall workflow.

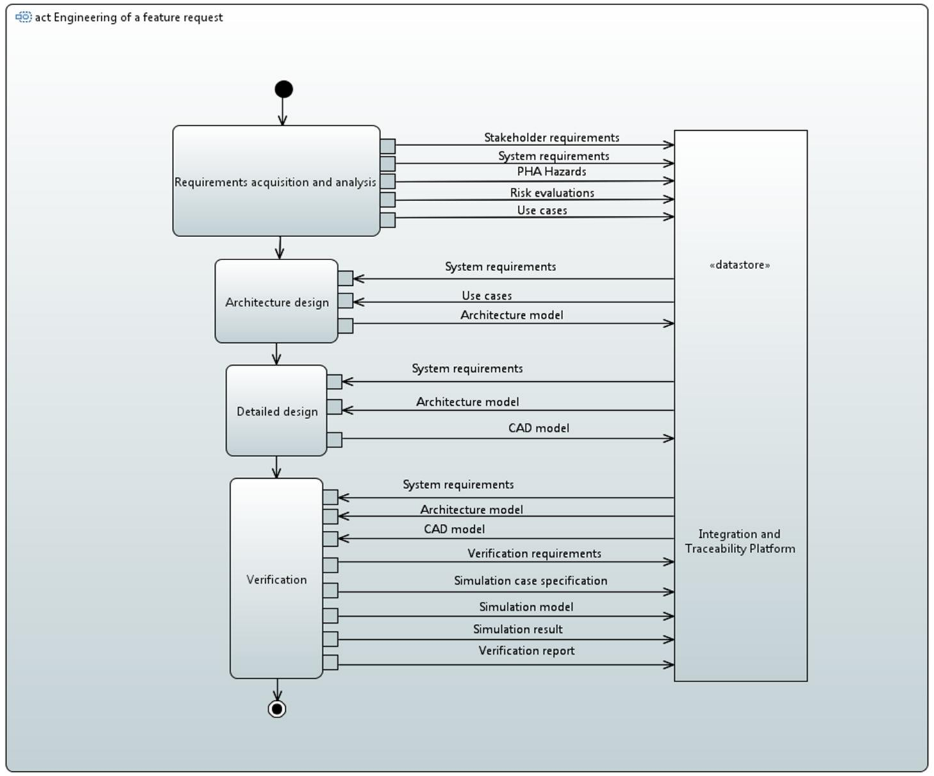


Figure 37. The overall workflow with detailed analysis of storing and using of engineering artefacts in the Integration and Traceability Platform (<<datastore>> in the SysML notation).

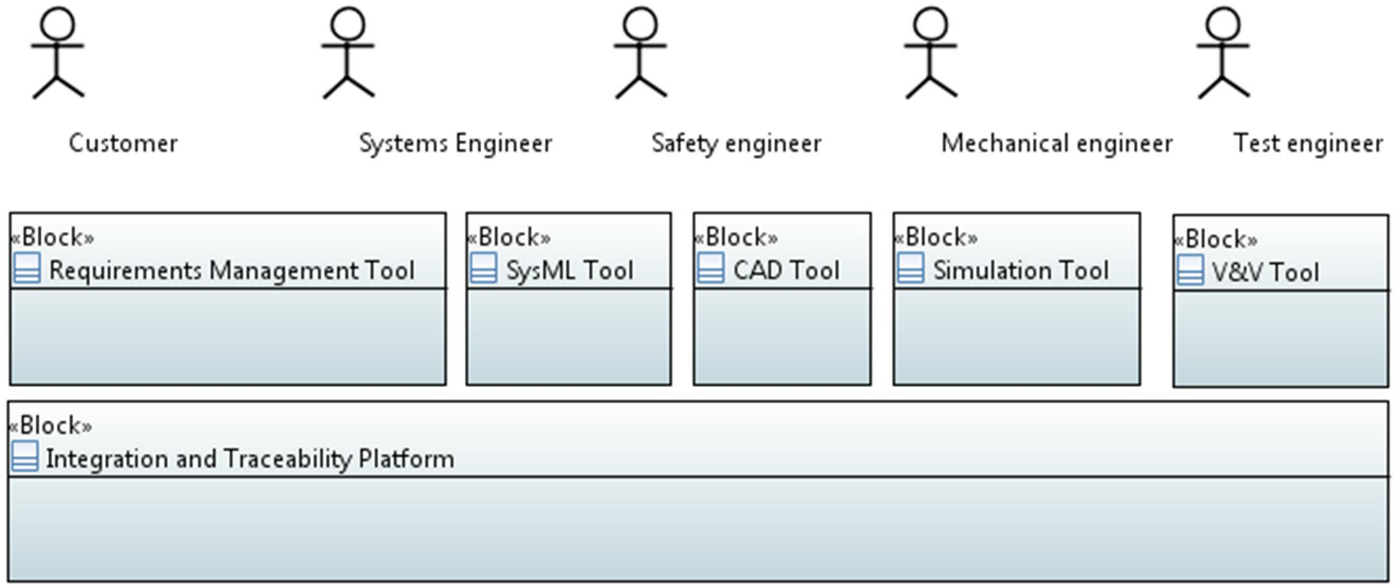


Figure 38. Context of the systems engineering workflow demonstration.

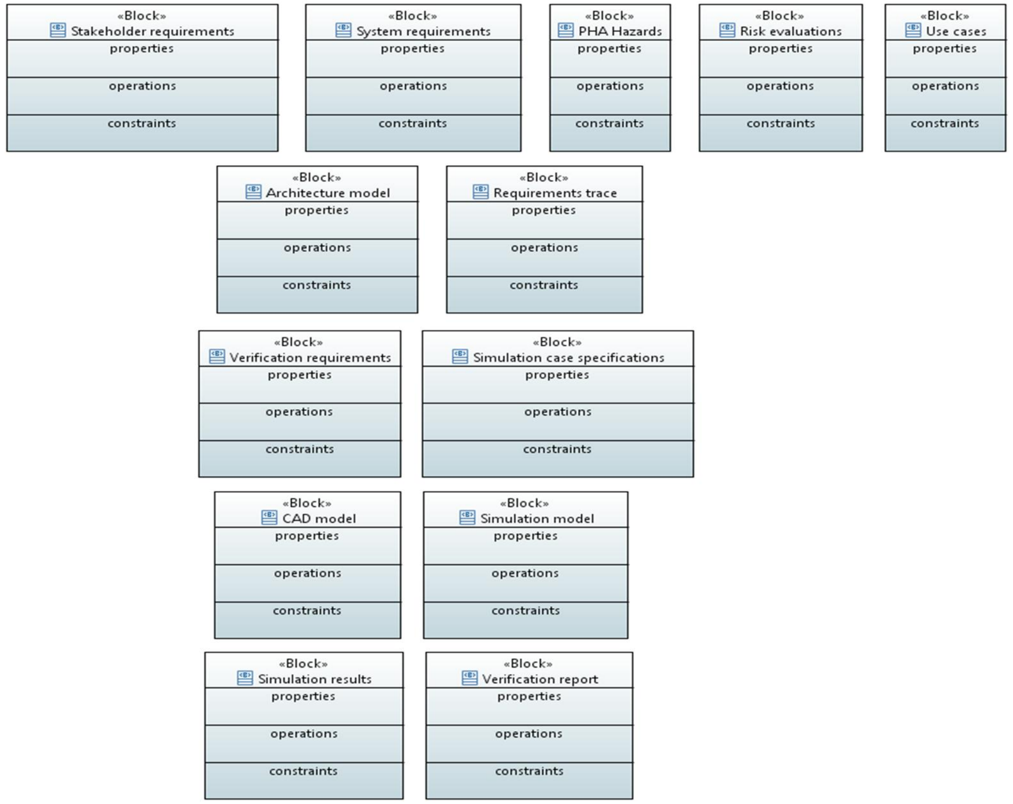


Figure 39. Engineering artefacts created during the workflow.

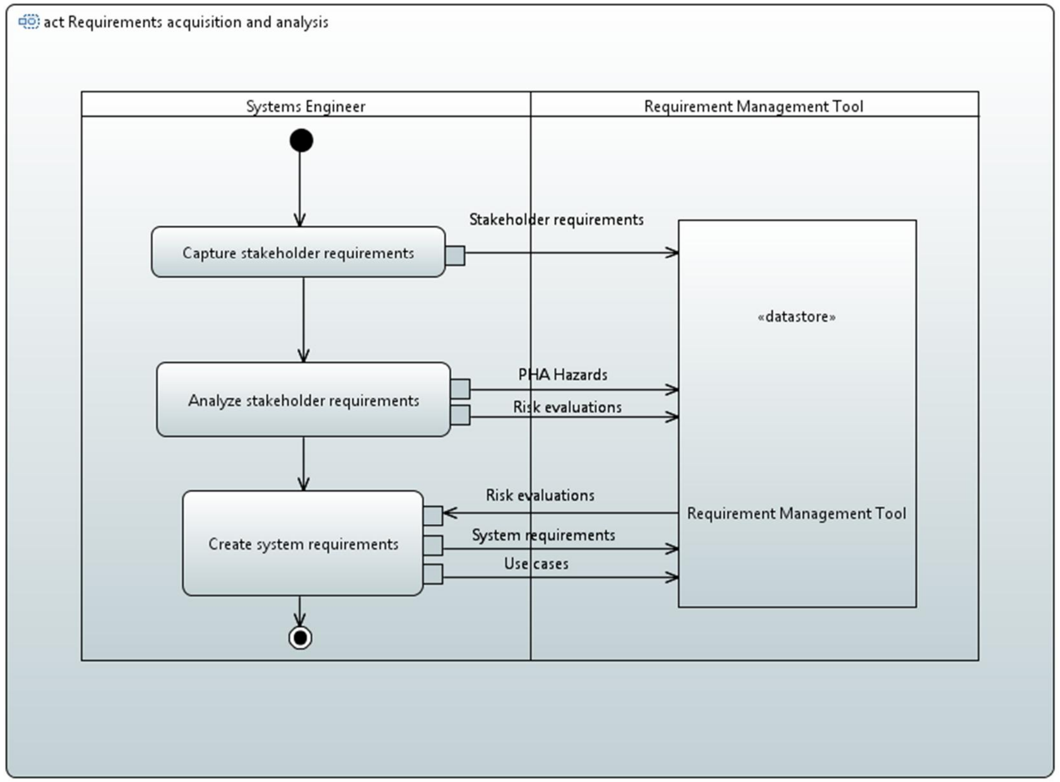

act Requirements acquisition and analysis
Systems Engineer
Requirement Management Tool
Capture stakeholder requirements
Stakeholder requirements
«datastore»
Analyze stakeholder requirements
PHA Hazards
Risk evaluations
Risk evaluations
Requirement Management Tool
Create system requirements
System requirements
Use cases

Figure 40. Requirements acquisition workflow.

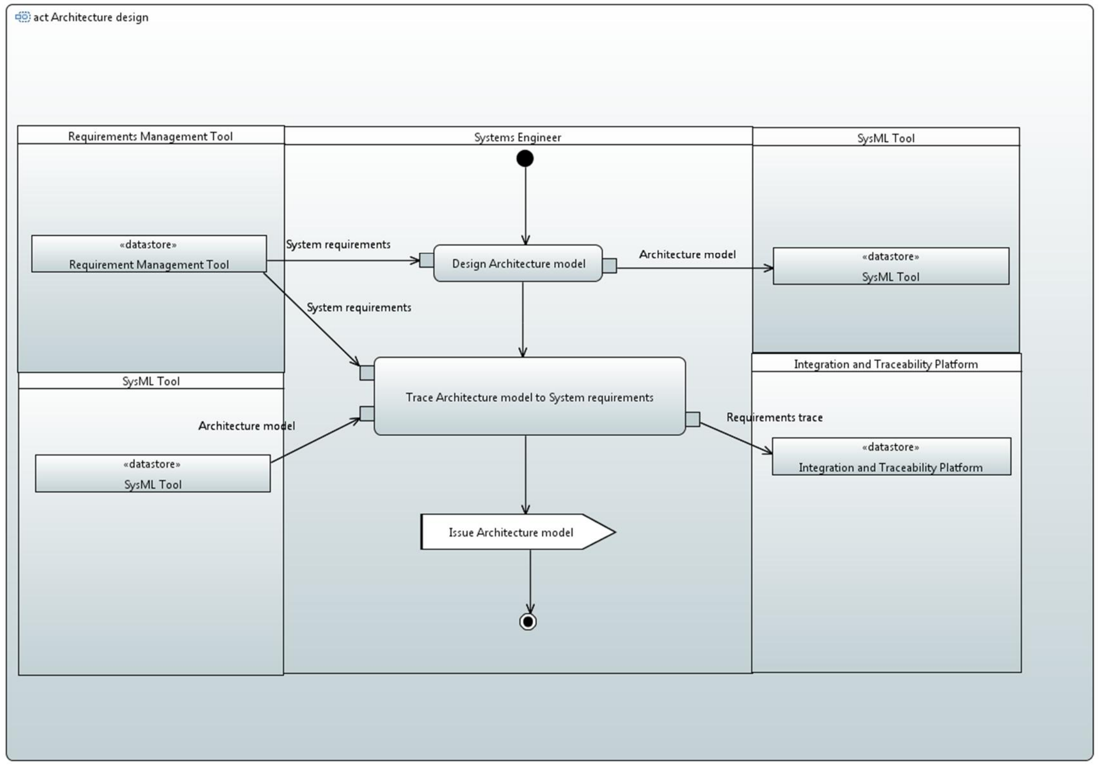


Figure 41. Architecture design workflow.

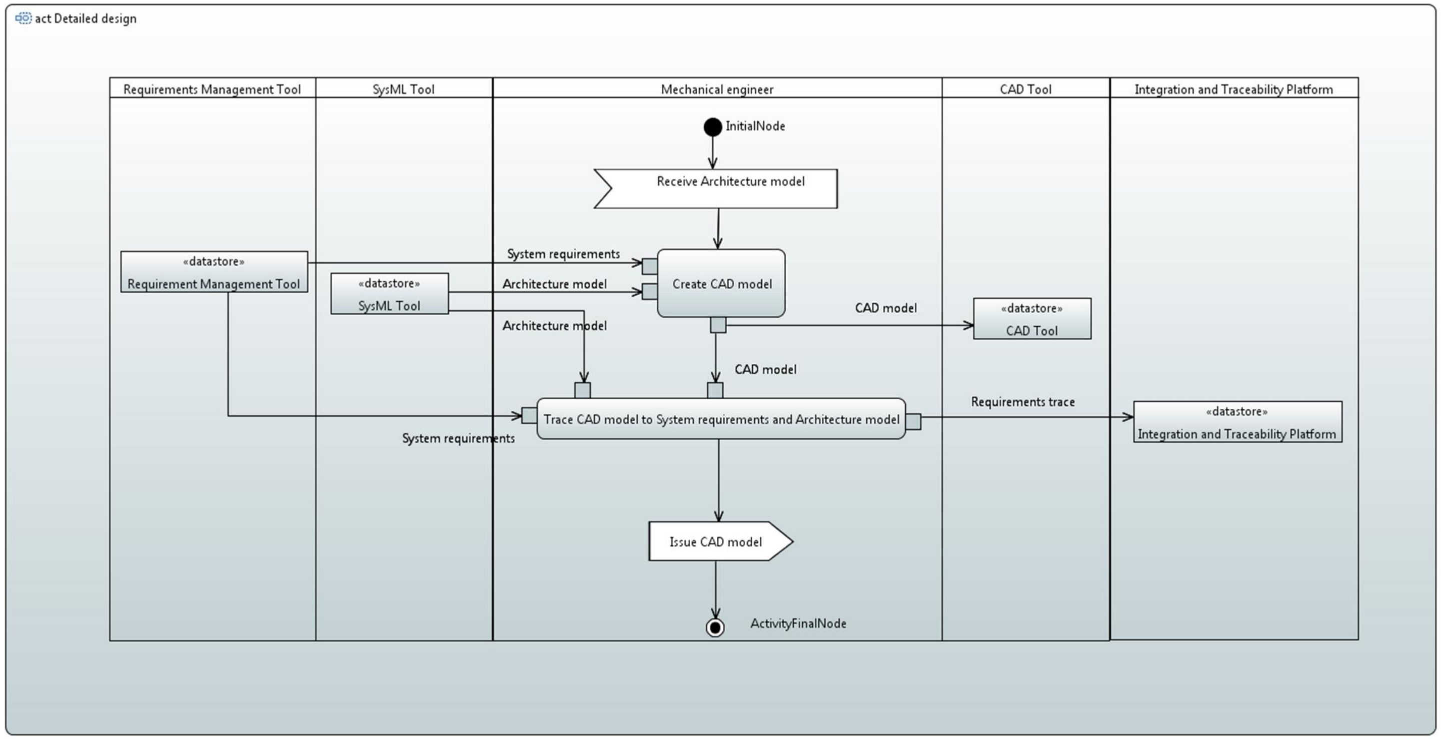


Figure 42. Detailed design (i.e. CAD model implementation) workflow.

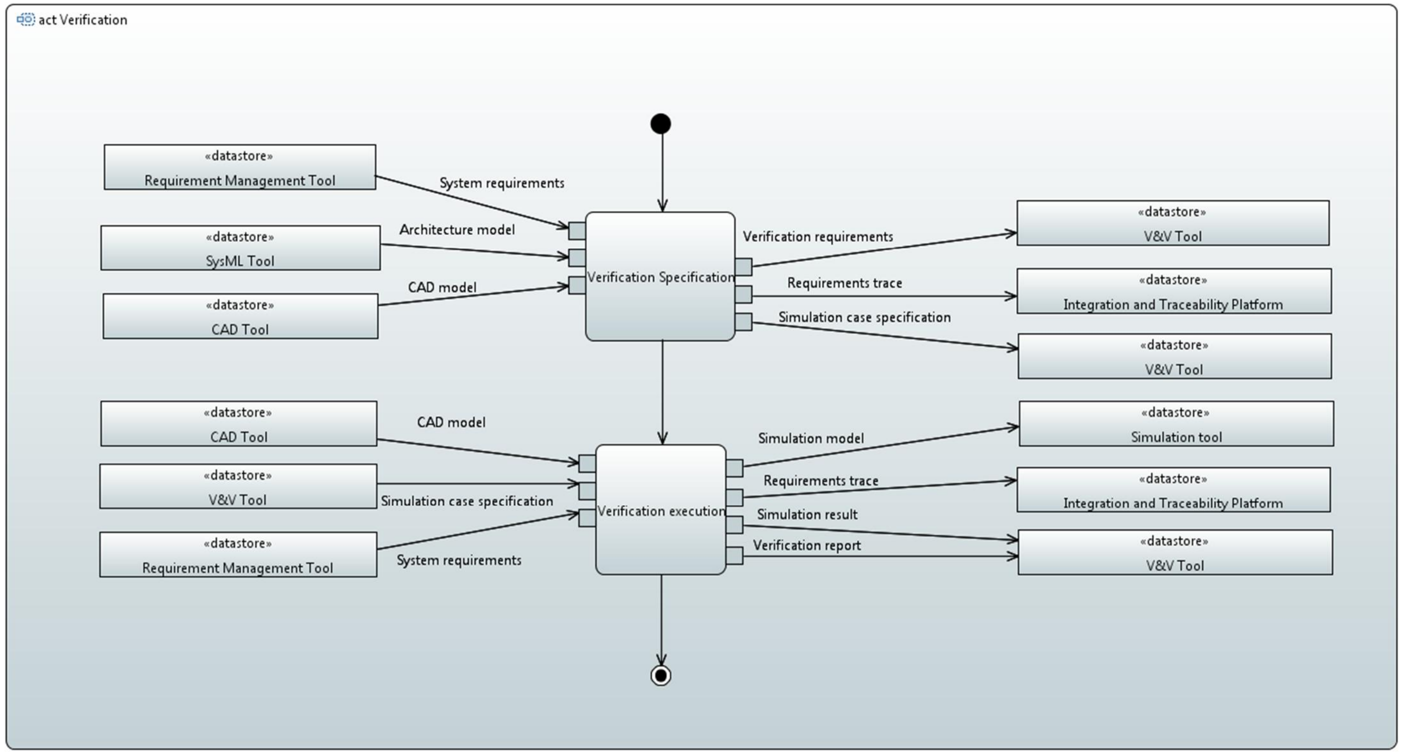


Figure 43. Verification workflow (overall) (see details of Verification specification in Figure 44 and Verification execution in Figure 46).

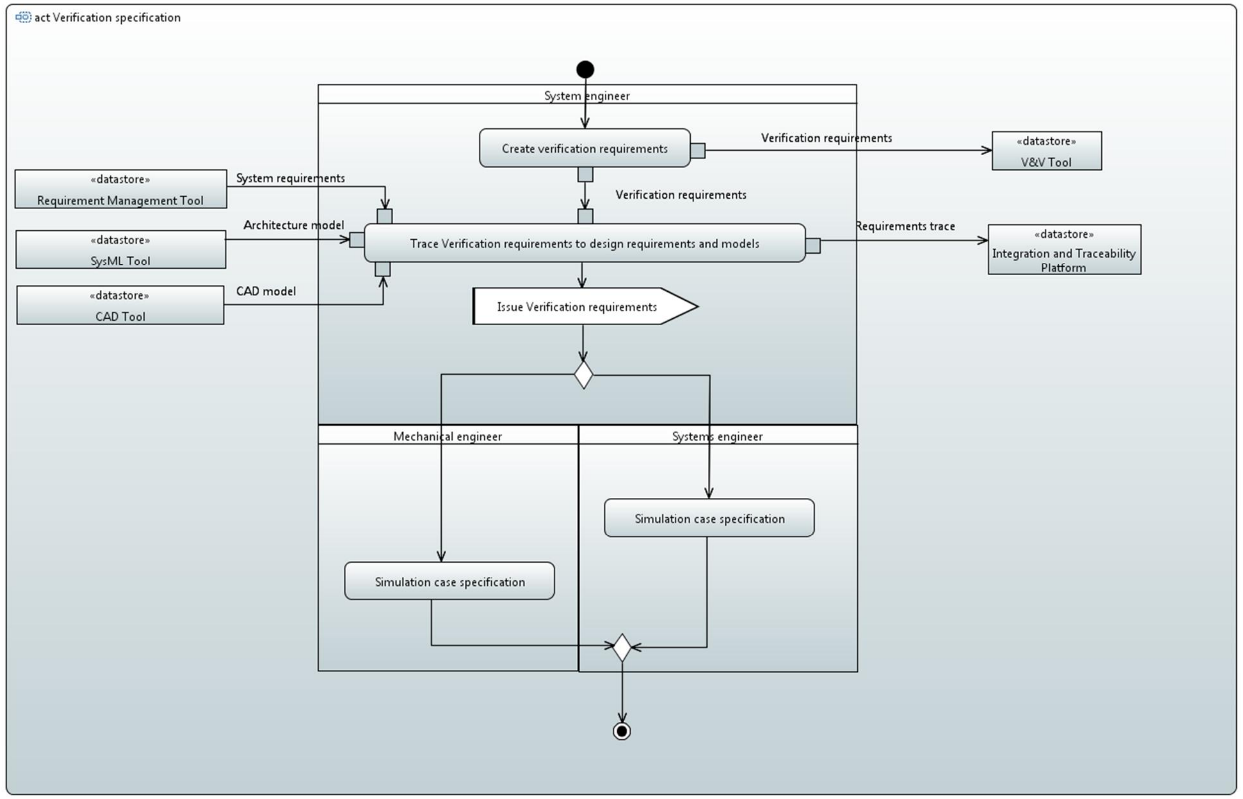

act Verification specification
System engineer
Create verification requirements
Verification requirements
«datastore»
V&V Tool
«datastore»
Requirement Management Tool
System requirements
Verification requirements
Architecture model
«datastore»
SysML Tool
Trace Verification requirements to design requirements and models
Requirements trace
«datastore»
Integration and Traceability Platform
«datastore»
CAD Tool
CAD model
Issue Verification requirements
Mechanical engineer
Systems engineer
Simulation case specification
Simulation case specification

Figure 44. Verification workflow (Verification specification) (see Simulation case specification details in Figure 45).

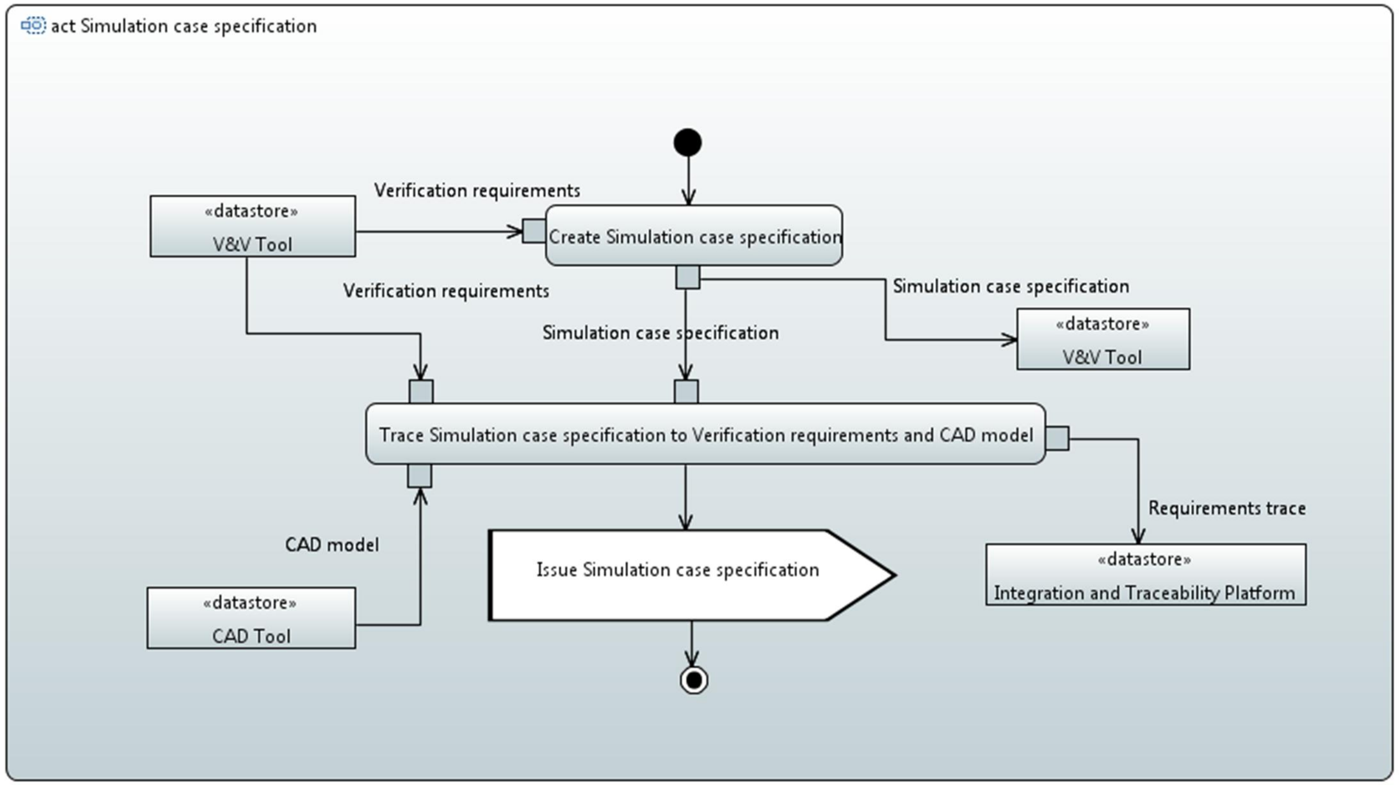

act Simulation case specification
«datastore»
V&V Tool
Verification requirements
Create Simulation case specification
Verification requirements
Simulation case specification
Simulation case specification
«datastore»
V&V Tool
Trace Simulation case specification to Verification requirements and CAD model
Requirements trace
CAD model
Issue Simulation case specification
«datastore»
Integration and Traceability Platform
«datastore»
CAD Tool

Figure 45. Verification workflow (Verification specification; Simulation case specification -part).

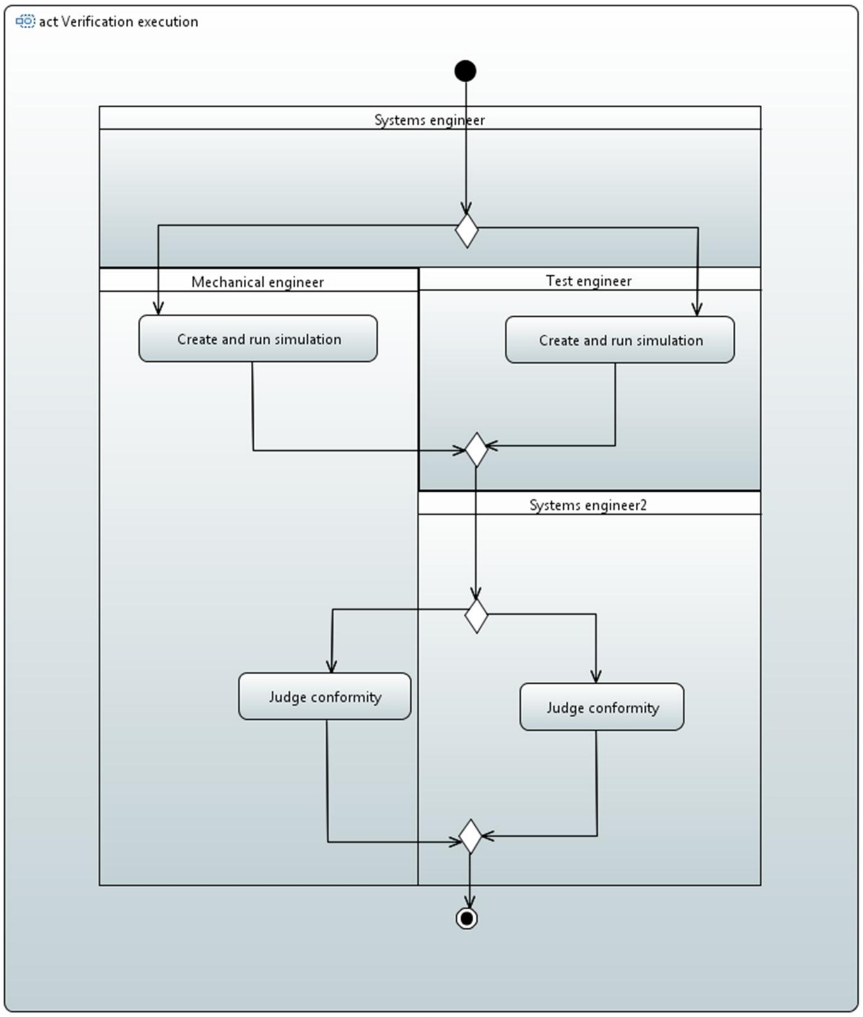


Figure 46. Verification workflow (Verification execution) (See Create and run simulation details in Figure 47 and Judge conformity in Figure 48).

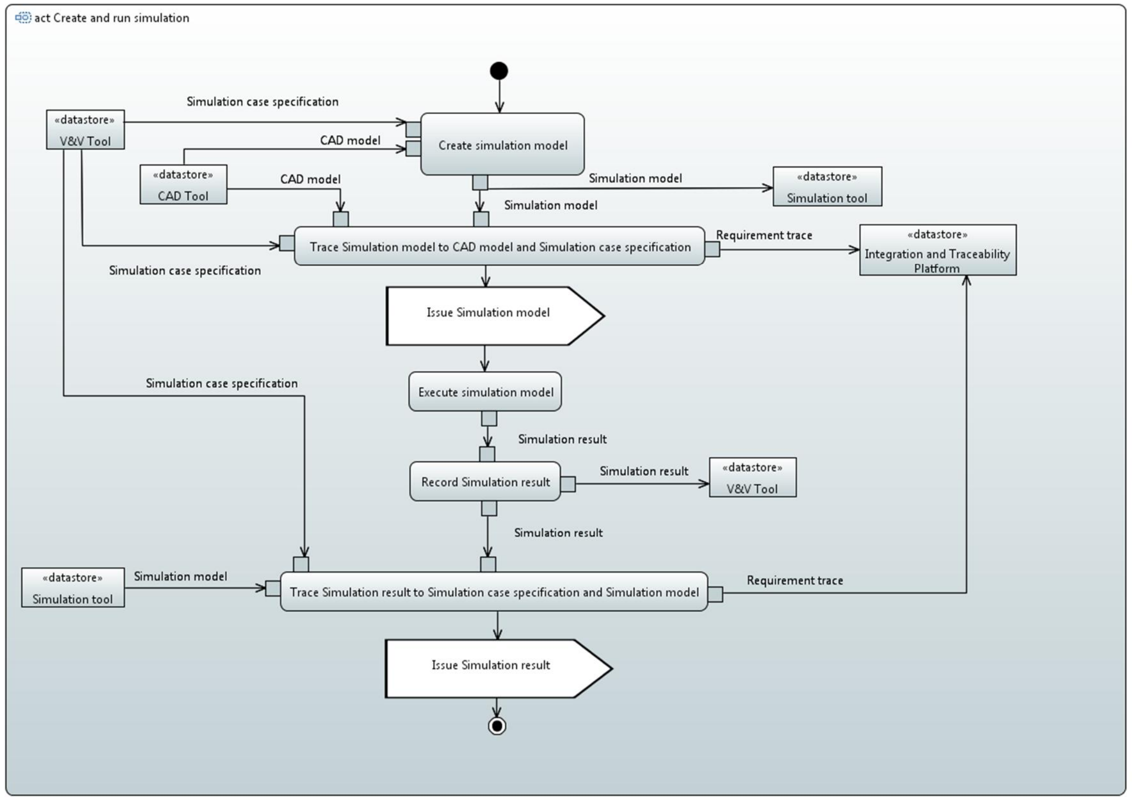

act Create and run simulation
«datastore» V&V Tool
Simulation case specification
CAD model
«datastore» CAD Tool
CAD model
Create simulation model
Simulation model
«datastore» Simulation tool
Simulation model
Trace Simulation model to CAD model and Simulation case specification
Requirement trace
«datastore» Integration and Traceability Platform
Simulation case specification
Issue Simulation model
Simulation case specification
Execute simulation model
Simulation result
Record Simulation result
Simulation result
«datastore» V&V Tool
Simulation result
«datastore» Simulation tool
Simulation model
Trace Simulation result to Simulation case specification and Simulation model
Requirement trace
Issue Simulation result

Figure 47. Verification workflow (Verification execution; Create and run simulation -part).

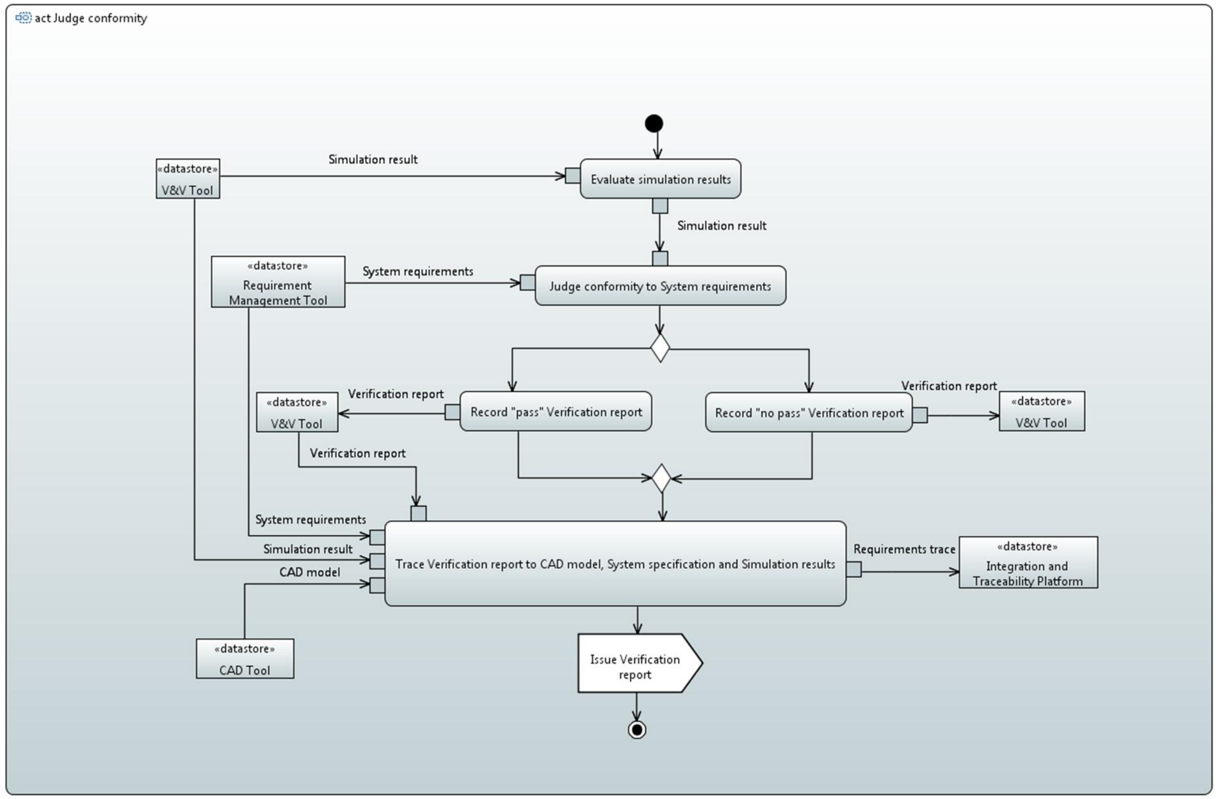

act Judge conformity
«datastore»
V&V Tool
Simulation result
Evaluate simulation results
Simulation result
«datastore»
Requirement Management Tool
System requirements
Judge conformity to System requirements
«datastore»
V&V Tool
Verification report
Record "pass" Verification report
Record "no pass" Verification report
Verification report
«datastore»
V&V Tool
Verification report
System requirements
Simulation result
CAD model
Trace Verification report to CAD model, System specification and Simulation results
Requirements trace
«datastore»
Integration and Traceability Platform
«datastore»
CAD Tool
Issue Verification report

Figure 48. Verification workflow (Verification execution; Judge conformity -part).

Series title and number

VTT Technology 236

Title	**Requirements traceability in simulation driven development**
Author(s)	Jarmo Alanen, Pekka Isto, Petri Tikka & Teemu Tommila
Abstract	The key success factors in development of complex systems are among the following: 1. systematic processes and life cycle model (such as ISO/IEC/IEEE 15288 and its daughter standards) 2. a systematic model for the engineering artefacts and their relations (such as specifications, CAD-models, pieces of information, and so forth) 3. An effective organisation model · well-defined roles and responsibilities (like systems engineer, requirements engineer, and so forth) · well-defined collaboration model (to facilitate consistent view in all involved organisations of the goal, data and state of the development) 4. well-planned use of project management and systems engineering tools · a good selection of engineering tools (model based tools advocated) · a flexible tool integration model (to allow integration of various tools used by the collaboration partners) 5. a tool to orchestrate all of the above (such as a PLM tool). This report addresses the second success factor, systematic model for the engineering artefacts. The particular focus is on the traceability of the engineering artefacts in simulation oriented systems development. The ultimate goal is to integrate the artefacts (such as simulation results) produced by the simulation engineers and their software tools with the other artefacts of the development processes; current practices easily leave simulation engineers on their distinct islands without knowing the original stakeholder requirements, the consequent system requirements and the specific rationale for a specific simulation task. Engineering artefacts include, among others, requirements specifications, system functions specifications, system architecture descriptions and verification and validation artefacts, simulation related artefacts being a part of the verification and validation artefacts. In complex systems, arrangements for traceability and impact analysis play an important role in managing the iterative systems development. To provide the traceability of engineering artefacts, a traceability information model and a tool to implement it is required. This report presents a systems engineering artefacts model titled SEAModel to facilitate creation of traceability information models. Some possible tools to implement SEAModel are discussed, and finally a demonstration done to ensure feasibility of SEAModel in simulation driven demonstration case is reported.
ISBN, ISSN, URN	ISBN 978-951-38-8359-1 (URL: http://www.vttresearch.com/impact/publications) ISSN-L 2242-1211 ISSN 2242-122X (Online) http://urn.fi/URN:ISBN:978-951-38-8359-1
Date	October 2015
Language	English
Pages	81 p. + app. 30 p.
Name of the project	SIMPRO - Computational methods in mechanical engineering product development'
Commissioned by	Tekes, VTT and industrial partners
Keywords	Systems Engineering, Requirements engineering, Mechanical engineering, Simulation, Traceability
Publisher	VTT Technical Research Centre of Finland Ltd P.O. Box 1000, FI-02044 VTT, Finland, Tel. 020 722 111

VTT TECHNOLOGY 236 Requirements traceability in simulation...

Requirements traceability in simulation driven development

The different results of an engineering project tend to be scattered to separate data repositories that store results from a single phase of the project such as requirements acquisition, system modelling or simulation. This makes it a challenge to maintain consistency between the various engineering artefacts as they evolve during the project.

This report provides data models for capturing the dependencies between the engineering artefacts and tracing the impacts of a change in the artefacts. Application of the models is demonstrated with a typical set of engineering tools. The report shows that the presented data models can be implemented to provide traceability with IT tools that provide a structured data repository, such as a relational database. Even though full integration of data from various engineering tools cannot be achieved with just trace links, arranging the traceability according to the presented models makes a significant step forward towards good management of shared engineering artefacts, and also towards model-based Systems Engineering.

The report is especially useful for SMEs that would like to improve maintaining the consistency of their engineering work products without investing in a full-blown Product Life Cycle Management tool.

ISBN 978-951-38-8359-1 (URL: http://www.vttresearch.com/impact/publications)
ISSN-L 2242-1211
ISSN 2242-122X (Online)
http://urn.fi/URN:ISBN:978-951-38-8359-1