VTT RESEARCH NOTES 2583

Jarmo Alanen, Iiro Vidberg, Heikki Nikula, Nikolaos Papakonstantinou, Teppo Pirttioja & Seppo Sierla

Engineering Data Model for Machine Automation Systems

VTT TIEDOTTEITA – RESEARCH NOTES 2583

Engineering Data Model for Machine Automation Systems

Jarmo Alanen & Iiro Vidberg

VTT

Heikki Nikula, Nikolaos Papakonstantinou,
Teppo Pirttioja & Seppo Sierla

Aalto University

ISBN 978-951-38-7711-8 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0865 (URL: http://www.vtt.fi/publications/index.jsp)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 5, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 5, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 5, P.O. Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax +358 20 722 4374

Cover photo by courtesy of Metso Minerals Oy

Jarmo Alanen, Iiro Vidberg, Heikki Nikula, Nikolaos Papakonstantinou, Teppo Pirttioja & Seppo Sierla. Engineering Data Model for Machine Automation SystemsEngineering Data Model for Machine Automation Systems. Espoo 2011. VTT Tiedotteita – Research Notes 2583. 131 p.

Keywords systems engineering, data model, risk assessment

Abstract

This research note presents a data model that defines the key artefacts of the systems engineering and safety processes of machine control systems. Existing data models, like the ISO 10303-233 and the German automotive specifications, especially MSRYS, are exploited when defining the model.

The model presented in this research note defines artefacts related to the overall system and its context, requirements and their validation, risk assessment, behaviour (system use cases and functional specifications), system structure and documentation. The emphasis is on defining the linkage between these artefacts also across process boundaries. These links form the traceability chains from risk assessments to safety requirements, functional specifications, designs, implementation, verification and validation. Besides providing traceability of artefacts, such a model allows centralized, single source, data repository implementations that ensure a consistent view of artefacts in different design disciplines, such as software development and electrical CAD.

Two implementations of the designed model are presented: Polarion ALM and MySQL with an MS Access front-end. Examples of tool integrations using the model are also demonstrated.

The benefits are also emphasized from the perspective of documentation. A centralized artefact repository can support the automated creation of documents based on demand. For example, the system requirements specification and the relevant parts of the technical file, which is required by the Machinery Directive, can be generated from the database.

Preface

In 2008, FIMA ry (Forum for Intelligent Machines) initiated a project to study the possibilities of creating a data model for the systems engineering artefacts relating to the development of programmable control systems of mobile work machines. The goal was to make the development of complex control systems more efficient for the machine manufacturers and to facilitate easy co-operation with their sub-contractor networks. The goal was to be achieved by:

- specifying an industry-wide data model of the systems engineering design artefacts that was independent of the tools used
- specifying unified development process workflows that used the data model
- demonstrating integration of off-the-shelf systems engineering tools via the implementation of the data model.

The project was named TIKOSU (Tietokantakeskeinen koneenohjausjärjestelmän suunnittelu – Database-centric development of machine control systems). The TIKOSU project was accepted into the Digital Product Process technology programme by Tekes (the Finnish Funding Agency for Technology and Innovation). The project started in January 2009 and ended in May 2011. It was financed by Tekes, FIMA, Aalto University and VTT.

The project was managed by the following steering group members:

- Kari Koskinen, Aalto University
- Jani Sarviluoma, Atostek Oy
- Jouni Törnqvist, Bronto Skylift Oy
- Pekka Yli-Paunu, Cargotec Finland Oy
- Marko Elo, CC Systems Oy
- Soini Särkilahti, Cybercom Oy
- Arto Orava, Epec Oy
- Esa Niemelä, FIMA ry
- Mikko Mäkinen, Metso Minerals Oy (later substituted by Toni Kujala from the same company)
- Jarkko Uotila, Sandvik Mining and Construction Oy
- Jyrki Keskinen, Wapice Oy
- Pauli Noronen, Tekes
- Risto Tiusanen, VTT.

The project was carried out by the following researchers:

- Research scientist, Jarmo Alanen, VTT (coordinator of the project)
- Research scientist, Kimmo Kauvo, VTT

- Research scientist, Iiro Vidberg, VTT
- Research scientist, Heikki Nikula, Aalto University, Department of Automation and Systems Technology
- Research scientist, Nikolaos Papakonstantinou, Aalto University, Department of Automation and Systems Technology
- Research scientist, Teppo Pirttioja, Aalto University, Department of Automation and Systems Technology
- Dr. Sven Scholz, Dresden University of Technology, 'Friedrich List' Faculty of Transportation and Traffic Sciences
- Research scientist, Seppo Sierla, Aalto University, Department of Automation and Systems Technology
- Prof. Kleanthis Thramboulidis, University of Patras, Department of Electrical and Computer Engineering.

This research note pulls together the results of the project in a concise format.

The authors would like to thank all the steering group members and their colleagues for their interest and valuable feedback on the work.

Tampere, May 2011

Authors

Jarmo Alanen, Heikki Nikula, Nikolaos Papakonstantinou, Teppo Pirttioja, Seppo Sierla, Iiro Vidberg

Contents

List of abbreviations

Abbreviation	Description
ALM	Application Lifecycle Management
API	Application Programming Interface
ARM	Application Reference Model
AUTOSAR	Automotive Open System Architecture Standard
CAD	Computer Aided Design
CAN	Controller Area Network
CANopen	An application layer specification for CAN
CCF	Common Cause Failure
DC	Diagnostic Coverage
DOM	Document Object Model
ECU	Electronic Control Unit
EER	Enhanced Entity-Relationship
EXPRESS	A data modelling language defined for the ISO STEP standards
EXPRESS-G	A graphical notation of EXPRESS
FMEA	Failure Mode and Effects Analysis
FTA	Fault Tree Analysis
FTP	File Transfer Protocol
HAZOP	Hazard and operability study
HIL	Hardware-in-the-Loop
HMI	Human Machine Interface
IEC	International Electrotechnical Commission
IFA	Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung
IIDAbase (model)	Integrated Industrial Design Artefacts database. An artefact repository model that integrates the design artefact data of a programmable control system
INCOSE	The International Council on Systems Engineering
ISO	International Organization for Standardization
JDBC	Java Database Connectivity
MEDOC	MSR Engineering Data Objects and Contents
MSR	Manufacturer Supplier Relationship

Abbreviation	Description
$MTTF_d$	Mean Time to Dangerous Failure
ODBC	Open Database Connectivity
OID	Object Identifier
OMG	Object Management Group
OMG SE DSIG	Object Management Group's Systems Engineering Domain Special Interest Group
PFH	Probability of Failure per Hour
PHA	Preliminary Hazard Analysis
PL	Performance Level (according to ISO 13849-1)
PLC	Programmable Logic Controller
PLCopen	An organization to promote IEC 61131-3 programming
PLM	Product Life Cycle Management
RTF	Rich Text Format
SAX	Simple API for XML
SDO	Service Data Object
SE	System Engineering
SEDRES	Systems Engineering Data Representation and Exchange Standardisation
SGML	Standard Generalized Mark-up Language
SISTEMA	Safety Integrity Software Tool for the Evaluation of Machine Applications
SRP/CS	Safety-Related Part of a Control System
SQL	Structured Query Language
STEP	Standard for the Exchange of Product model data (formally defined as ISO 10303 Product data representation and exchange)
SysML	Systems Modelling Language
SVN	Subversion (version management software)
SW	Software
TIKOSU	Project that develops the IIDAbase data model. Project name: Database-centric design of machine control systems
UCSA	Use Case Safety Analysis
UML	Unified Modelling Language
WI	Work Item

Abbreviation	Description
XML	Extensible Mark-up Language
XSD	XML Schema Definition
XSLT	Extensible Style-sheet Language Transformations

1. Introduction

Systems engineering processes for complex control systems for mobile machines still rely on very basic use of information technology. For example, system requirements and functional specifications are very often stored, transferred and presented in word processing documents. As the complexity of control systems increases, it is difficult, if not impossible, to manage the ever-changing set of systems engineering artefacts and their traces to each other such that, e.g., the technical file required by the Machinery Directive can be produced efficiently for the right version and variation of the control system. Hence, it is necessary to exploit the data management and workflow support capabilities of modern information technology in order to cope with the challenges of legislation and system complexity.

Fortunately, there are many tools available to systems engineers to alleviate the pain of developing complex systems. These are often stand-alone tools, however, focusing on one design discipline at a time with poor integration capabilities with each other. The original source of artefacts may also be obscure. Hence, there can be overlapping information content in the repositories of different tools, making projects prone to problems of data inconsistency, for example, when the system engineer draws a system architecture diagram and names the signals that flow between the various modules, the module software developer can use different names for the same signal flows. This causes confusion, especially during debugging and fault finding in the maintenance phase, and breaks the traceability chains that provide evidence of the fulfilment of safety requirements.

One solution to this problem is a comprehensive data model that specifies the types, content and relationships of systems engineering artefacts. Such models are presented in, e.g., the ISO AP233 standard (ISO 10303-233) and the MSRSYS specification created by German automotive companies. Their applicability to the context of machine control systems is poor due to negligible tool support and because they do not embrace the risk assessment model of ISO 14121-1, the Machinery Directive's harmonized standard for risk assessments. Hence, a tailored model for the context of work machines is needed. In the TIKOSU project, carried out under the Tekes Digital Product Process technology programme, a systems engineering data model for supporting the management and traceability of specification, design and verification artefacts was created. This model is presented in this research note and here it is called the IIDAbase model (Integrated Industrial Design Artefacts database) (see Figure 1).

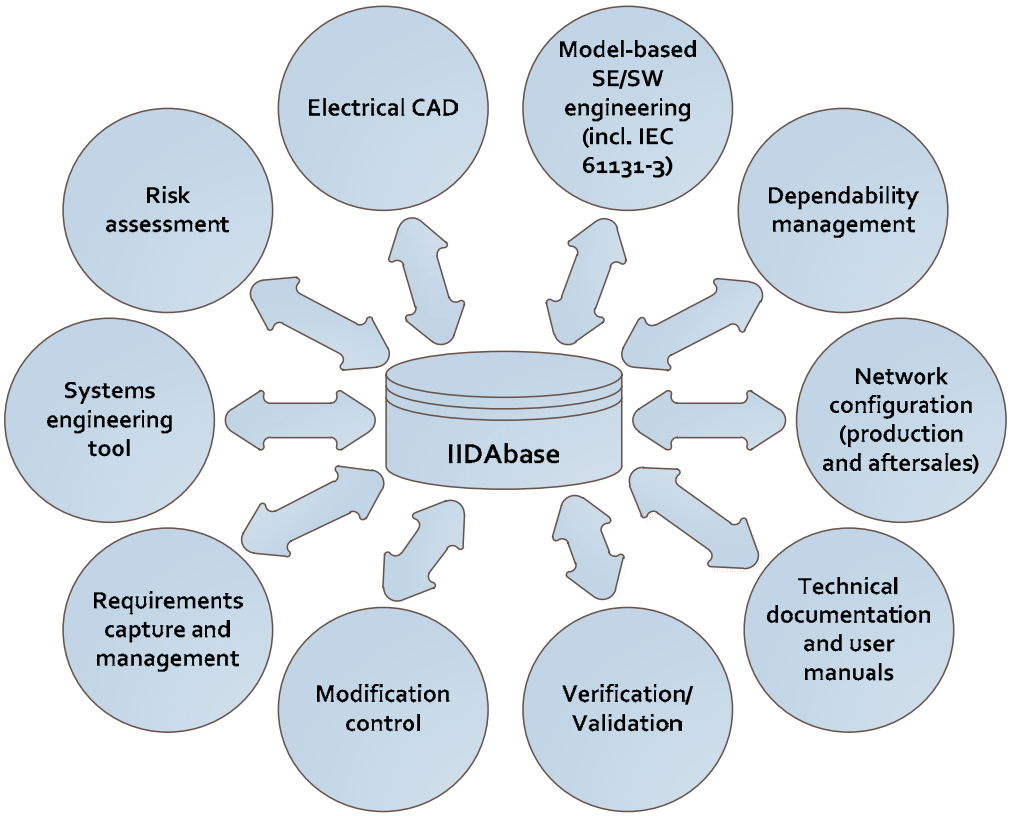


Figure 1. IIDAbase context (only some of the systems engineering processes and tools are depicted).

The IIDAbase model defines the key artefacts of the systems engineering and safety processes. The emphasis is on defining the linkage between these artefacts also across process boundaries. These links form the traceability chains from risk assessments to safety requirements, functional specifications, designs, implementation, verification and validation. The data model, supporting tools and workflows defined in TIKOSU support and enforce the creation and maintenance of these links, even when the systems and safety engineers work in different organizations. In its current state, the model defines artefacts related to the overall system and its context, requirements and their validation, risk assessment, behaviour (system use cases and functional specifications), system structure and documentation. The model has been demonstrated both on a commercial Application Lifecycle Management tool (Polarion) and in a database environment (MySQL with an MS Access interface).

The advantages of such an implementation are as follows:

- There is only a single source of information. This makes the development and version control of complex control systems less error prone.
- Traceability of artefacts, such as from requirements to design and validation, can be provided.
- Evidence of fulfilling the Machinery Directive can be pointed out more systematically, and the automatic generation of documentation, such as parts of the Technical File, becomes possible.

- A structured data model facilitates management of sub-contractor work as all participants do their work on the same data repository, and the workflows and boundaries of the work tasks can easily be set.
- The number of tool integrations is reduced (theoretically *1 x n* instead of *n x (n-1) / 2*, where *n* is the number of systems engineering tools).

This research note presents the artefact model and its implementation demonstrations, and provides examples of tool integrations using the model. The benefits are also emphasized from the perspective of automation creation of documentation.

2. Existing systems engineering data models

The idea of a centralized data repository for systems engineering data is not unique. There are models and tools for different fields of engineering. During the project, a review of such models and some of the tools was carried out. The review included the following[1]:

- Föderdal information architecture
- AutomationML
- PG-Pla-INC project
- GENESYS project
- Vector informatik's eAsee tool
- CANopen XML specification
- PLCopen XML specification
- AP233 of ISO 10303
- ASAM automotive specifications.

Of these, the AP233 standard and one of the ASAM specifications, MSRSYS, are closest to the scope and status of the current practices in machine automation systems engineering and are thus presented in more detail in Sections 2.1 and 2.2 respectively. The following table (Table 1) presents all of the above models, specifications and tools, however, in a concise format.

[1] Besides these, a short state-of-the-art review in the field of the building industry, automation industry and automotive industry was made.

Table 1. Systems engineering data specifications and tools under review.

	Description	Relevance here	More information
Föderdal information architecture	Definition of a model-driven engineering framework that supports reusability of (mechatronic) components	Interesting, but its implementation model (Eclipse framework) was perceived unsatisfactory for the scope of this work	http://www.foederal.org http://www.aquimo.org
AutomationML	Data exchange format based on XML for manufacturing tools integration	Not relevant (except for the PLCopen XML specification) due to poor support for control system development tools	http://www.automationml.org
PG-Pla-INC project	Information-centric development of component-based embedded real-time systems	Not highly relevant, because it concentrates on software engineering (especially on component-based SW engineering), not on systems engineering	http://www.mrtc.mdh.se/index.php?choice=projects&id=0088
GENESYS project	Defines a cross-domain reference architecture for embedded systems. Presents an outline for an integrated development environment	Presents only an outline for the integrated development environment; it can be claimed that the outlined model fits the one implemented in this work	http://www.vtt.fi/inf/pdf/publications/2009/P705.pdf
Vector informatik's eAsee tool	Tool environment for managing all of the process and product data of complex software and electronic automotive systems	Automotive-specific only. Could not be tailored for machine automation	http://www.vector.com/vi_easee_as_1_en.html
CANopen XML specification	A file format specification for electronic data sheets and configuration data of CANopen devices	Was considered relevant but was not implemented due to the constraints on project resources	CiA DSP 311 CANopen XML-based device description (contact CAN in Automation organization)
PLCopen XML specification	Specifies a file format for interoperability of PLC projects based on the IEC 61131 standard	Was considered relevant. Was used in this work for the integration of PLC tools	http://www.plcopen.org/pages/tc6_xml
AP233 of ISO 10303	Application protocol for systems engineering data	Highly relevant but too complex for the current state of systems engineering; poor availability of tools. Was used as a reference for this work	http://www.ap233.org
ASAM automotive specifications	A set of specifications to structure the data of automotive electronics systems and their development process	Automotive-specific, but some of its concepts were used in this work	http://www.msr-wg.de/medoc/downlo.html

2.1 AP233

AP233 is a systems engineering data model standard. It is a STEP[2]-based data exchange standard targeted at systems engineering tools, and it is consistent with emerging standards in CAD, structural, electrical, engineering analysis and support domains. AP233 collaborates with the OMG SE DSIG[3] team (develops system engineering extensions for SysML) and INCOSE (the International Council on Systems Engineering). [1]

AP233's formal name is *ISO 10303-233 Product data representation and exchange – Application Protocol – Systems engineering.* As the name implies, AP233 falls exactly within the scope of this study.

AP233 is based on ISO/PAS 20542 [2], which was developed in the European projects SEDRES (1996-1999) and SEDRES2 (2000–2001). SEDRES is an abbreviation of Systems Engineering Data Representation and Exchange Standardisation. AP233 has evolved a long way from ISO/PAS 20542.

Note: AP233 is still undergoing standardization in ISO at the time of publication of this research note; all the information presented in this report is thus subject to change.

According to [3], AP233 is designed to:

- *"capture lifecycle system requirements, design, etc.*
- *support integration of systems engineering tools*
- *enable INCOSE vision of Model-Based Systems Engineering*
- *align with OMG SysML*
- *provide a 'front end' to PLCS-based support engineering tools*
- *link with detailed design, PDM, analysis, etc. through other STEP protocols."*

[2] STEP (Standard for the Exchange of Product model data, formally defined as ISO 10303 Product data representation and exchange).

[3] The Object Management Group's Systems Engineering Domain Special Interest Group.

AP233's main motivation, to exchange data between systems engineering tools, is well depicted in Figure 2:

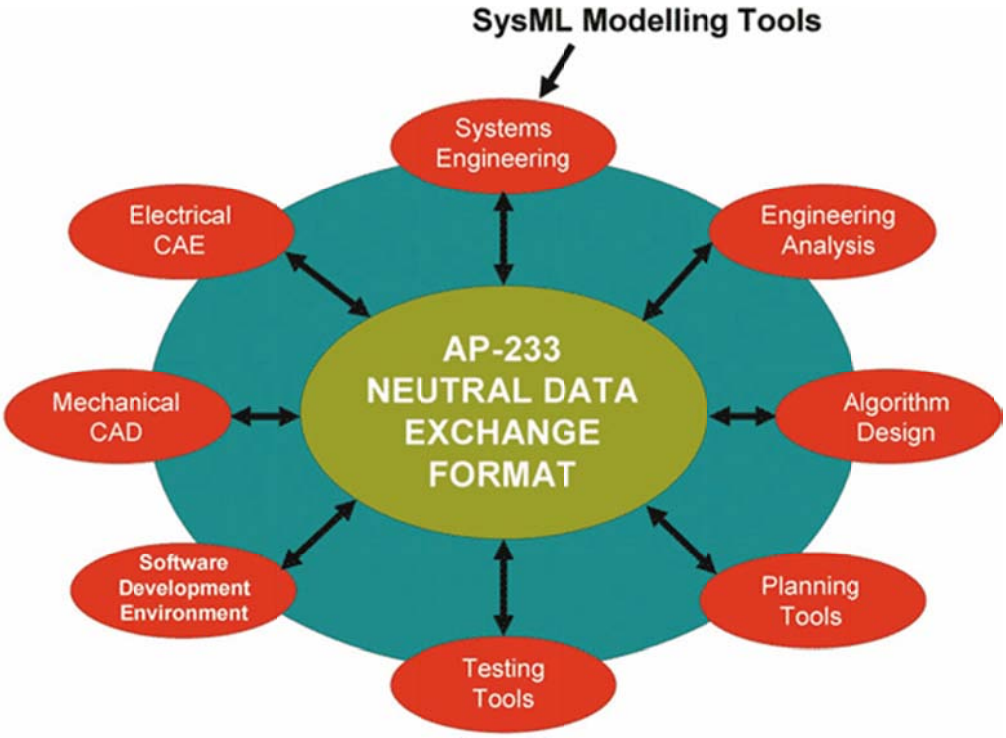


Figure 2. AP233's main motivation: to exchange data between tools [4].

As can be noted, the aim of the AP233 standard is practically the same as that of the IIDAbase model, though there is a slight difference: IIDAbase is not meant to be a data format for exchanging artefacts between tools but a central artefact repository that relieves us of the need to transfer information from one tool to another. In practice, IIDAbase will need to transfer information in many cases, but the transfer will occur between the tool and the central repository. The exchange format of AP233, on the other hand, can in theory be used as a central repository data format.

Figure 3 presents a breakdown of AP233 and shows the hierarchy of the concepts.

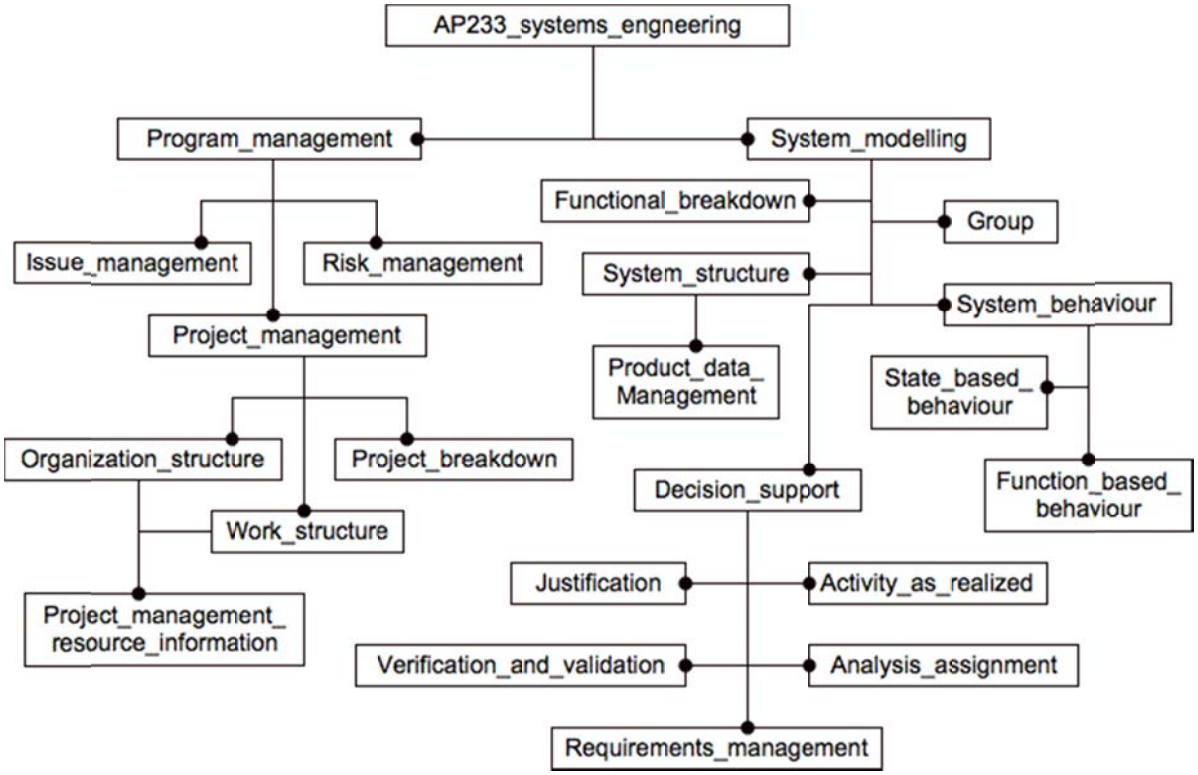


Figure 3. Top-level hierarchy of ISO 10303-233 modules [5].

The *System_modelling* branch is of special interest in this study. It should be noted here that AP233 is a top-level standard: the modelling elements are defined in other parts of ISO 10303, the top-level modules of AP233 being defined in ISO 10303-433 and, e.g., the *System_modelling* module (see Figure 3) is defined in ISO 10303-1477, but when the AP233 standard is purchased the relevant parts of the ISO 10303 standard are included.

In the following, two example figures (Figure 4 and Figure 5) of two of the models in Figure 3 are provided, namely the *Function_based_behaviour* model and the *System_structure* model:

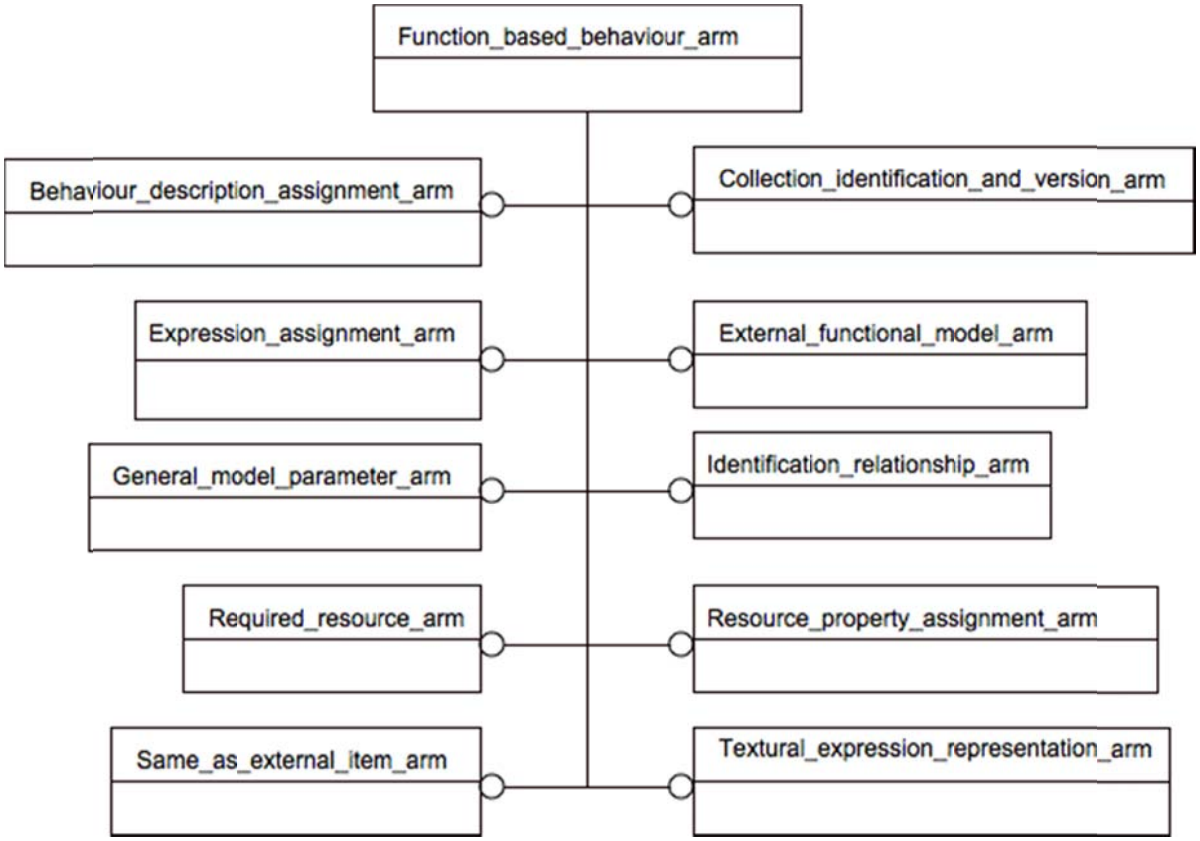


Figure 4. Application reference model (ARM) for function-based behaviour [5].

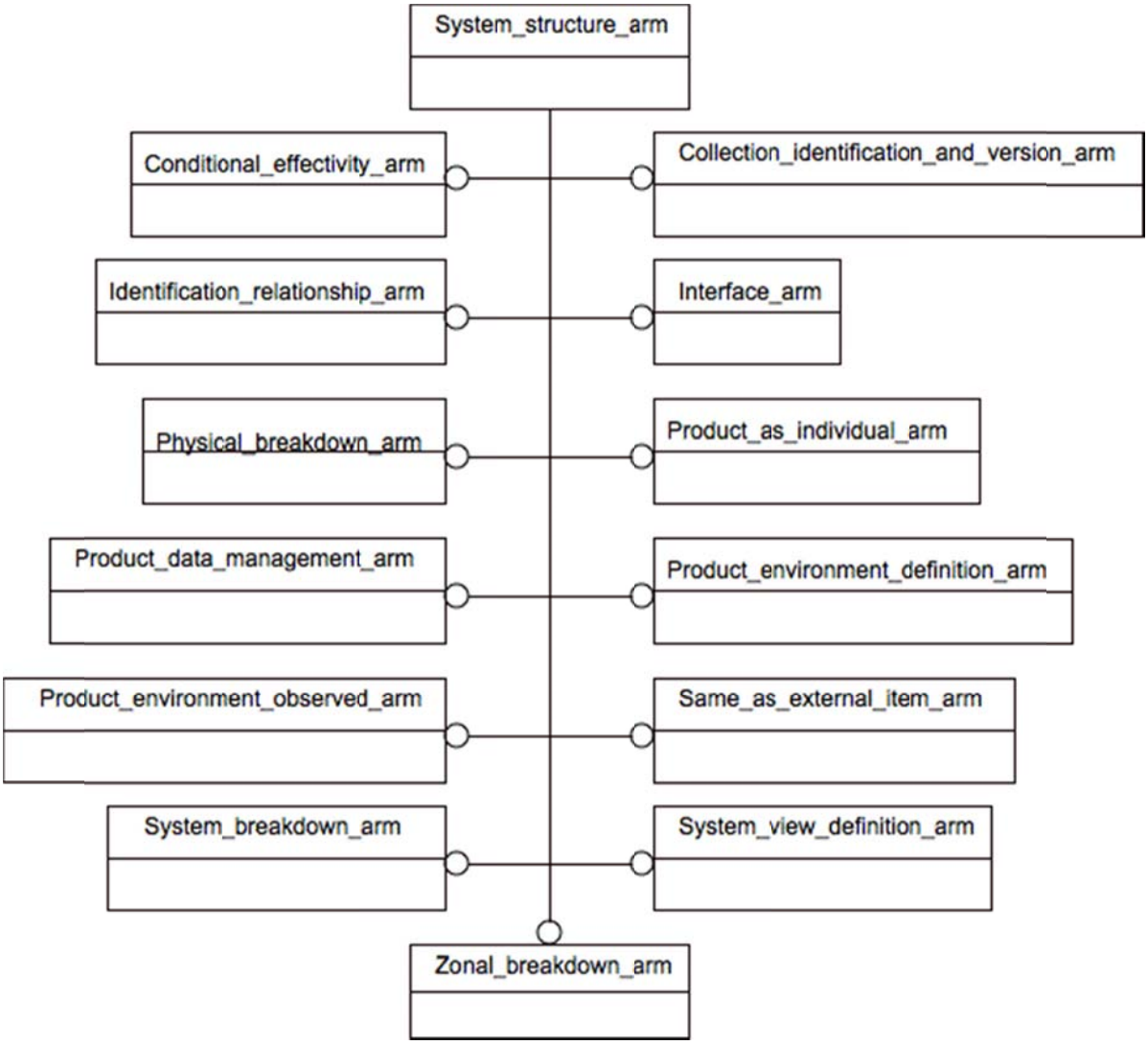


Figure 5. Application reference model (ARM) for system structure [5].

In its Appendix H, AP233 provides a systems engineering concept model that satisfies the requirements, views and semantics of different stakeholders including the ISO AP233 development team, INCOSE and OMG. Figure 6 depicts the top-level systems engineering concept model.

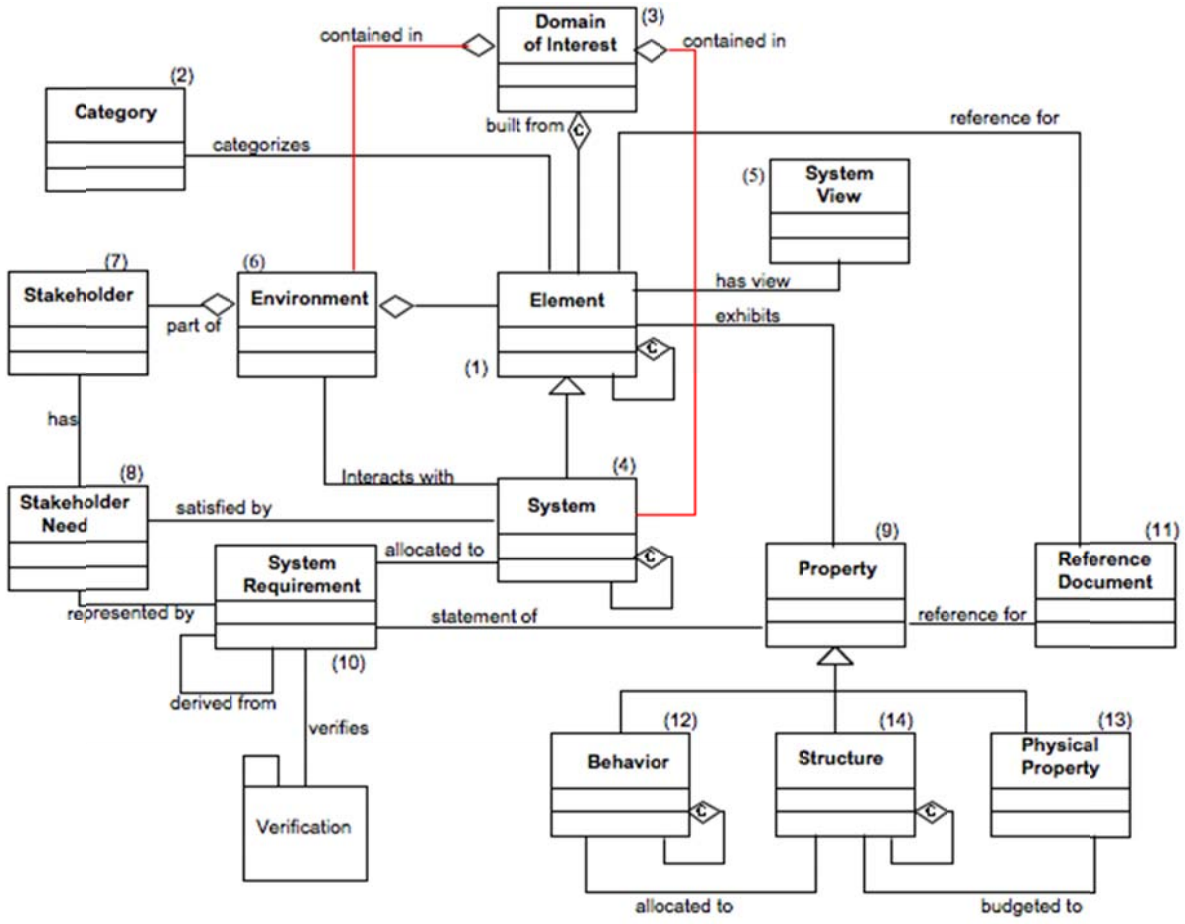


Figure 6. Top-level systems engineering concept model [5].

As an example, Figure 7 illustrates the breakdown of the *Structure* class.

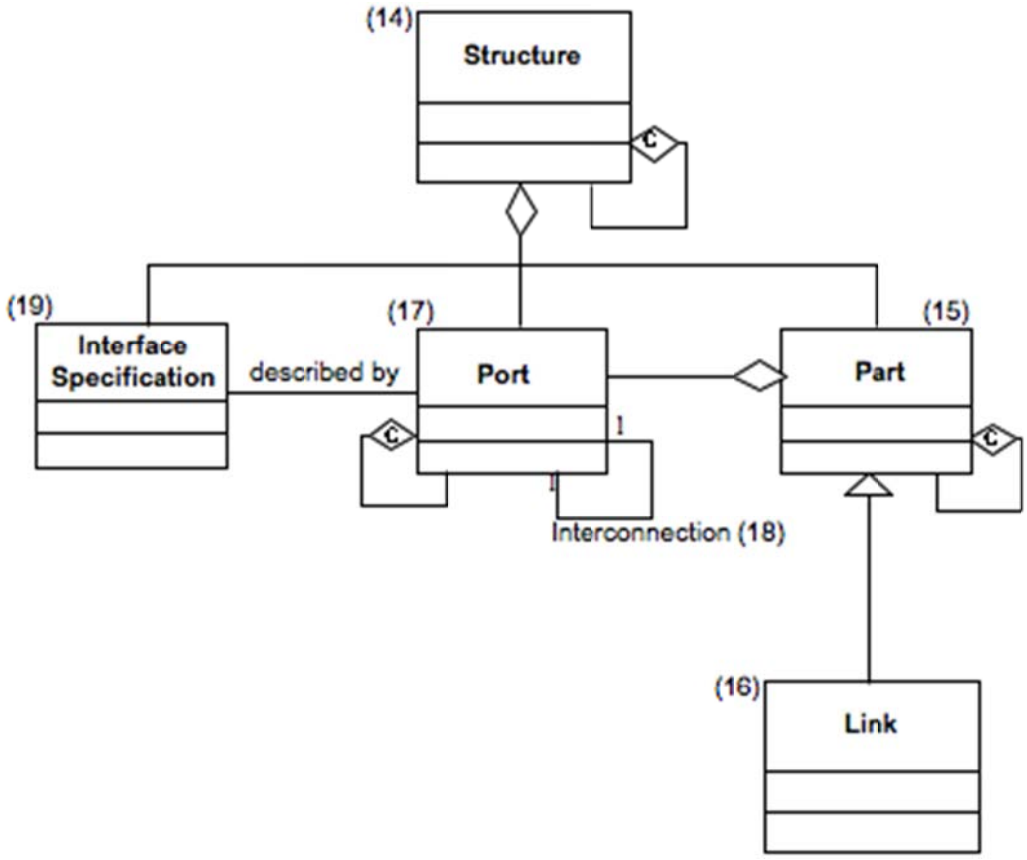


Figure 7. Concept model for system static structure [5].

Furthermore, Figure 8 illustrates a breakdown of the *Behavior* class.

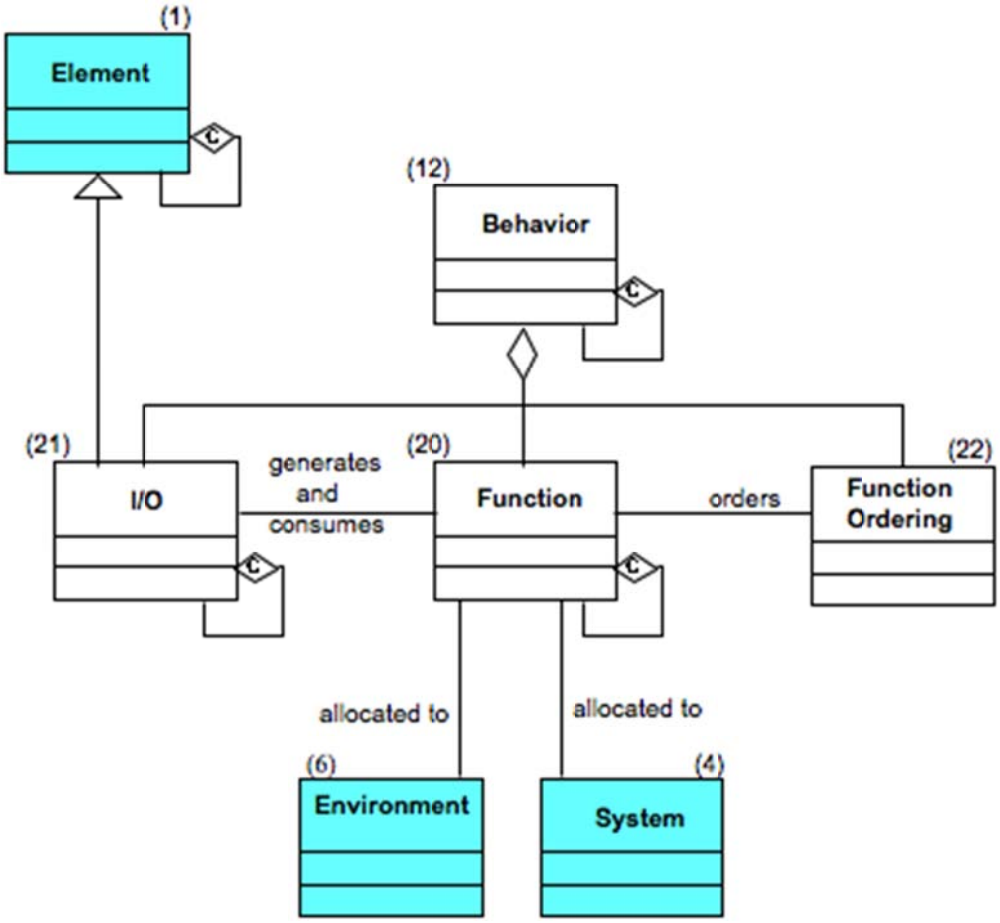


Figure 8. Concept model for system behaviour [5].

When describing the data elements, AP233 uses EXPRESS language, though XML will be provided by other organizations than ISO. EXPRESS is a language defined for the STEP standards. As an example of EXPRESS language notation, the specification of the data element 'Product' is provided below (source: ISO/TS 10303-1017):

Program 1. Example of ISO EXPRESS notation; definition of the artefact 'Product'.

```
ENTITY Product;
  id : STRING;
  name : STRING;
  description : OPTIONAL STRING;
END_ENTITY;
```

The corresponding EXPRESS-G graphical notation is as follows (Figure 9):

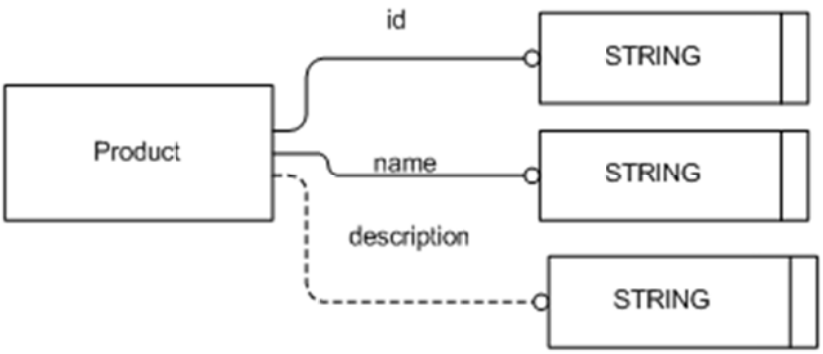


Figure 9. Data element ‘Product’ in EXPRESS-G notation [5].

Relationships between the data elements can best be presented in EXPRESS-G notation (see Figure 10), but the graphical notation can be provided as an EXPRESS listing, as supplied in Program 2 ([5]).

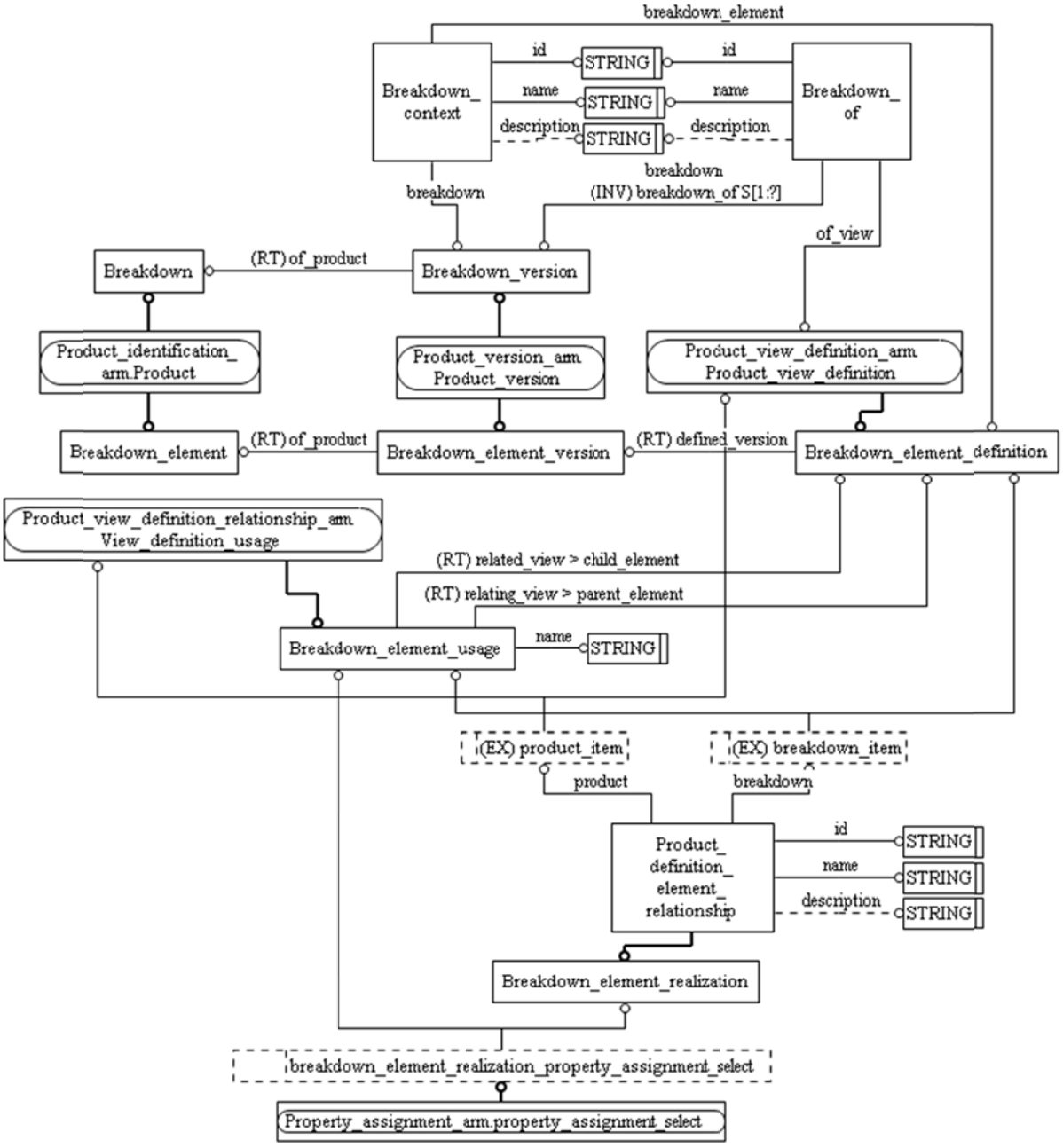


Figure 10. Product breakdown Application Module in EXPRESS-G notation [5].

Program 2. An excerpt of the EXPRESS listing that represents the graphical notation in Figure 10.

```
ENTITY Breakdown_context;
  id : STRING;
  name : STRING;
  description : OPTIONAL STRING;
  breakdown : Breakdown_version;
  breakdown_element : Breakdown_element_definition;
END_ENTITY;

ENTITY Breakdown_version
  SUBTYPE OF (Product_version);
  SELF\Product_version.of_product : Breakdown;
INVERSE
  breakdown_of : SET[1:?] OF Breakdown_of FOR breakdown;
END_ENTITY;

ENTITY Product_version;
  id : STRING;
  description : OPTIONAL STRING;
  of_product : Product;
END_ENTITY;
…
```

AP233 incorporates a risk management module. AP233 defines 'Risk' as follows: "*Likelihood and impact of failure to meet any technical or development program goal*", but it in principle the model can be used for creating a data model for safety engineering. The risk management model is very complex, however, compared with what is required by the Machinery Directive (especially the risk assessment standard ISO 14121-1). In Figure 11, part of the application reference model (ARM) entity level diagram of 'Risk' is provided to show the complexity of the data models (note that the diagram in Figure 11 is only part 1 of a total of four diagrams).

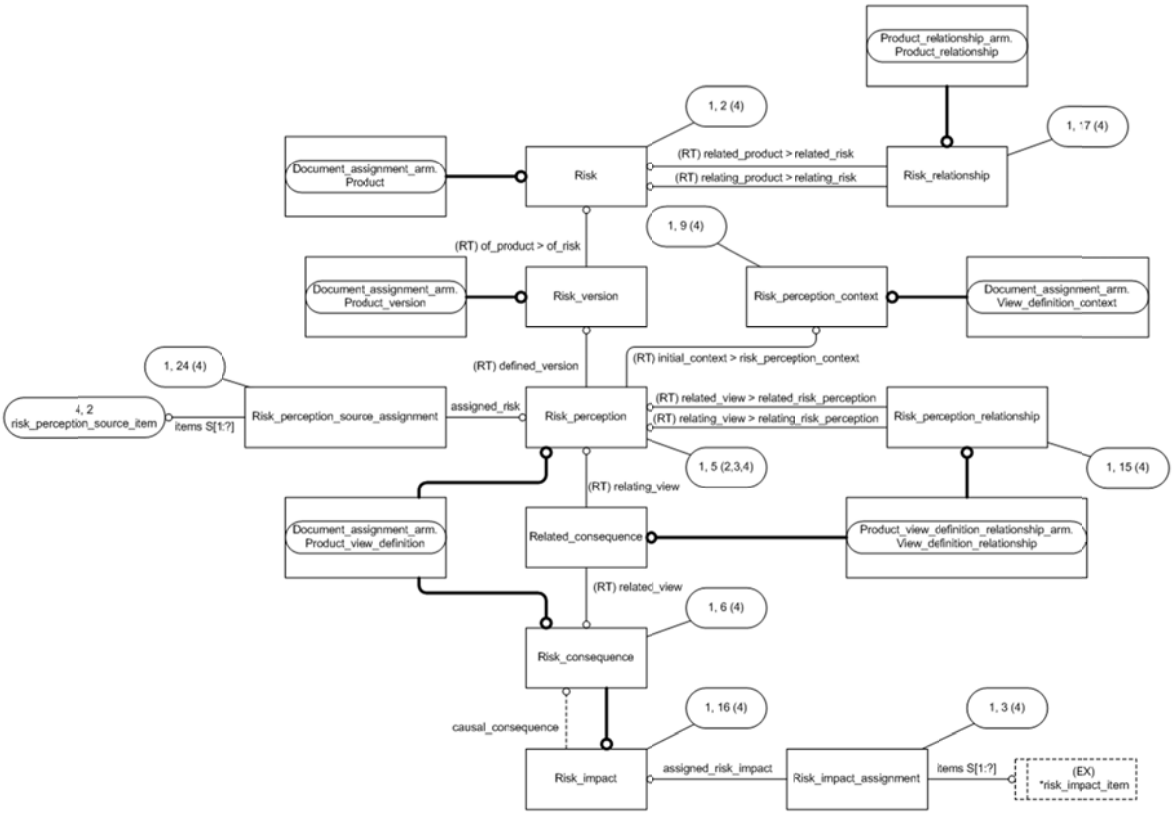


Figure 11. ARM entity level diagram of 'Risk' (part 1 of 4) [5].

AP233 is closely related to AP239, Product Life Cycle Support (PLCS); in fact, according to ISO 10303-233 Appendix H, 70% of the modules are common between AP233 and AP239.

AP233 has been on the scenes since 1996 if we count the time starting from the SEDRES projects, and the standard is still not ready (in March 2011). At the time of writing this research note, AP233 is at the DIS stage. All this informs us about a very exhaustive standard and hence very toilsome implementation of tools; Figure 10 and Figure 11 above expose only a small fraction of the complexity of the systems engineering data model of AP233. Nevertheless, AP233 is so exact in the scope of this study that it cannot be completely disregarded. The top-level models and concepts are used as a reference for the IIDAbase model as much as possible to facilitate migration to the AP233 data model as soon as the standard is accepted by tool vendors.

2.2 MSRSYS

An interesting set of specifications among the ASAM standards is the documents originating from the MSR organization (http://www.msr-wg.de [Referenced 21.03.2011]). MSR stands for Manufacturer Supplier Relationship. The particular set of MSR standards is as follows:

Table 2. The MSR standards.

Standard	Title
ASAM AAS MSRSW	MSR Description for Software of ECUs
ASAM AAS MSRSYS	ECU Control System Description
ASAM AAS MSRDCI	Document Control Information
ASAM AAS MSRFMEA	Failure Mode and Effects Analysis
ASAM AAS MSRMEPRO	Methodology for Engineering Process-Synchronization

The MSR standards are part of the MSR MEDOC collection in which MEDOC stands for MSR Engineering Data Objects and Contents. The objective of MSR/MEDOC according to [6] was:

- *"Seamless, continuous and consistently structured product documentation for electric/ electronic components and systems across the entire life cycle e.g. for requirement specification*
- *Support of views appearing within the development cycle*
- *Integration of heterogeneous IT environments Product Data Management / Technical Data Management."*

This sounds very similar to the objectives of our work. Of the MSR standards, the first two are especially interesting, MSRSYS and MSRSW; they define the data model for electrical hardware systems and components as well as for software for electronic control systems. MSRSYS defines the system as consisting of the following specification items [7]:

- General Product Data 1
    - Introduction
    - Product Description
    - Function Overview
    - Key Data
    - Product Demarcation
    - Similar Products
    - General Hardware Description
    - General Software Description
    - General Interfaces
    - Failure Management
    - Resource Allocation
    - Calibration

    - Safety
    - Quality
    - Maintenance
    - General Conditions
    - Additional Design Documentation
    - Development Process Specifications
    - Additional Specifications
- General Test Specification
- Behaviour
- Architecture
    - Scheme Diagrams
    - Interface Specification
    - Signal Specification
    - Connection Components Specification
    - Connection Specification
    - Network Specifications
    - Additional Specifications
- Electrical Characteristics
- Environmental Characteristics
- Other Physical Characteristics
- Construction
- Human Machine Interface
- Additional Specifications
- Parts in Part Type.

To model this data, 758 data elements have been defined. The data models are described by SGML. An extract of the contents of the element and attributes document [8] is provided in Figure 12. The figure also depicts two examples of diagrams of data elements, the definition of overvoltage resistance test and the definition of electrical parameters of a port (i.e., an I/O point).

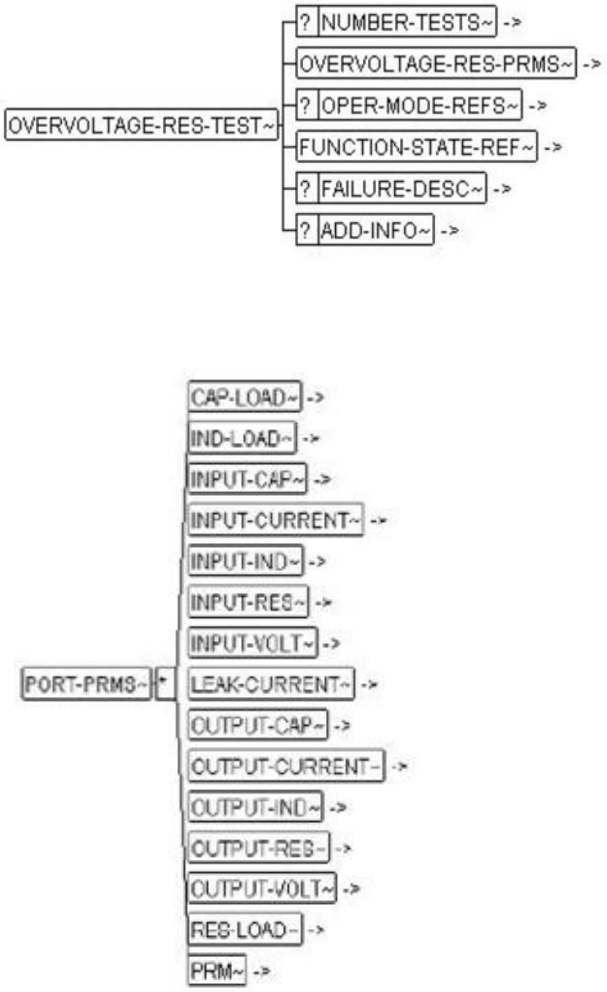


Figure 12. Excerpts of the MSRSYS element and attributes document [8].

In a similar way, MSRSW defines the data model for the software specification items and related items with a total of 777 data elements. MSRSW is a structure used for specifying and documenting software for electronic control units. It includes a data dictionary, functional specifications and calibration parameters.

In describing the software, MSRSW does not seem to relate closely to any programming language. It uses the general concepts of SW components, a data dictionary that includes global variables, parameters and classes (as in the object-oriented paradigm). Class implementations are defined as well as class instances.

Whereas PLCopen XML describes software in a way that is strongly depended on the IEC 61131-3 standard [9], using function blocks and the other languages described in the IEC 61131-3, MSRSW provides a more general description using a data dictionary and the concept of classes. It does not seem to prefer an execution environment or a specific programming language although it was designed to be used for storing information about ECU software.

MSRSW was used as an external interface to a system design tool (ASCET/SD by ETAS). MSRSW does not relate directly to UML. UML is a graphical way of describing and designing software while MSRSW is used as a structure to hold the information about the ECU software. MSRSW is now being replaced by work done in AUTOSAR. There are also migration scenarios from MSRSW to AUTOSAR. The key members of AUTOSAR are the same as the ones in MSR.

The MSR specifications are an interesting reference for our work, but the difference between the automotive field and the machine automation field makes the adoption possibilities of the specifications minimal. The MSR documents have not been updated since 2002. This is a fact that indicates poor acceptance by the automotive industry. The MSR specifications are very elaborate, which may have resulted in problems providing tools that follow the specifications. MSR specifications as such are not an option for the machine automation data model roadmap, but the specifications can work as a reference implementation of the paradigm of well-modelled design artefacts data. In fact, some ideas of the MSRSYS specification have been adopted in the IIDAbase concept model. An example of this is the adoption of the specification parameter concept in the context of the part type library (see Section 3.3).

3. IIDAbase concept model

3.1 Introduction

The model described in this research note was created in the TIKOSU project. We call the model the IIDAbase model. It has some influence from the AP233 standard [5] and MSRSYS [7], and it also accommodates some issues from the CANopen XML standard [10] to provide easier interfacing with CANopen tools as soon as XML-based datasheets from CANopen device vendors start to appear.

The AP233 standard has not been adopted as such; that would have been an impossible and impractical effort in the current context, especially due to the lack of tools that support AP233. Some hints from the top-level hierarchy of the AP233 were adopted for the IIDAbase model however (see Figure 13).

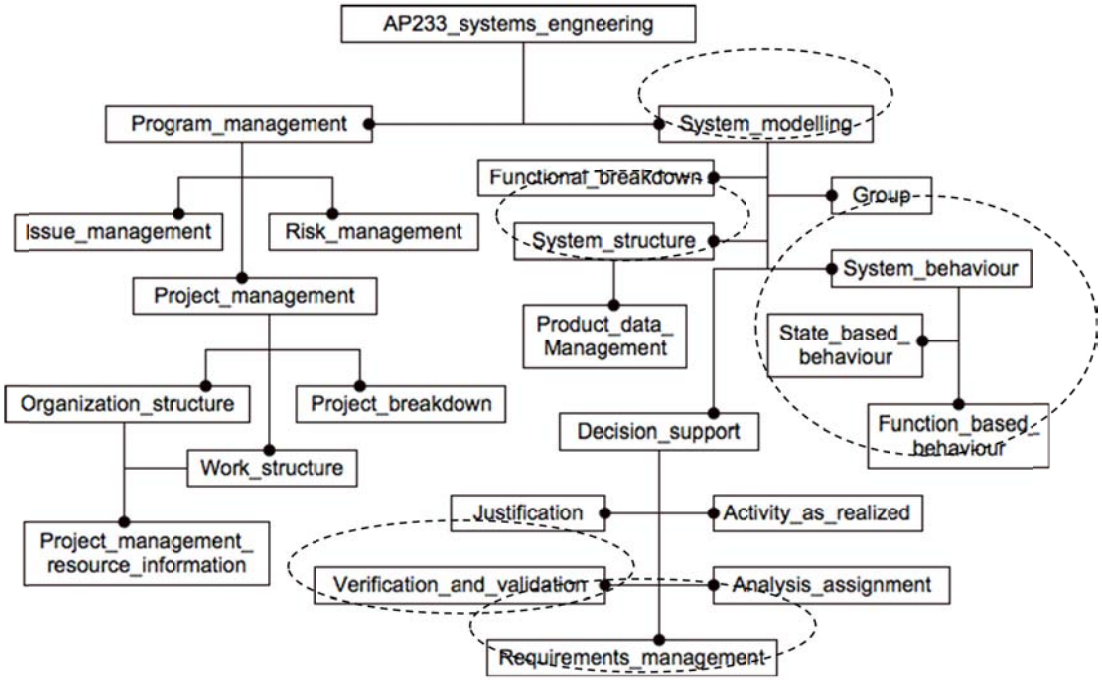


Figure 13. Top-level hierarchy of AP233 [5]; the dashed ovals point out the modules that are covered by the IIDAbase model (note that the risk assessment part of the IIDAbase model is included in the Verification_and_validation module, not in the Risk_management module).

The top-level concept model behind the AP233 data model was also used as a reference for creating the IIDAbase model. The concept model is depicted in Figure 14.

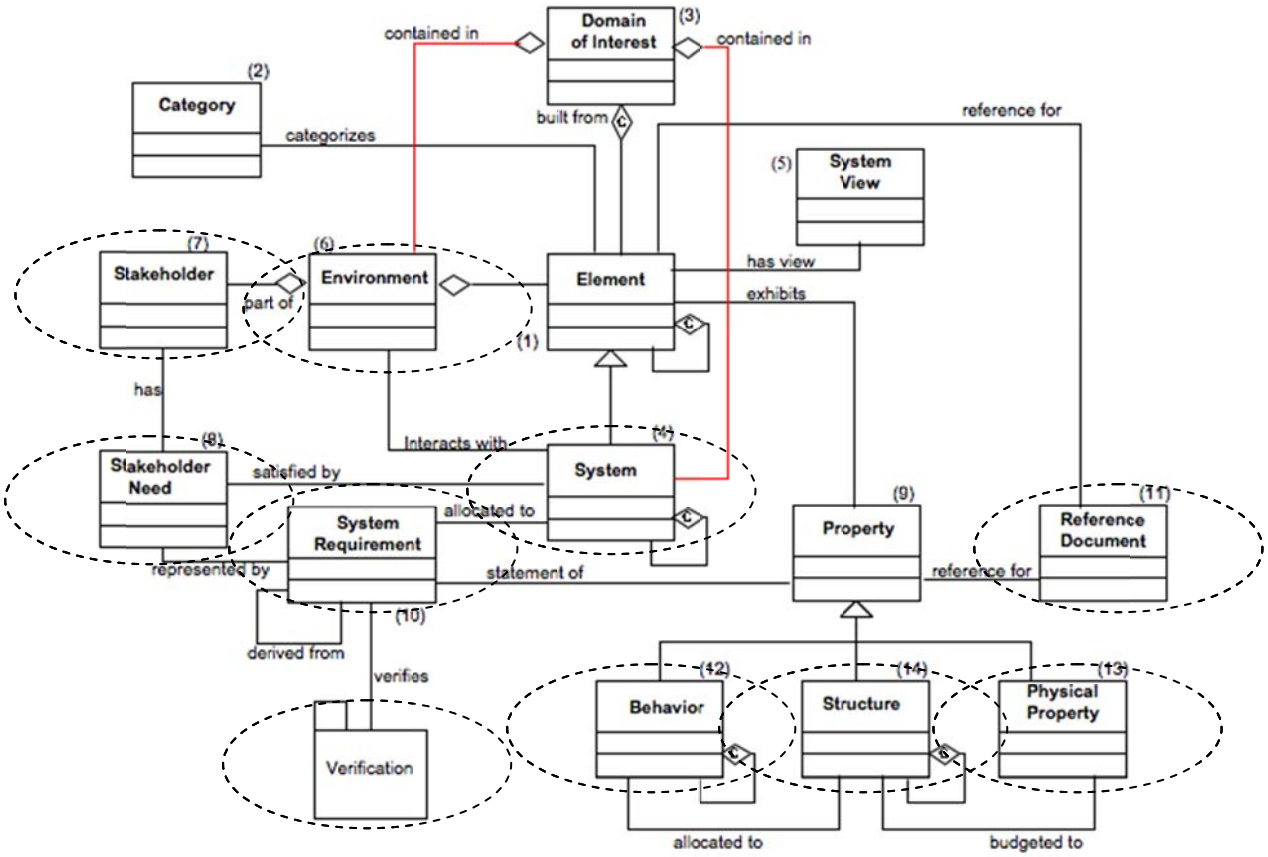


Figure 14. Top-level concept model of AP233 [5]; the dashed ovals point out the modules that are covered by IIDAbase model.

Figure 15 below depicts the *Structure* class of Figure 14 in more detail. It has worked as a reference for the IIDAbase system structure model presented in Section 3.6.

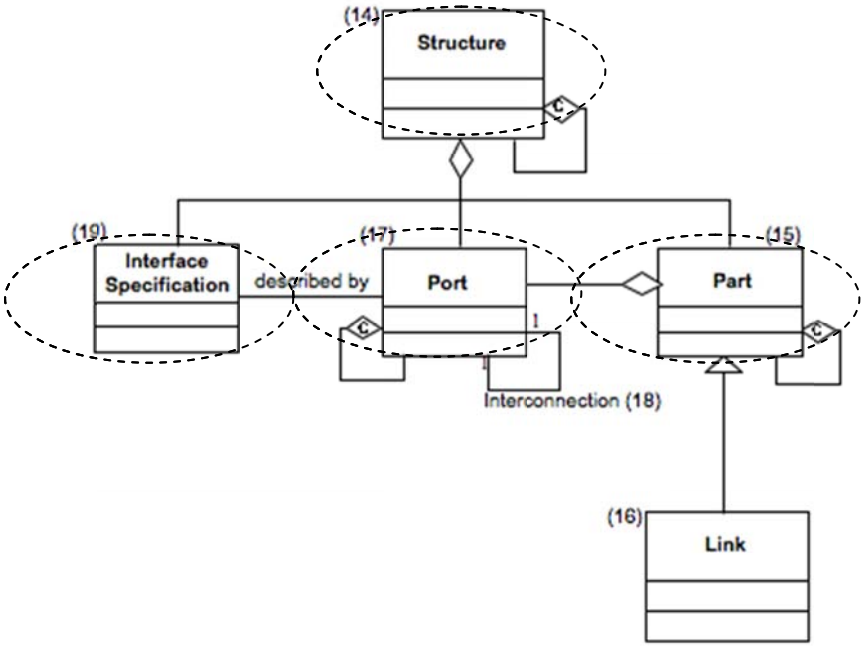


Figure 15. System structure concept model of AP233 [5]; the dashed ovals point out the modules that are covered by IIDAbase model.

The SYSMOD SysML profile presented in [11] has also influenced the model somewhat, especially by providing the concept of *system process*, and it also have worked as an backrest to ensure that the data model does not grow too far from the SysML world to provide adaptation of the data model to accommodate SysML models in the future.

One of the challenges of making the data model was to define the model for the system structure. Several models are available, e.g., the one in IEEE 15288 [12] in which the system is considered to consist of lower level systems until a non-decomposable system is called a system element. There are therefore only two modelling items, system and system element, to describe the system structure. Another model is presented in the INCOSE systems engineering handbook [13]. It defines a very detailed system hierarchy structure with the following hierarchy levels: system – element – subsystem – assembly – subassembly – component – part. This model is too complicated for the IIDAbase model. MSRSYS [7] only defines one modelling item: part type (and its instance: part). Hence, a system in the MSR model is a part (type) that consists of several parts, each of which consists of several parts and so on. Finally, Weilkiens [11] defines the system model such that the system is at the top-level; the system consists of subsystems that in turn consist of blocks that can consist of blocks, etc. The model adopted for IIDAbase is close to the one presented in the IEEE 15288 standard, but instead of calling the non-decomposable element a 'system element', we call it a 'part'. A clear distinction between a part and a system is made in the fact that a part has a spare part number and is hence atomic, but a system (whether a top-level system or a subsystem) consist of several parts and does not have a spare part number itself.

The IIDAbase concept model is presented below in eight parts:

- System
- Requirements
- Risk assessment
- Structure
- Behaviour
- System functions
- Documents
- Network.

These parts will be described in detail starting from Section 3.3.

3.2 Notation for the concept models

UML class diagram notation is used to present the concept model. For those who do not know UML, Figure 16 provides a guide for interpreting the models in this chapter.

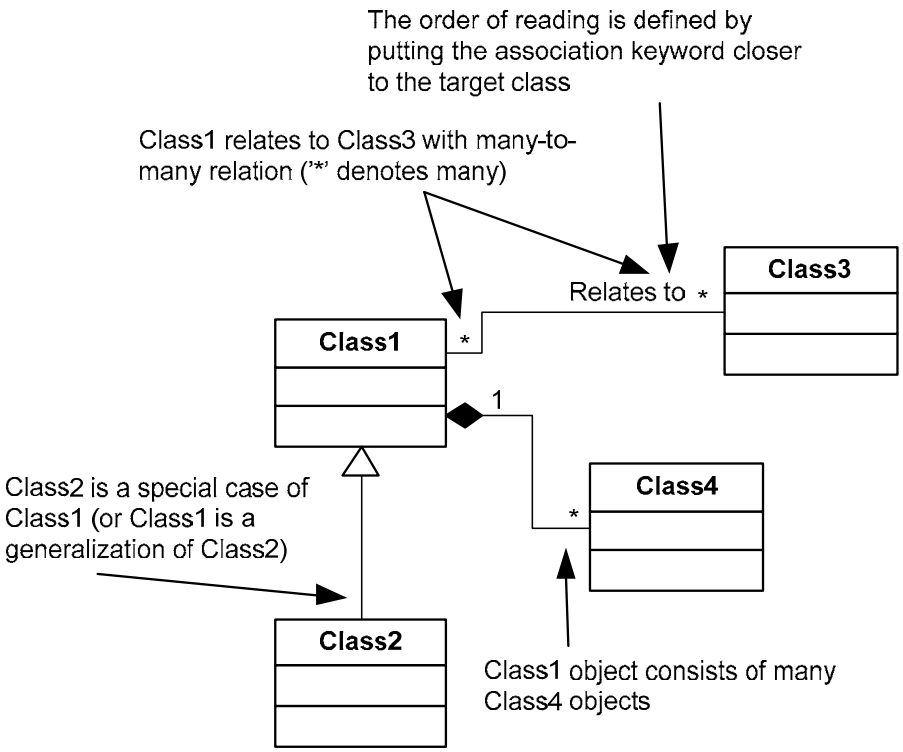


Figure 16. Guide for interpreting the model notation.

The attributes of the classes are not shown in the concept model diagrams; they are defined in the context of the database implementation model that is presented in Section 4.1, but a list of the attributes is not provided in this research note.

3.3 System

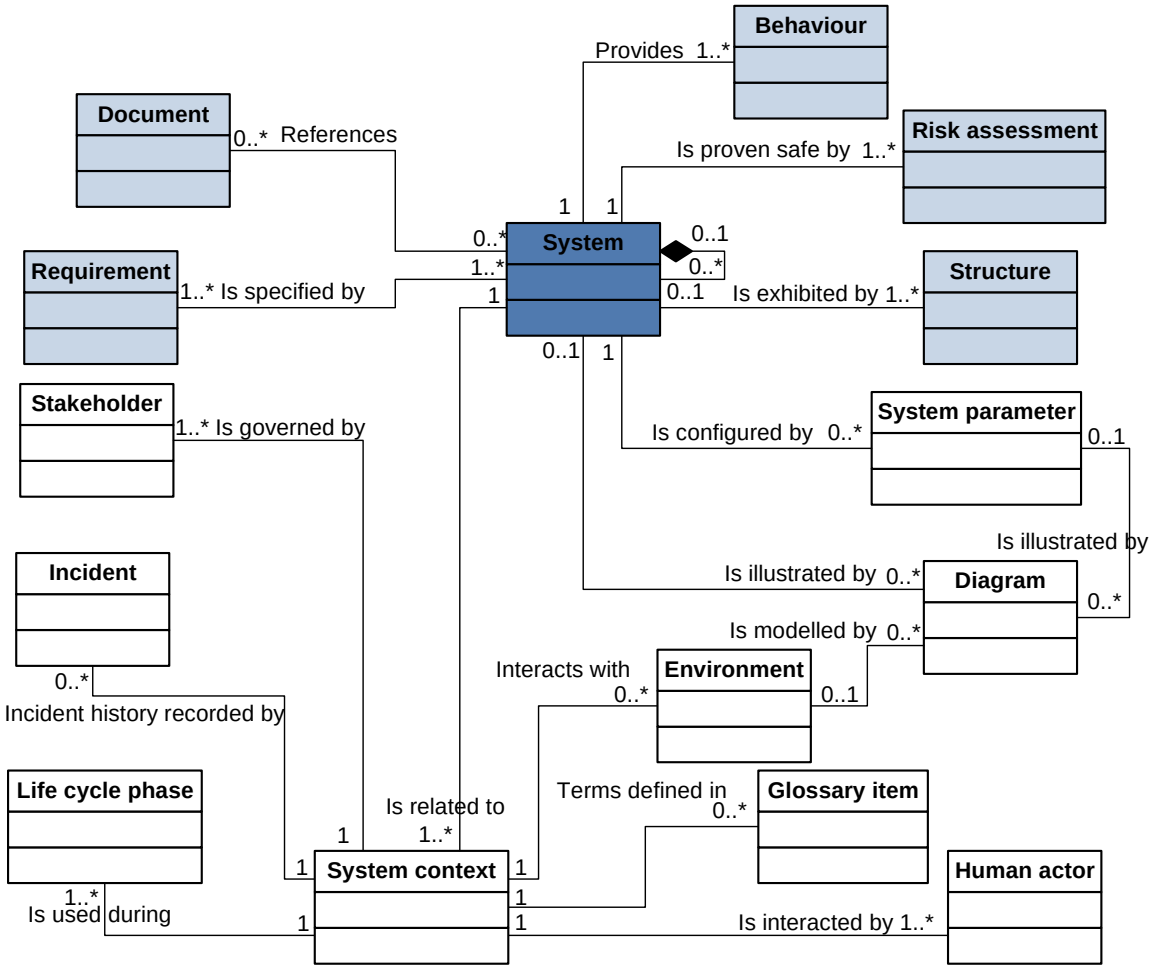


Figure 17. System concept model.

The system concept model in Figure 17 makes the top-level model. The main parts of the system model (besides the *System* class) are the *Requirement*, *Structure*, *Behaviour* and *Risk assessment* classes[4]. A separate model for each is provided in the subsequent chapters. The *Document* and *Diagram* classes are also presented in the document model provided later in this document (Section 3.9).

The rest of the information is more or less dictated by the risk assessment standard ISO 14121-1 [14] in its Section 4.2 and Chapter 5. Most of the issues required by the particular ISO 14121-1 chapters fall under the *System context* class and related classes, but some of the issues are covered by the behaviour model that is described in Section 3.7.

The *System* class only includes a small number of attributes, mainly name and a short description of the system. A system can consist of several subsystems each of which has a structure of its own, but not necessarily a behaviour or risk assessment of its own. The model does not make any distinction

[4] Here and throughout the document a ‘class’ is a synonym for ‘artefact type’.

between a system and a subsystem (except that a system does not have a parent system), however, and hence, e.g., risk assessments of subsystems are supported by the model.

A system may provide several *Behaviours*. This is useful in cases in which the system or the machine clearly has separate ways of working, for example, a forest machine can work both as a forwarder and as a harvester.

The *System context* class holds descriptions about the following issues requested by ISO 14121-1: the machine to which the system belongs, 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. The *System context* class works as the main node to collect, through associations, the system-context-related classes. The *System context* class provides a way (by its one-to-many relation from *System*) of introducing several contexts for a system if the system is going to be used in very different contexts.

The environment in which the machine can work is described in the *Environment* class. An environment can be mechanical, climatic, chemical, ergonomic, external system, domain knowledge or other. The ‘domain knowledge’ concept needs more explanation: The concept of domain knowledge is derived from the SYSMOD profile [11]. It provides a systematic way to model the ‘functionality’ of the context in which the machine is doing its work, e.g., in a mining environment the domain knowledge would include information about the shifts, entering policies into the mining area and the infrastructure of the mine. This information can be modelled systematically, e.g., by the SysML modelling language. The IIDAbase model does not include full-blown support for SysML but provides a facility to link a diagram, like a SysML diagram, to the *Environment* class.

The *Life cycle phase* class lists the life cycles of the machine according to ISO 14121-1. It is basically an enumeration table, but it allows descriptions to be added about the interpretation of the life cycle phases in the particular system context.

The *Incident* class fulfils the ISO 14121-1 requirement for providing more systematic information for the risk assessment about the experience of use. The *Incident* class gathers the accident, incident or malfunction history of the machine or system type or of a similar machine or system type.

The *Human actor* class is what the name implies, a class to identify the human actors dealing with the system. Non-human actors are not listed here but in the *System use cases* class (see Section 3.7). It should be noted that there may be other persons working in the hazard zone of the machine who are unintentionally in touch with the machine and are thus a safety concern. Such persons are also listed in the *Human actor* class.

The *Stakeholder* class is used to list the stakeholders of the system. The stakeholders include persons and organizations with interests in the characteristics of the system. Thus, they form part of the system context. The *Stakeholder* class also relates to requirements; this relation is shown and described in Section 3.4.

The *Glossary* class collects the domain-related terms. A consistent understanding of terms is also relevant to risk assessment.

Note that ISO 14121-1 requires the relevant safety standards, regulations, technical specifications and data sheets to be referenced. This is done by associating the *Document* class with the *System* class.

The attributes of the classes are defined and described in the context of the database model that is discussed in Section 4.1. A full list of attributes is not provided in this research note however.

3.4 Requirements

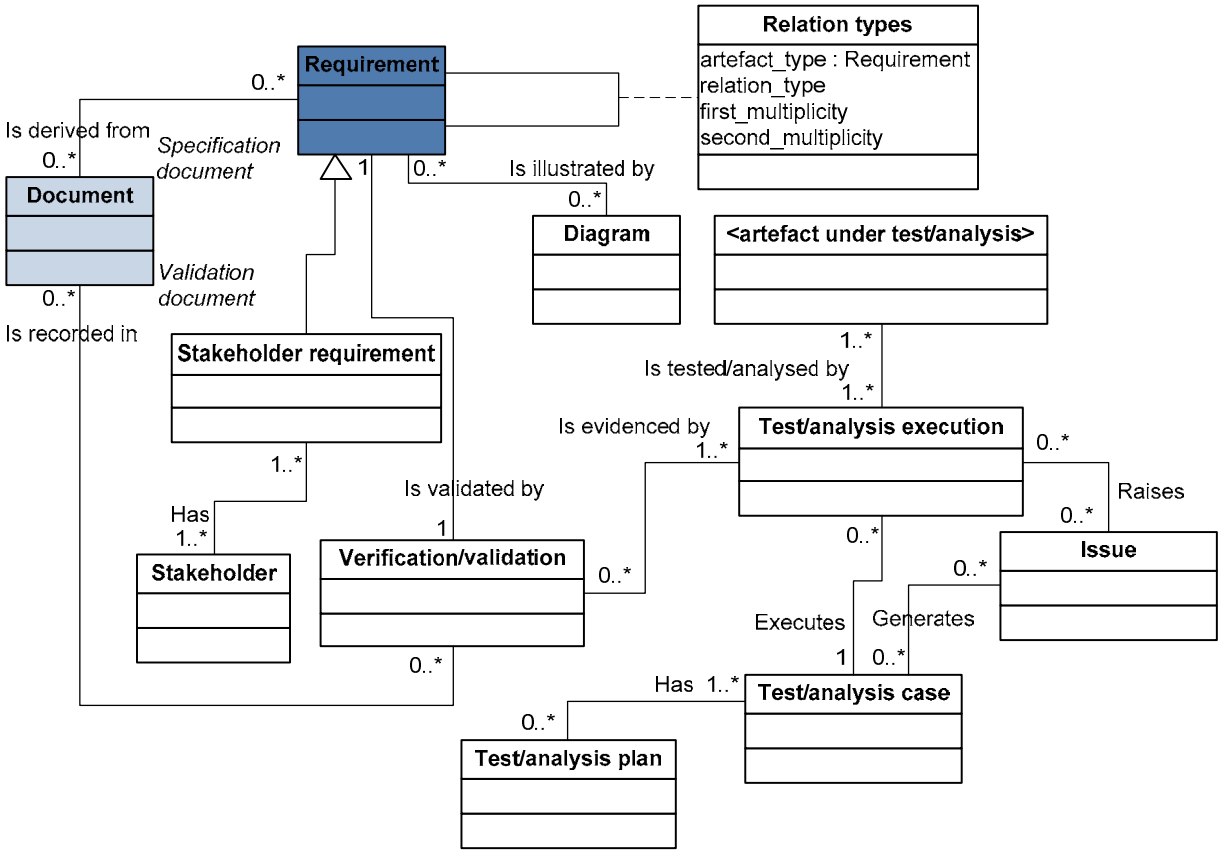


Figure 18. Requirements concept model.

The requirements are managed in a specific requirements management or application lifecycle management tool. The requirements concept model in Figure 18 can normally be implemented with such tools. In some cases, a specific test case management tool may be needed besides the requirements management tool.

A requirement can have child requirements, e.g., to refine the requirement. In this case, the trace role is ‘refine’, but other roles can also be supported by the *Relation types* class, which defines all the relations of the artefacts. *Diagrams* and *Documents* can be linked to the requirements where necessary.

The *Stakeholder requirement* is a special case requirement: it relates to one or more *Stakeholders* and needs to be validated at the end of the project to prove that the system fulfils the expectations of the stakeholders.

The implementation of the requirement is validated at the end of the development project. A special set of artefact items are provided for this purpose: *Verification/validation*, *Test/analysis execution* and *Test/analysis case*. Tests or analyses are needed for evidence of fulfilling the requirement. The intention is for the test and analyses to be managed in an ALM or PLM tool or a test case management tool and for the interfacing to these tools to be done by mirroring the test cases and the test executions from the test management tool to the IIDAbase *Test/analysis execution* and *Test/analysis case* classes re-

spectively. A safety engineer who uses an IIDAbase-connected safety tool links the validation of a requirement to the appropriate *Test/analysis executions* to provide evidence for the validation. A *Test/analysis execution* must be linked to the artefact under test (or analysis) to prompt for a test (or analysis) re-execution if the artefact under test (or analysis) is updated.

If the execution of a test or an analysis fails, an *Issue* can be raised to call for corrective actions. A link to the log file of the particular *Test/analysis execution* can be attached to the issue artefact.

The *Test/analysis plan* class collects a set of *Test/analysis cases* to form a specific sequence of tests for a specific purpose, such as for the Factory Acceptance Test (FAT).

The validation model is designed to work with the ISO 13849-2 [15] validation procedure. For example, when validating the category of a safety function, 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 put into the *Test/analysis case* class, e.g., as: *Inspect that components/systems are properly selected, combined, arranged, assembled and installed*[5]. There are several such tasks, not only the basic and well-tried safety principles, in ISO 13849-2 that can be presented as test or analysis cases. The fault modes to be considered can also be presented as test or analysis cases[6]. The result of the test or analysis is stored in the *Test/analysis execution* class, and the validation result is recorded in the *Verification/validation* class. 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 *Verification/validation* class has a one-to-many relation to the *Test/analysis execution* class.

The attributes of the classes are defined and described in the context of the database model that is discussed in Section 4.1. A full list of attributes is not provided in this research note however.

---

[5] It should be noted that such a basic safety principle is in fact a requirement and could be stored as such in the *Requirement* class, but as the need for the basic and well-tried safety principles depends on the category, which is an intermediate requirement after deciding how the initially required Performance Level will be achieved by the safety-related parts of the control system, it is more natural to keep the basic and safety principles as *Test/analysis cases.* If, on the other hand, we make them requirements, the example requirement would be: *Components/systems must be properly selected, combined, arranged, assembled and installed*, and the corresponding validation would be: *Inspect that components/systems are properly selected, combined, arranged, assembled and installed*, leading to unnecessary redundancy. We could simply write the validation task, however, as: *Inspect documentation to see that this requirement is fulfilled* or simply: *Inspection*. It is recommended here that the basic and well-tried safety principles are presented in the *Test/analysis cases* for the reason that ISO 13849-2 regards the basic and well-tried safety principles as a validation issue, not a requirement issue.

[6] Here again, we may ask why the fault modes are presented as late as in the validation phase and not in the requirements phase to tell the engineers that a particular fault mode must not cause the loss of a safety function. It is suggested here that the fault modes are listed in the *Test/analysis cases* for the reason that ISO 13849-2 regards the fault modes as a validation issue, not a requirement issue.

3.5 Risk assessment

The risk assessment concept model follows the ISO 14121-1 risk assessment process model depicted in Figure 19.

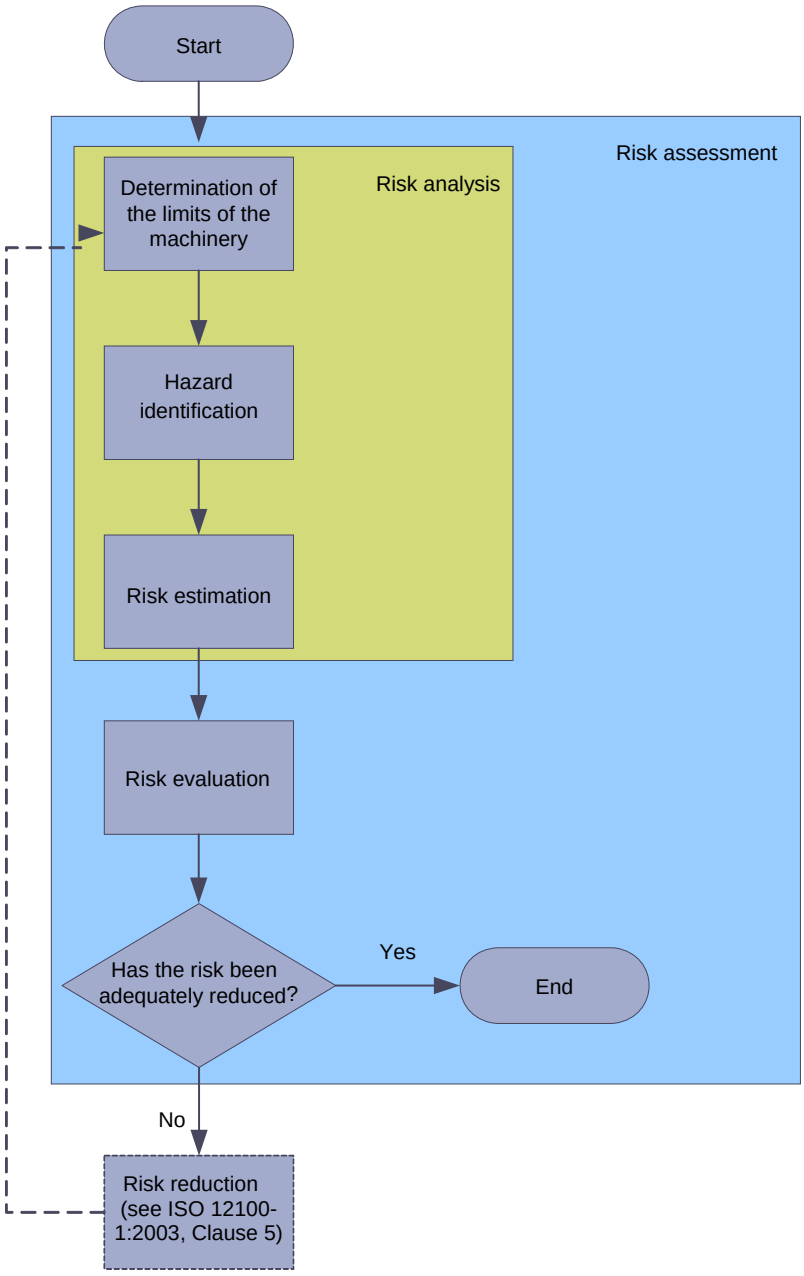


Figure 19. Risk assessment process model according to ISO 14121-1 [14] (Risk analysis = combination of the specification of the limits of the machine, hazard identification and risk estimation; Risk estimation = definition of likely severity of harm and probability of its occurrence; Risk evaluation = judgment, on the basis of risk analysis, of whether the risk reduction objectives have been achieved; definitions from ISO 14121-1).

The artefact concept model derived from the ISO 14121-1 risk assessment process model is provided below (Figure 20).

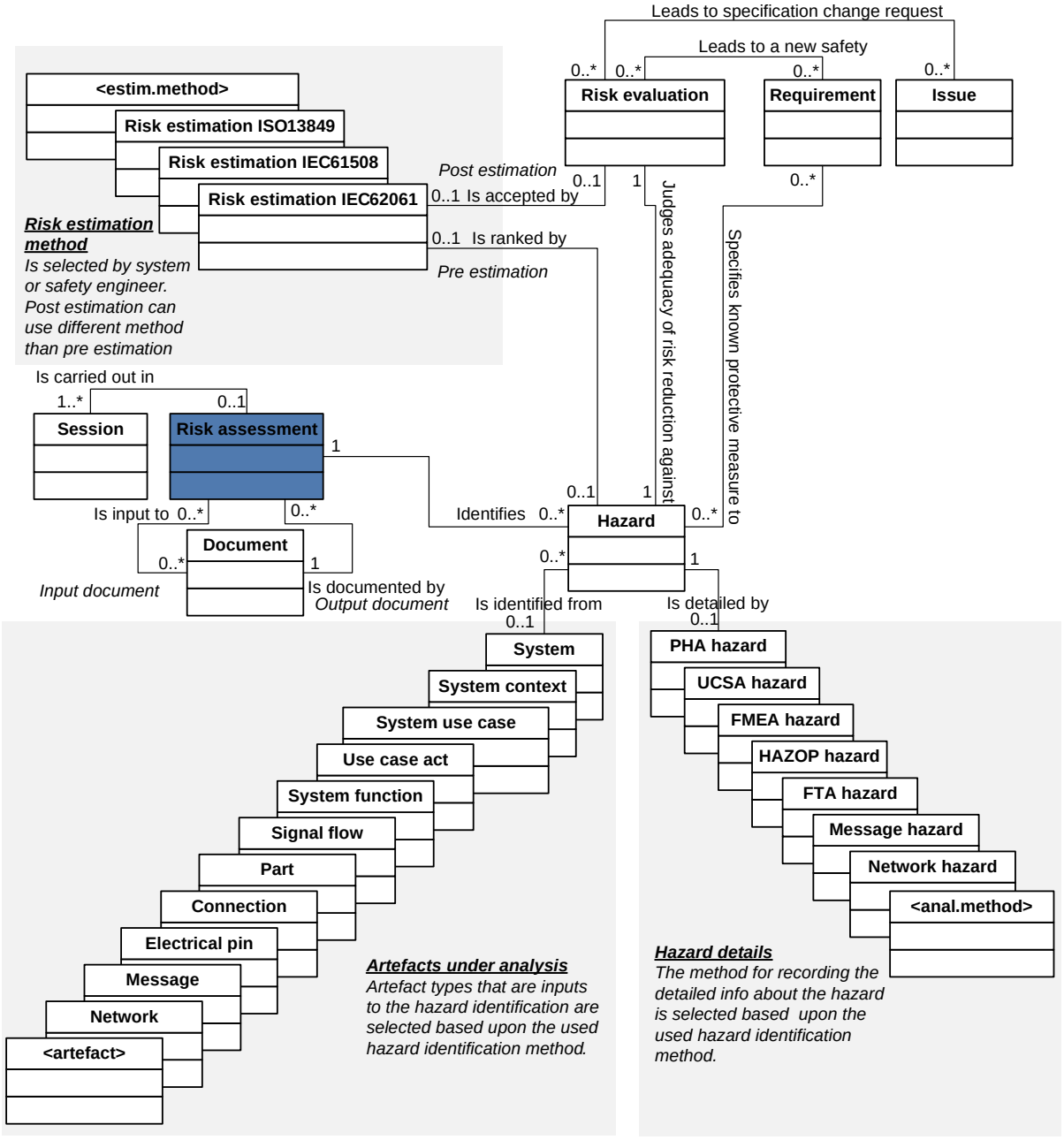


Figure 20. Risk assessment concept model.

The generic information concerning the risk assessment is stored in the *Risk assessment* class. 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[7] (UCSA), Failure Mode and Effects Analysis (FMEA), Fault Tree Analysis (FTA), Hazard and Operability study (HAZOP) and communications analysis (message hazard and network hazards analyses).

A risk assessment is performed in several analysis sessions, the minutes of which are recorded in the *Session* class. During the PHA, UCSA, etc. sessions, *Hazards* are identified based on the analysis type-specific methods. The source information for the analyses types can be:

- *System* (typically for PHA)
- *System context* (typically for PHA)
- *System use case* (typically for PHA and UCSA [the supplemented use case])
- *Use case act* (for UCSA only)
- *System function* (typically for FMEA, FTA or HAZOP)
- *Signal flow* (typically for HAZOP)
- *Part* (typically for FMEA)
- *Connection* (typically for FMEA)
- *Electrical pin* (typically for FMEA, but only for rare cases if any)
- *Message* (for communications analysis only)
- *Network* (for communications analysis only)
- any other artefact[8].

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

- *PHA hazard*
- *UCSA hazard*
- *FMEA hazard*
- *HAZOP hazard*
- *FTA hazard*
- *Message hazard*
- *Network hazard*

---

[7] A special modification by VTT of the better-known Operating Hazard Analysis (OHA). In UCSA, the operator's (and other actors') transactions with the machine are systematically described in system use cases that are detailed by use case acts; for each act in the sequence of acts, a set of hazard guidewords is applied to identify potential hazards. Hence, UCSA is more systematic than typical OHA.

[8] It should be noted that there can be more than one input item to a hazard, e.g., in the case of UCSA, if the list of human actors is changed, the hazards may be affected even though the actual artefact under analysis, the use case act, is not changed. If there is no link from the hazard to the human actors, however, a change in the human actors list may be left unnoticed. Hence, the safety analyst must be careful when linking all the relevant artefacts to *Hazard*, and the analysis tool must support such work being carried out smoothly.

After a hazard has been identified, its risk will be estimated and recorded in the *Risk estimation* class. The model enables several alternatives for the risk estimation method; currently, the risk estimation methods of IEC 61508 [16], IEC 62061 [17] and ISO 13849-1 [18] are supported. The risk estimation method is determined by an attribute in the *Risk assessment* class; this attribute is set by the systems engineer or the safety engineer.

Corrective actions will be recommended if the existing protective measures are not sufficient to reduce the risk. The existing protective measures must be documented in the form of safety requirements and linked to the particular hazard, e.g., during analysis sessions a person may claim that there is an overload limiting device in the system and thus that the risk of an identified hazard is minimal. The analyst may not simply write the claimed protective measure down, he must browse the *Requirement* class (a database or similar storage in practice) and pick up the requirement for the overload limiter there 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 the adequacy of the suggested risk reduction measures and decide on the final implementation of the protective measures against the identified hazard. The judgement is recorded in the *Risk evaluation* class, 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* class includes rationale on the modifications or direct acceptance of the corrective action recommendations. The resulting safety requirements are linked to the risk evaluation to provide a trace to the hazard causing the safety requirement. In the end, the particular safety requirements are validated according to the requirements model in Section 3.4.

There are cases in which the risk evaluation may lead to a change in the original specifications instead of the creation of new safety requirements however, 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 implement and create an *Issue* artefact to change the original specifications, e.g., to strip off features that are difficult to implement cost-effectively with an acceptably low safety risk.

The evaluator team together with the safety engineer can redo the risk estimation[9] to ensure that an acceptable risk level has been reached with the stated new safety requirements.

The communications analysis is performed in two parts: a *Message* analysis and a *Network* analysis. The former is performed according to the model of [19] and the latter according to the network validation questions by the Swedish Pålbus project [20] with VTT modifications.

---

[9] It is possible for the risk estimation method during the risk evaluation to differ from the one used during the risk analysis. The reason is that the protective measures very often affect the occurrence probability of the hazardous event, but the particular probability parameter is not among the probability parameters of the ISO 13849-1 risk graph. Hence no matter how much the probability of a hazardous event can be reduced, the level of risk remains the same before and after risk reduction if the ISO 13849-1 risk graph is used for both risk analysis and risk evaluation.

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.

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

The attributes of the classes are defined and described in the context of the database model that is discussed in Section 4.1. A full list of attributes is not provided in this research note however.

3.5.1 Workflow model

3.5.1.1 Overall model

The overall safety process is compliant with the ISO 14121-1 risk assessment standard and with the ISO 13849-1 standard for safety-related control systems. Its upper level workflow is illustrated in Figure 21. The scope of the model is systems engineering. The mechanics, electrical/electronic hardware and software development process, as well as system integration, are considered to be carried out in the *Implementation* phase of the systems engineering process model in Figure 21.

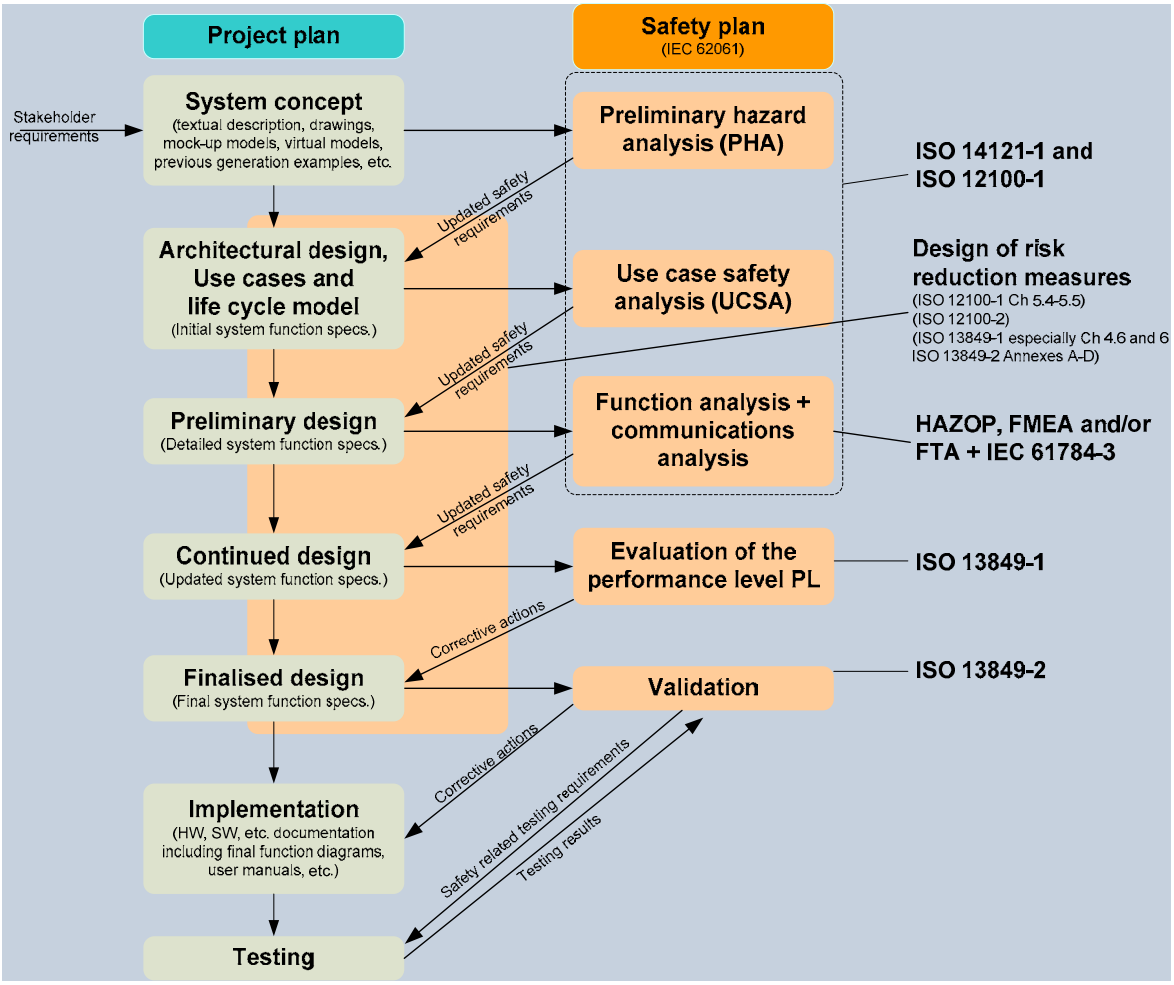


Figure 21. KOTOTU safety process.

The safety process reference model in Figure 21 is called the KOTOTU[10] safety process and is described in more detail in [21].The model defines a three-stage risk assessment model in which risk assessments are carried out for the system concept (Preliminary Hazard Analysis), for the architectural design in which the system use cases are defined platform dependently (Use Case Safety Analysis), and for the system function analysis when the first versions of the system function specifications are ready. The system function analysis can be carried out either by a Hazard and Operability study (HAZOP), a Failure Mode and Effects Analysis (FMEA) or a Fault Tree Analysis (FTA). The selection between these is made case by case by the safety engineer. Besides the system function analysis, a special analysis for the communications system is needed (communications analysis).

The process continues with the Performance Level (PL) evaluation phase according to ISO 13849-1 when the system function designs are considered to be ready after the updates caused by the safety requirements from the risk assessment phases. If the required PL has not been reached, the design is updated accordingly.

Finally, the safety-related control system is validated according to ISO 13849-2.

In the following sections, detailed workflows for risk assessments are defined such that the IIDAbase concept model is used to provide and store the risk–assessment-related artefacts.

3.5.1.2 General risk assessment workflow

The general risk assessment workflow is presented in Figure 22. It uses the classes depicted in Figure 20. The figure is self-explanatory.

[10] The KOTOTU safety process has been defined by VTT in a research project called KOTOTU.

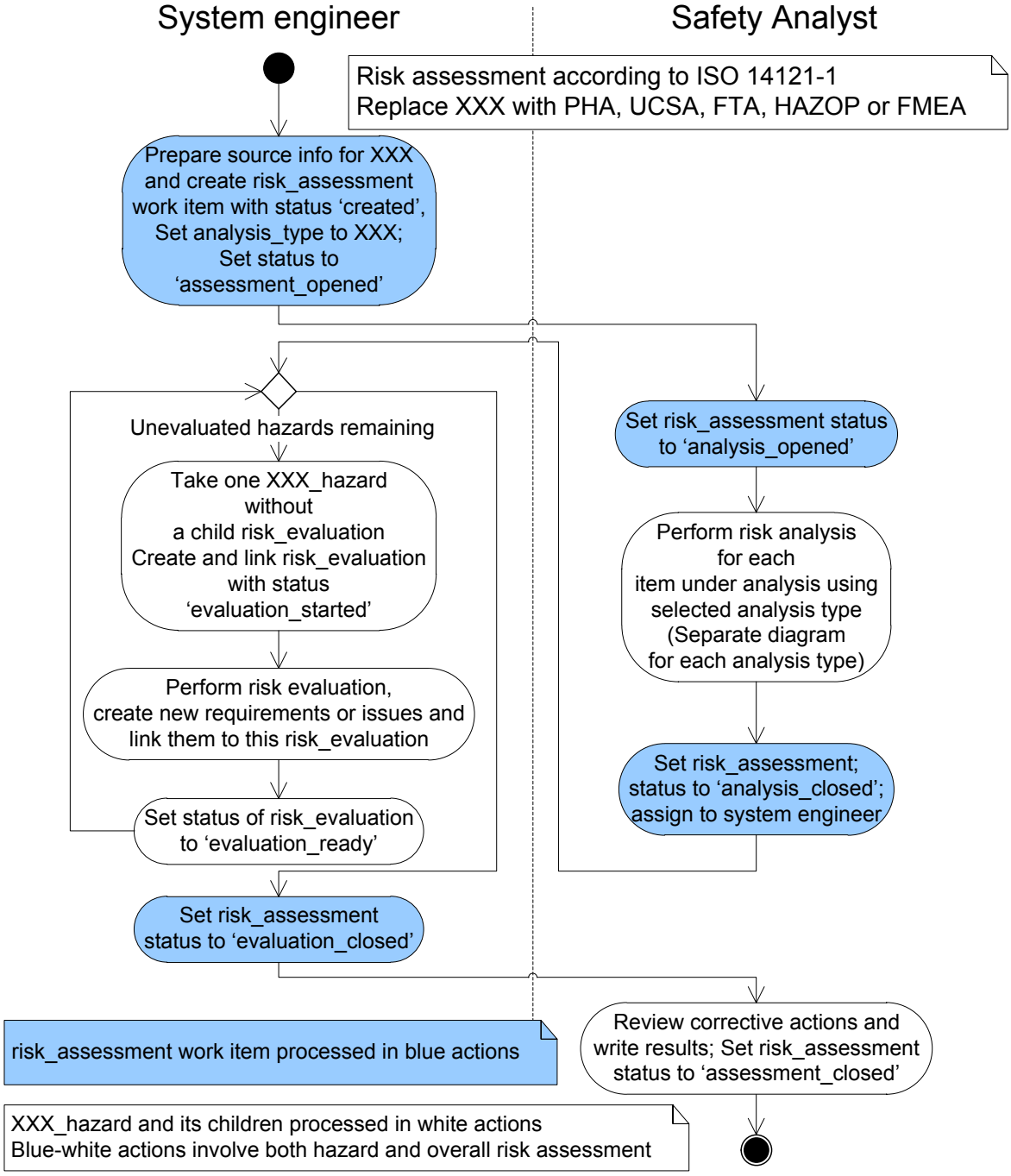


Figure 22. General risk assessment workflow.

The updating of the *Session* class is not depicted in the workflow model. The *Perform risk analysis* oval on the right-hand side is described for PHA, UCSA, HAZOP, FMEA and FTA in more detail in the following sections.

3.5.1.3 PHA workflow

The PHA-specific part of the risk assessment workflow is presented in Figure 23. It uses the classes depicted in Figure 20. The figure is self-explanatory.

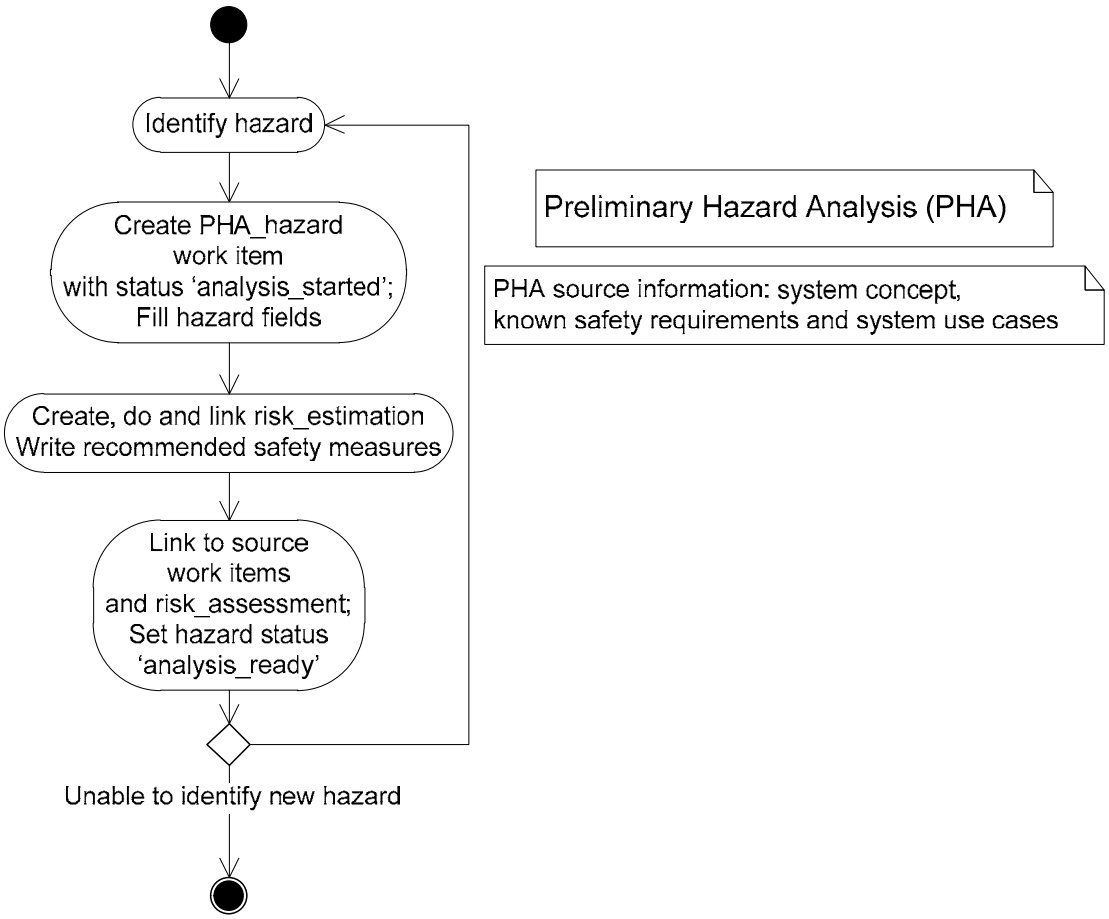


Figure 23. PHA part of the risk assessment workflow.

Note that if the system use cases are input into PHA, only an initial, platform-independent use case description is needed in this concept phase. Hence, the platform-dependent description of the sequence of acts needs not (and must not) be ready in this phase, e.g., it is not wise to fix the type of HMI control devices in the concept phase because the results of the PHA may lead to requirements and constraints on the selection of control devices.

3.5.1.4 UCSA workflow

After the control devices are selected, i.e., the architecture of the controlling platform has been designed, it is time to start UCSA with the completed use case specifications that include the detailed descriptions of the sequence of the acts. The acts shall be described in a platform-dependent way be-

cause the results of UCSA are poor if only abstract descriptions of the HMI devices are provided; UCSA is only powerful if the analyst is able to analyse the possible mishaps caused by the use of the particular type of control device.

The UCSA-specific part of the risk assessment workflow is presented in Figure 24. It uses the classes depicted in Figure 20. The figure is self-explanatory.

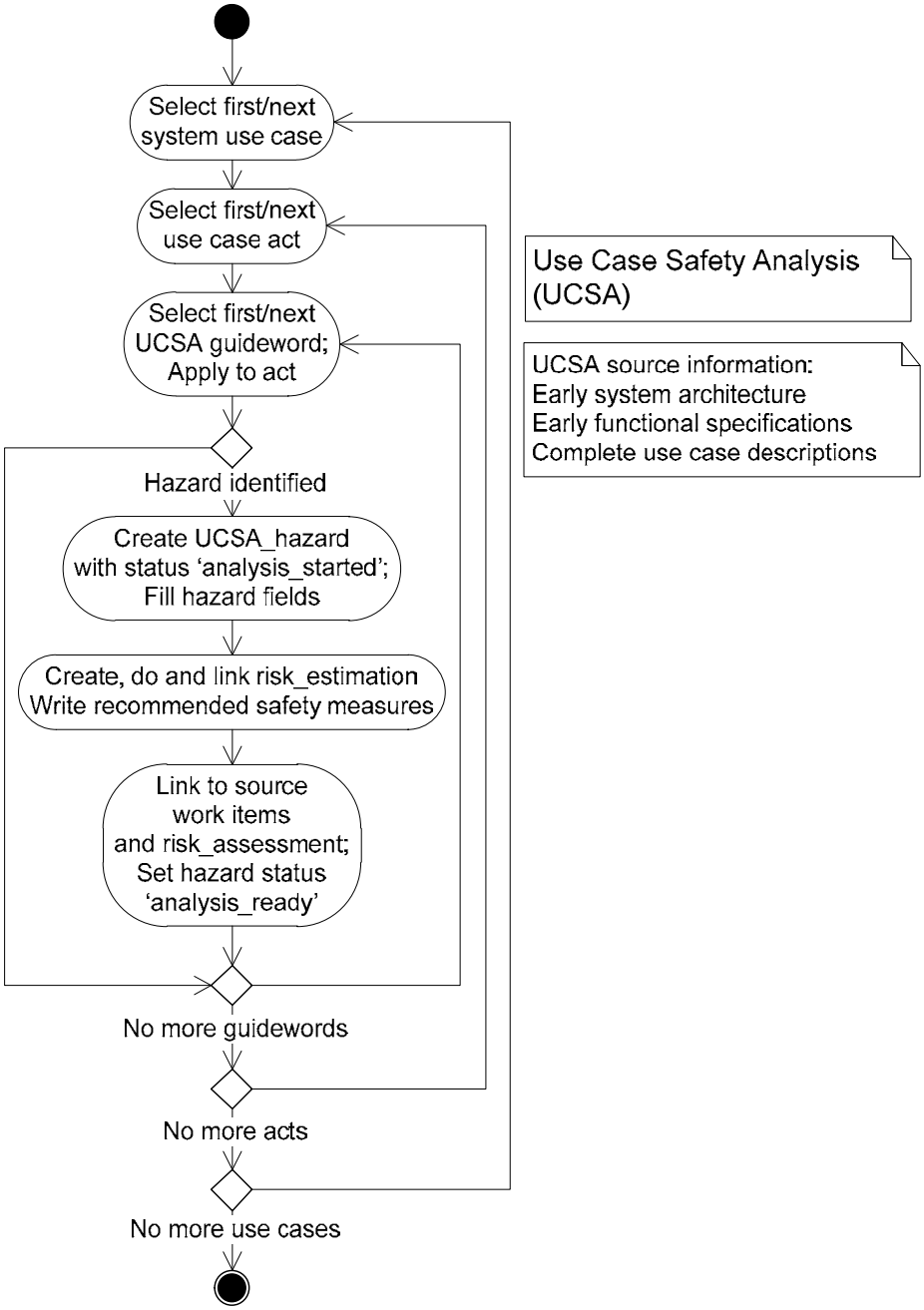


Figure 24. UCSA part of the risk assessment workflow.

3.5.1.5 HAZOP workflow

The HAZOP-specific part of the risk assessment workflow is presented in Figure 25. It uses the classes depicted in Figure 20. The figure is self-explanatory.

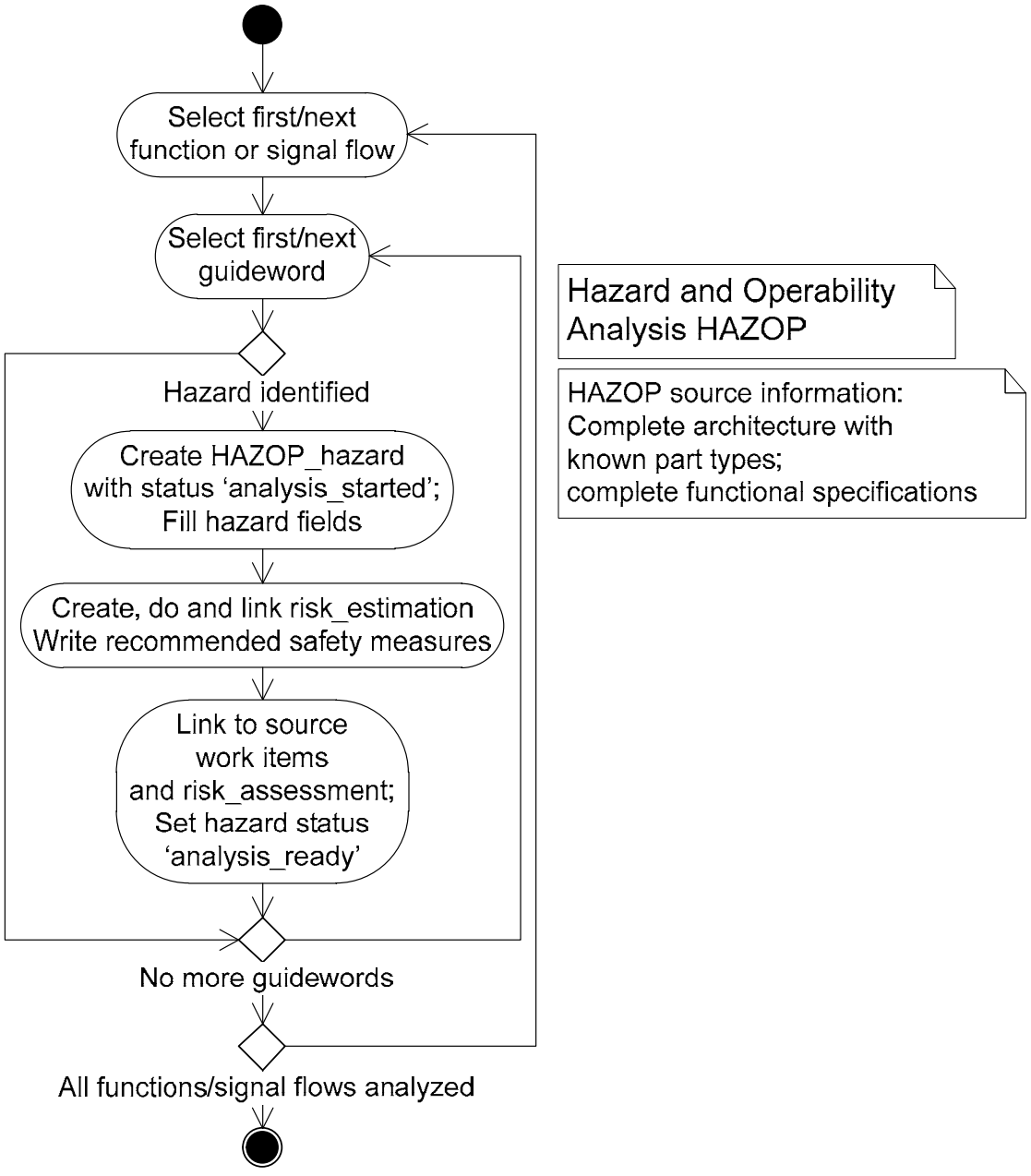


Figure 25. HAZOP part of the risk assessment workflow.

3.5.1.6 FMEA workflow

The FMEA-specific part of the risk assessment workflow is presented in Figure 26. It uses the classes depicted in Figure 20. The figure is self-explanatory.

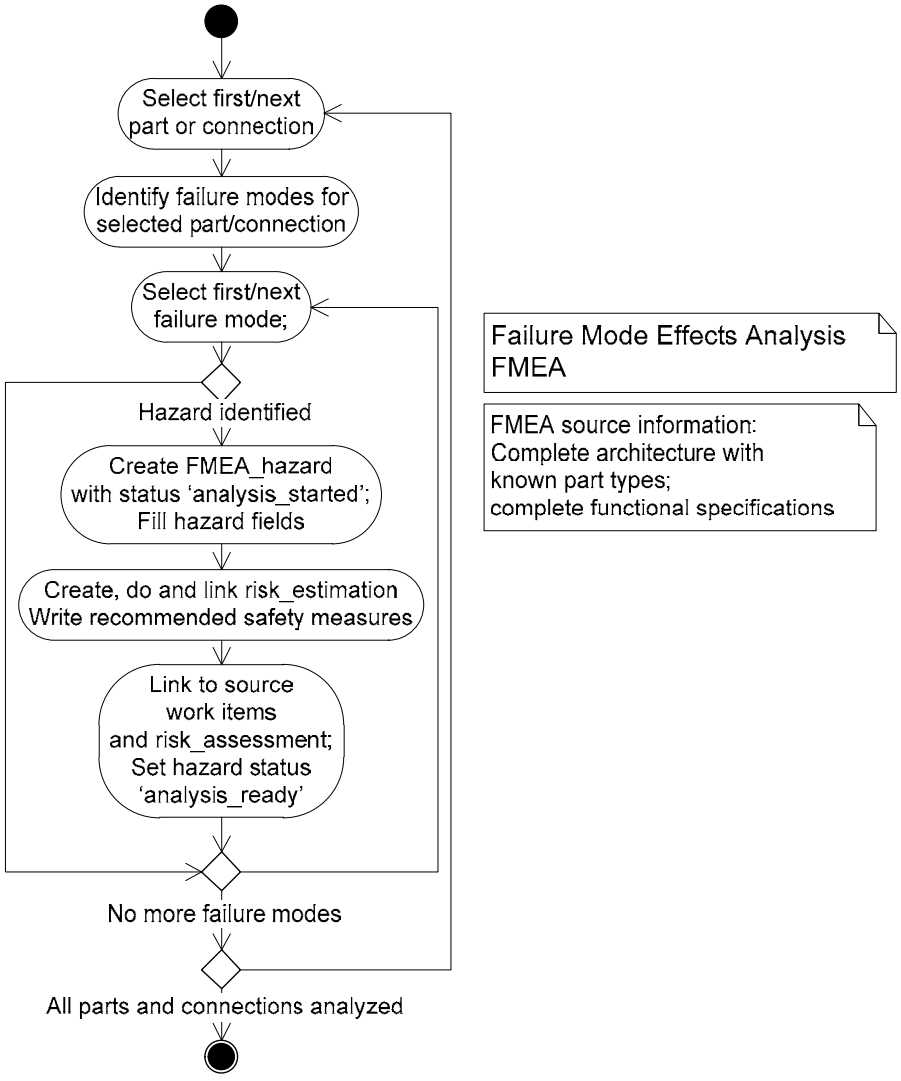


Figure 26. FMEA part of the risk assessment workflow.

3.5.1.7 FTA workflow

The FTA-specific part of the risk assessment workflow is presented in Figure 27. It uses the classes depicted in Figure 20. The figure is self-explanatory.

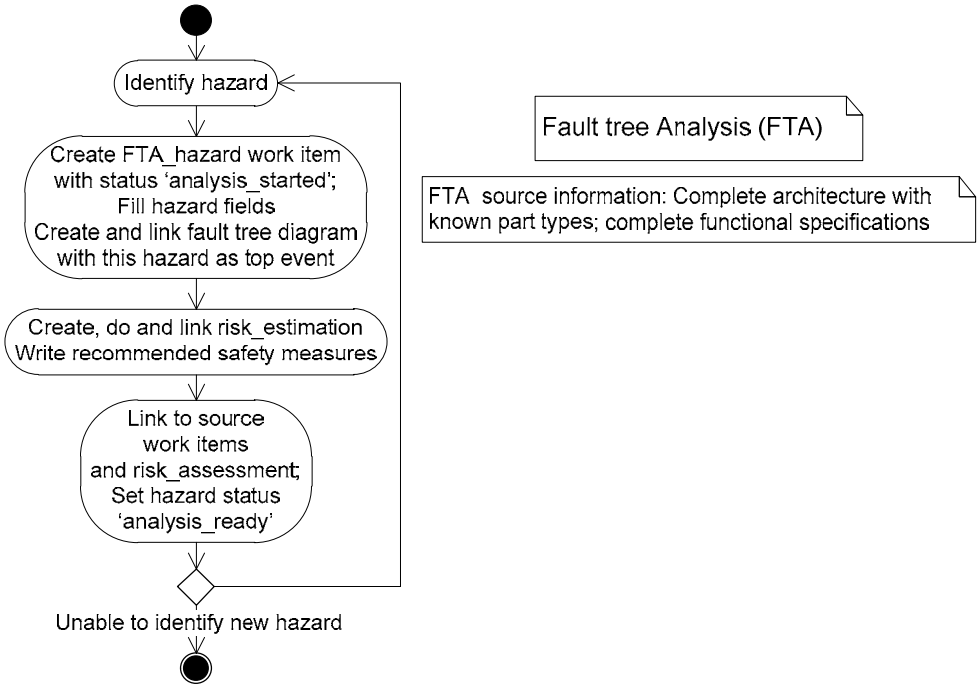


Figure 27. FTA part of the risk assessment workflow.

3.5.1.8 Communications analysis workflow

Network modelling was not part of this work, but the following workflow presented by VTT [22] for the analysis of the CAN communications system can be adapted to create a workflow model for a general case:

The procedure for assessing the capability of the designed CAN system for defending against the communication error types is a ten-step procedure as follows:

1. Initially, a list of all messages is prepared (i.e., the message specification is compiled); all messages are marked safety critical by default.
2. Each signal (i.e., a network variable) of a message is assessed to find out whether any of the message error types listed in IEC 61784-3 (plus the inconsistency error and the unacceptable jitter error) may have a safety-critical consequence for the particular signal.
3. Messages that carry only non-safety-critical signals are marked as non-safety-critical messages. Others remain safety-critical.
4. For a message that is marked safety-critical, the relevant error types for the message are recorded.
5. The number of safety-critical consumers (i.e., receivers) for the particular message is recorded; it does not matter whether the safety-critical consumers read different or the same signals in the message. (The inconsistency threat is not considered relevant if none of the safety-critical consumers reads the same signals however.)

6. The effective message rate value is recorded if a corruption error is considered relevant to the particular message.
7. The deletion condition is recorded for the messages for which ‘loss’ is recorded to be a relevant error type. The deletion condition can be recorded either as a number of deleted messages or as the minimum time between two successive messages that has critical consequences.
8. The allowed maximum delay (latency requirement) is recorded precisely for the messages that are considered vulnerable to the delay error; for the other messages, the message period is recorded as the latency requirement value if a more precise latency requirement is not known.
9. Protective measures that are already designed against message errors are recorded (i.e., the corresponding existing safety requirements for the communications system are linked), and new necessary and optional protective measures are designed and suggested as corrective actions. IEC 61784-3 gives hints on typical protective measures.

IEC 61784-3 does not deal with network faults that fall within the scope of network management. Network faults include, e.g., missing communications participants in the case of distributed control systems. Such faults are typically relevant to machine control systems and must be analysed separately. Hence, the tenth step is defined as follows:

10. The safety characteristics of network management are assessed with the help of check-list questions derived mainly from the check list defined by the Swedish Pålbus project. The check-list questions are stored in the *Network validation question* class. This step also produces suggestions for corrective actions.

Note that if a safety-certified communications system is used, the communications analysis as such is unnecessary. In this case, the validation of the safety of communications is only performed in the validation phase to check that the specifications of the particular communications system are followed.

3.6 Structure

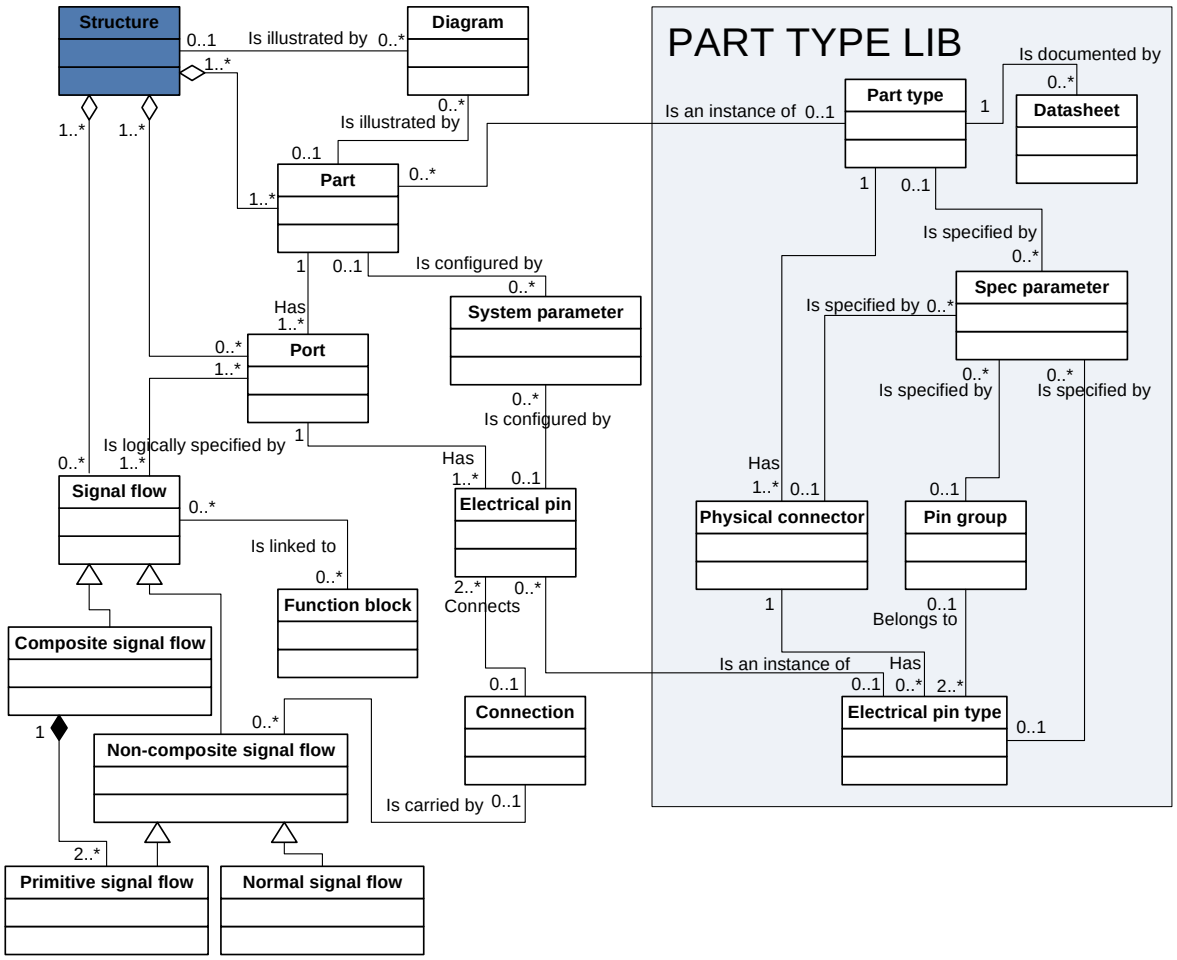


Figure 28. Structure concept model.

The concept model of the system structure (see Figure 28), i.e., the system hardware architecture, is based on the AP233 system structure concept model presented in Figure 15 in Section 3.1. The main idea is that *Structure* is defined by its *Parts* and interfaces. The interfaces are modelled by *Ports* specified by *Signal flows* that flow through the ports. The *Port* has one or more *Electrical pins* (mechanical, hydraulic, pneumatic or optical ports are not currently included in the model). Electrical pins can be connected together with *Connection*. A connection is a logical issue that may manifest itself in the real cable harness in very many ways, not only as a single wire but with cable splices etc. that are defined during the electrical CAD work. Nevertheless, *Connection* represents a galvanic connection between the pins that it connects. As a comparison with the electronics schematics and printed circuit design, the *Connection* class represents a line in electronics schematics, and the electrical CAD work represents a printed circuit design. The internal issues of electrical CAD are not modelled in the IIDAbase model but are left for the electrical CAD tools. It is assured that the model includes the attributes

needed for electrical CAD integration, however. For example, the location information shown in electrical CAD drawings for cables etc. can be part of the 'code' attribute of a *Part* artefact.

In principle, the association from *Structure* to *Port* seems unnecessary because *Structure* consists of all the *Port* artefacts under the *Part* artefacts anyway. In the model, *Port* can now be directly associated with *Structure* if necessary[11], however, even such that a *Port* artefact is not linked to any *Part* artefact. It should be noted that the structure model of AP233 in Figure 7 in Section 2.1 has the same model in this respect: a structure consists of parts and ports, and a part consists of ports. The AP233 standard goes even further: a structure also consists of interface specifications, which in the IIDAbase model are called *Signal flows*. Our model also provides such an association, and this is in fact useful in the case of system functions (see Section 3.8). A system function can be associated with a structure of its own such that, e.g., the safety analyst or a maintenance person can be provided with structure information that points out the parts, ports and signal flows that are used and needed by the particular system function. Without the direct relation from *Structure* to *Port* and *Signal flow*, a system function structure would be impossible to present because a part inherits all its ports and a port inherits all its electrical pins, not only the ports and pins that are relevant to the particular system function structure. Such a system-function-specific structure is a partial structure of the whole system structure and is required by the safety analyst. It is also helpful for the maintenance persons to see which parts, ports and signal flows need to be faultless for the function to work correctly.

For this reason, the model diagram in Figure 28 illustrates the fact that a part, port or signal flow can belong to several structures, to the system structure and to one or many system function structures. Physically they only belong to the system structure however; the system function structure is only a partial view of the system structure.

The *Structure* itself and the *Part* can be illustrated in one or more *Diagrams*. The functionality of a *Part* artefact and of each *Electrical pin* artefact may be configured by one or more *System parameters*. *Ports* are not considered to need configurable system parameters.

There are basically three types of *Signal flows*: *Normal signal flow*, *Primitive signal flow* and *Composite signal flow*. The concept of a composite signal flow is needed in cases in which the actual signal flow is composed of two or more signal flows, which we call primitive signal flows here. Such an example is a quadrature encoder sensor in which the *position Signal flow* is composed of two primitive signal flows, *Channel A* and *Channel B*. In an IEC 61131-3 PLC-programming tool such a composed signal may be generated by a function block from the two input variables, here *Channel A* and *Channel B*. Hence, the model allows a composite signal flow to be related to a *Function block* artefact to provide an unbroken view of the signal flow from its source to the point where it is finally consumed. This helps the safety engineer define clear safety analysis borders. It also provides the possibility of deepening the HAZOP studies to cover the software. It is possible to relate a normal or primitive signal flow to a *Function block* if necessary. In this case, the signal flow will not become the output from, but the input to, the *Function block*. The normal and primitive signal flows can be distinguished in a database implementation by the fact that the foreign key to the composed signal flow is NULL in the

[11] It could be considered that in the case of a system structure, the *Port* and *Signal flow* artefacts that are related directly to a *Structure* artefact could be the ones that constitute the external interface of the system.

case of normal signal flows. At the time of writing this research note, however, no usage scenario is known in which normal signals should be distinguished from primitive signals.

Signal flow is mapped to *Connection* for the purpose of safety analysis: 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 connection between electrical pin x of part X and pin y of part Y. If, however, a more detailed estimation of the probability of the connection break is needed for the safety analysis, electrical CAD drawings will be needed or the information on the cabling implementation will have to be brought by the persons attending the analysis sessions. In the case that the cabling has not yet been designed, the analysis may provide requirements for the structure and quality of the cabling. It is of course suggested that the safety analyses of system functions are carried out before electrical CAD work.

As can be seen from Figure 20, the right part of the concept model is enclosed in a frame called *Part type lib*. The idea is that the datasheet information of the part types is stored in the database. It is also possible to attach a conventional *Datasheet* with a *Part type* artefact if necessary however. The optimal workflow would of course be such that the component manufacturers provide the datasheets in XML files that can easily be incorporated into IIDAbase.

The part type library contains all the generic information about the parts, their connectors and their electrical pins. *Part* is an instance of *Part type* and *Electrical pin* is an instance of *Electrical pin type*. Hence, a part inherits all the information from its part type. Similarly, an electrical pin inherits all the information from its electrical pin type. For electrical pins it is quite normal to provide several functionalities, like analogue input and digital input, and this can be configured according to the application needs. Hence, in this case, the electrical pin cannot simply inherit the I/O type of the pin (because it is configurable). To denote the actual I/O type, two special attributes, *actual_io_type* and *direction*[12], are included in the *Electrical pin* class.

Part type, its *Physical connectors* and their *Electrical pins* can be specified with several *Spec parameters*. The specification parameters cannot be changed; they have been defined by the part vendor, e.g., a part can have as its specification parameter, weight, dimensions, allowable temperature range, etc. Such information is normally presented in an easily readable format in conventional datasheets, but the provision of such a structured way of storing specification parameters in *Spec parameters* facilitates showing of the parameters in documents or drawings created from IIDAbase.

There is also a class for pin groups (the *Pin group* class). The reason for this is that in certain cases it is not reasonable to assign a specification parameter to an electrical pin but to a group of pins, e.g., in the case of a CAN interface, it is reasonable to assign the physical ratings of the interface to a pin group called *CAN interface* instead of replicating the information for each pin, *CAN_H*, *CAN_L*, *CAN_Supply+*, etc.

Connector groups are not modelled as a separate class, but if there is a need to assign connectors to a group, the *Physical connector* class has an attribute for this purpose.

---

[12] The two attributes are redundant in the sense that direction can be derived from the *actual_io_type* if the list of actual I/O types is fixed and well selected.

The *Spec parameter* class has been motivated by the MSRSYS specification [7] and contains many of the attributes defined by it. The *Part type* class has been defined such that it can accommodate the *DeviceIdentity* element from the CANopen XML-based device profile according to CiA DSP 311 [10].

The part type library artefacts (*Part type*, *Physical connector*, *Electrical pin* and *Pin group*) cannot link to a *Diagram* artefact, but they can be attached to figures through the *Spec parameter* class, as a specification parameter can include one figure. Hence, if, for example, a part-type vendor wants to illustrate its part type with photos from the front and behind the device, it can simply create two specification parameters called Front view and Rear view and attach a figure to each of the parameters. The problem with this scheme is that in the case of a database implementation, the database will become large if there are many figures. This is due to the fact that currently a figure is stored in a database as a large binary object, not as a link to the figure file.

The attributes of the classes are defined and described in the context of the database model that is discussed in Section 4.1. A full list of attributes is not provided in this research note however.

3.7 Behaviour

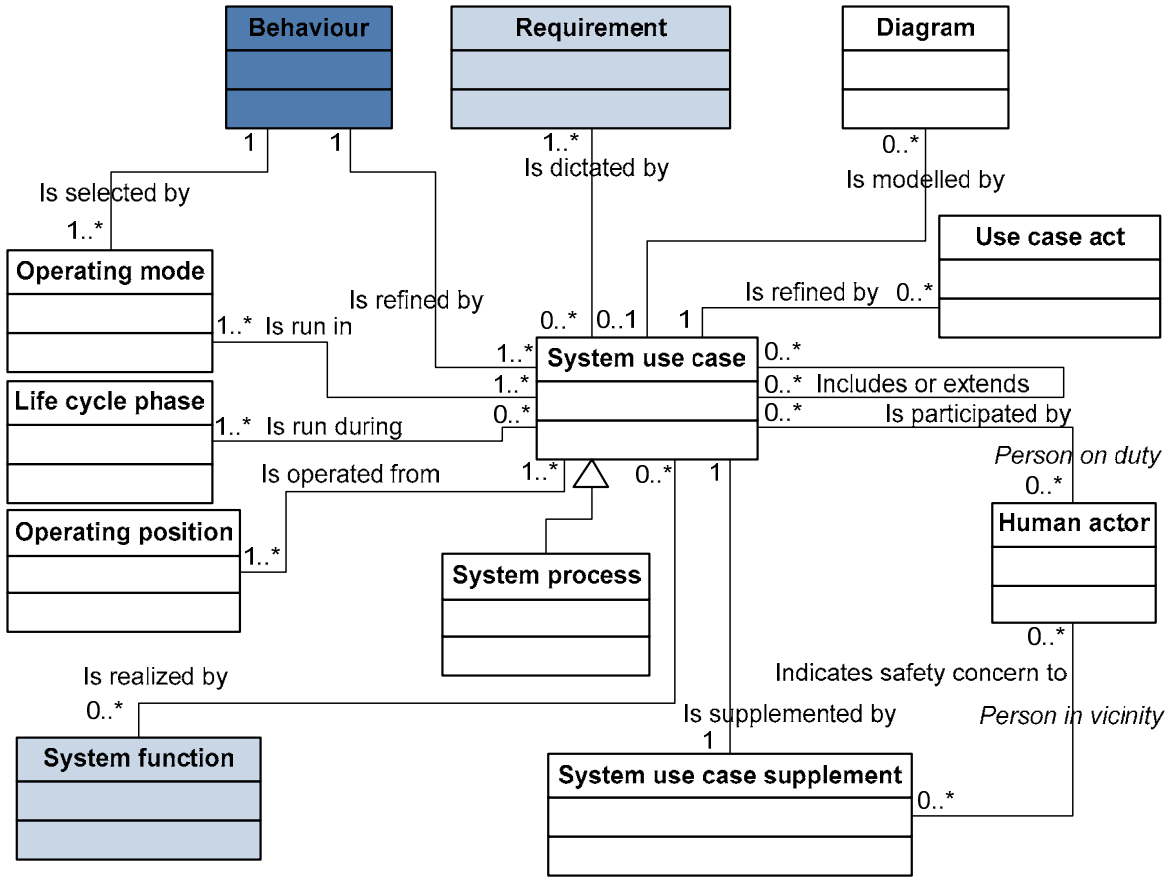


Figure 29. Behaviour concept model.

The behaviour concept model in Figure 29 encompasses the description of the functionality of the system. Its core class is the *System use case* class. The *Behaviour* class only gives a basic description of the system 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 machine as phrased by ISO 14121-1), but it also stores the description of the reasonably foreseeable misuse that must be considered according to ISO 14121-1. The *Operating mode* class is also provided due to the ISO 14121-1 requirements, though they must of course be systematically listed anyway. Life cycle phases and operating positions are listed and described systematically in the *Life cycle phase* and *Operating position* classes respectively.

Behaviour is described in a more systematic way in special use cases called *System processes*. This modelling paradigm is derived from the SYSMOD profile [11]. *System process* is a special case of *System use case*. As described in [11], the system use cases contain the system process information implicitly through the preconditions-post-conditions chain, while the system process contains the whole story in a single use case explicitly. The system process can be illustrated by an activity diagram in which each activity in the diagram is described by a system use case.

System use case artefacts are created to describe the functional requirements stated by the stakeholders in a systematic way. This is why the system use cases are related to the *Requirement* class as depicted in Figure 29.

System use case can include finer grained use cases or extend another system use case. The sequence of acts of a system use case is stored in the *Use case act* class. The reason for separating the *Use case act* class from the *System use case* class is that during the Use Case Safety Analysis (UCSA) we need to be able to link a single use case act to an identified hazard to provide traceability. It must be ensured that to extend traceability such that if, e.g., the set of *Human actor* artefacts is changed not only the *System use case* artefacts related to the changed *Human actor* artefacts are marked suspect but also the related *Use case act* artefacts and the related hazards. This is reasonable if we think about a case in which a new actor is introduced into the system. It is then highly relevant to re-analyse, using the UCSA method, all the system use cases to which the new actor is linked or, if one of the human actors is removed from the human actor list, one or more hazards identified by UCSA may become irrelevant or need an update.

UCSA requires a more rigorous specification of system use cases. Hence, the *System use case* class is amplified by a supplement: the *System use case supplement* class. The reason for separating *System use case supplement* from *System use case* is workflow: the supplement part may be filled in a different phase of the project. The system use case objects could not be marked as ready while the supplement part was waiting for its contents if the system use case table also included the supplement part.

System use cases are realized by *System functions*. A use case can use one or more system functions, and a system function can belong to several use cases. The system function model is described in more detail in Section 3.8.

The attributes of the classes are defined and described in the context of the database model discussed in Section 4.1. A full list of attributes is not provided in this research note however.

3.8 System functions

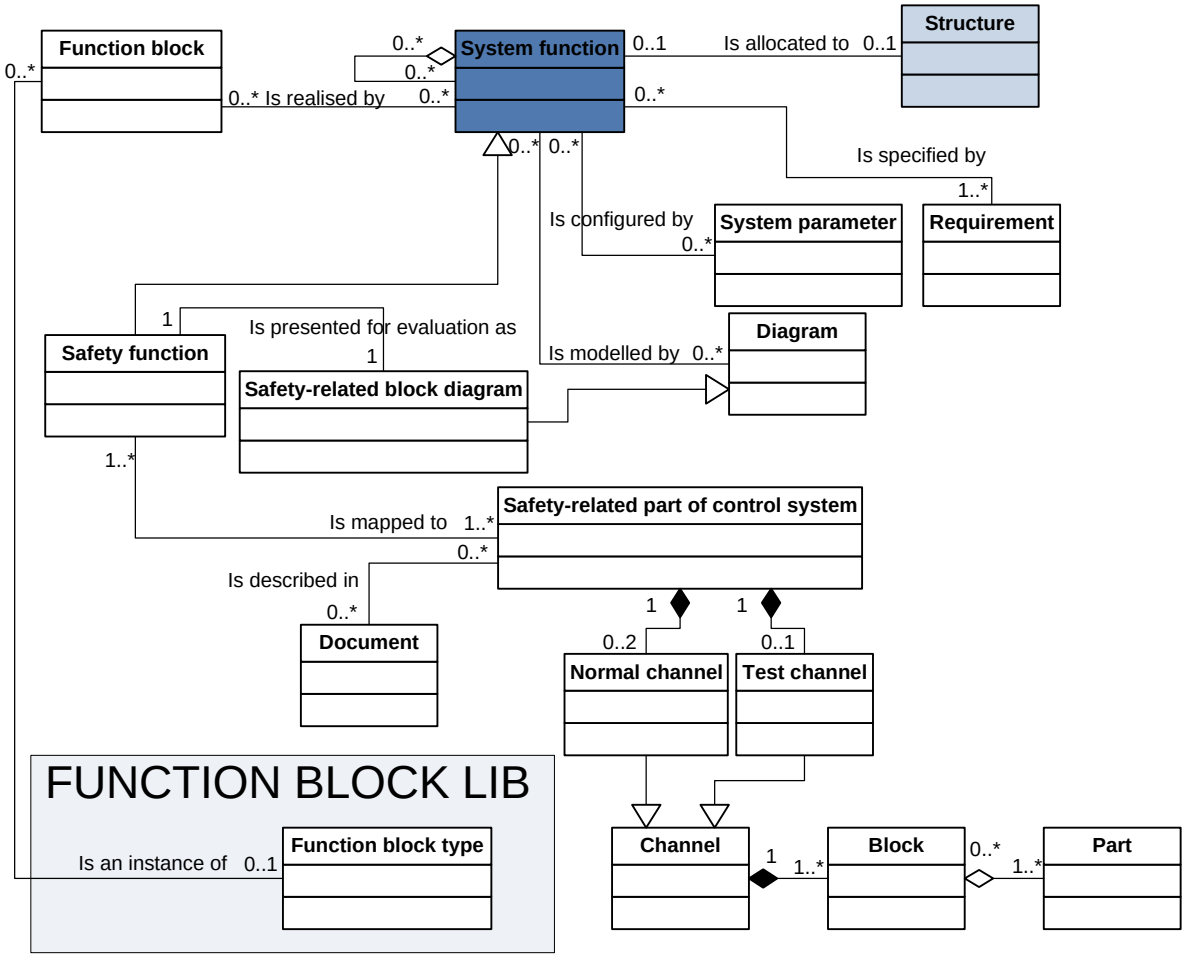


Figure 30. System function concept model.

The system use cases are realized by *System functions*. *System function* is specified exhaustively in this concept model (see Figure 30) such that the software engineer can implement the software for the system function based on it. The attributes of the *System function* class are selected such that the requirements for a safety function specification according to IEC 62061 [17] are fulfilled. *Safety function*[13] is a special case of *System function* and is often a 'sub-function'[14] or, to be more precise, a 'para-

[13] The concept of a safety function can be somewhat obscure to a machine automation 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, a dead-man's switch and a hydraulic enable valve are added. Now the requirement for the safety function 'prevention of unexpected movement' could be, e.g., PL d, which is achieved by a two-channel approach (i.e., with Category 3 according to ISO 13849-1). What are the two channels? The

sitic function' of a system function. Hence, the model allows a system function to consist of one or more sub-functions. There can of course be sub-functions that are not safety functions.

Furthermore, the model contains almost all the information needed to make a safety analysis for a system function with analysis methods like FMEA, HAZOP or FTA. The missing part in this model in regard to safety analysis is the communications part, which is described in a model of its own (see Section 3.10).

A system function is allocated to a *Structure* artefact of its own. As described in Section 3.6, such a system-function-specific structure is a partial view of the actual system structure; see the rationale for this in Section 3.6.

A system function is specified by a set of *Requirement* artefacts configured by *System parameters* and can be modelled by *Diagrams*. Any diagrams can of course be attached with a system function.

A system function is realized by one or more *Function blocks* in the case of IEC 61131-3 programming or similar. A function block is an instance of *Function block type* in the function block library.

The model also includes the classes needed to carry out the Performance Level (PL) evaluation according to ISO 13849-1 [18]. Neither the IEC 62061 nor the IEC 61508 safety integrity level (SIL) is currently supported, although the *System function* specification is done according to IEC 62061[15]. *Safety function* must be represented for the PL evaluation in a manner that cannot be fulfilled by the *Structure* concept model presented in Section 3.6. Hence, a special set of classes is attached to *Safety function*. *Safety-related block diagram* needs to be drawn to define the logical structure of the safety function to illustrate which blocks (i.e., unities of parts) are logically connected in series and which in parallel in the fault tolerance sense. ISO 13849-1 gives guidance on drawing such diagrams. Such a diagram is in theory drawn for each safety-related part of a control system (SRP/CS)[16], but the model requires a combined block diagram to be connected to the safety function due to the fact that it is more common to present the combined diagrams. The model also 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[17] (SRP/CS) can be a one channel system or a two channel system. Such channels are denoted *Normal channels* in the model. *Test channel* may also be defined.

---

first one is the normal centre position stop and the second one is the dead-man's switch – enable valve – channel. Now this leads to the fact that half of the safety function is allocated to 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.

[14] Safety function is not a sub-function in the sense that the system function does not necessarily call it, but the safety function exists along with the system function to provide the necessary functional safety measures.

[15] The reason for adopting the IEC 62061 function specification format while otherwise following ISO 13849-1 is that ISO 13849-1 does not provide such a systematic safety function specification as IEC 62061.

[16] 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.

[17] Note that the SISTEMA tool by BGIA calls these subsystems. As ISO 13849-1 does not use this term, we simply call them SRP/CSs and, in fact, SISTEMA treats them as SRP/CSs according to ISO 13849-1 even though it uses redundant terminology.

The *Normal channels* and *Test channels* are special cases of the *Channel* class. Each channel consists of *Blocks* and *Block* consists of *Parts*.

The attributes of the classes are defined and described in the context of the database model discussed in Section 4.1. A full list of attributes is not provided in this research note however.

3.9 Documents

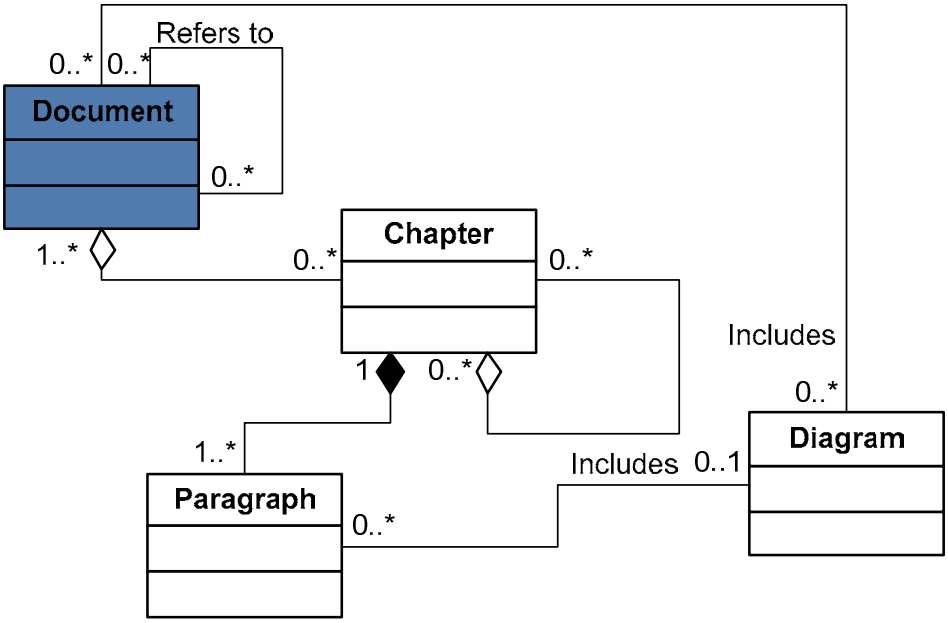


Figure 31. Document concept model.

The *Document* concept model in Figure 31 is simple: *Document* contains *Chapters* that can consist of lower level *Chapters*. A chapter consists of one or more *Paragraphs*. A paragraph can contain one or more *Diagrams*.

A chapter can belong to one or more *Documents*. The idea is that *Chapter* text can be used in different documents through dynamic linking such that if a change is made to a chapter it is reflected in all the documents that include that particular chapter.

The *Document* model is a simple book-like model. In fact, the *Chapter-Paragraph* model is only useful in cases in which the same chapters or paragraphs are used in several documents and the paragraphs do not contain info from the other artefacts, such as *Requirements*.

The attributes of the classes are defined and described in the context of the database model discussed in Section 4.1. A full list of attributes is not provided in this research note however.

3.10 Network (CANopen)

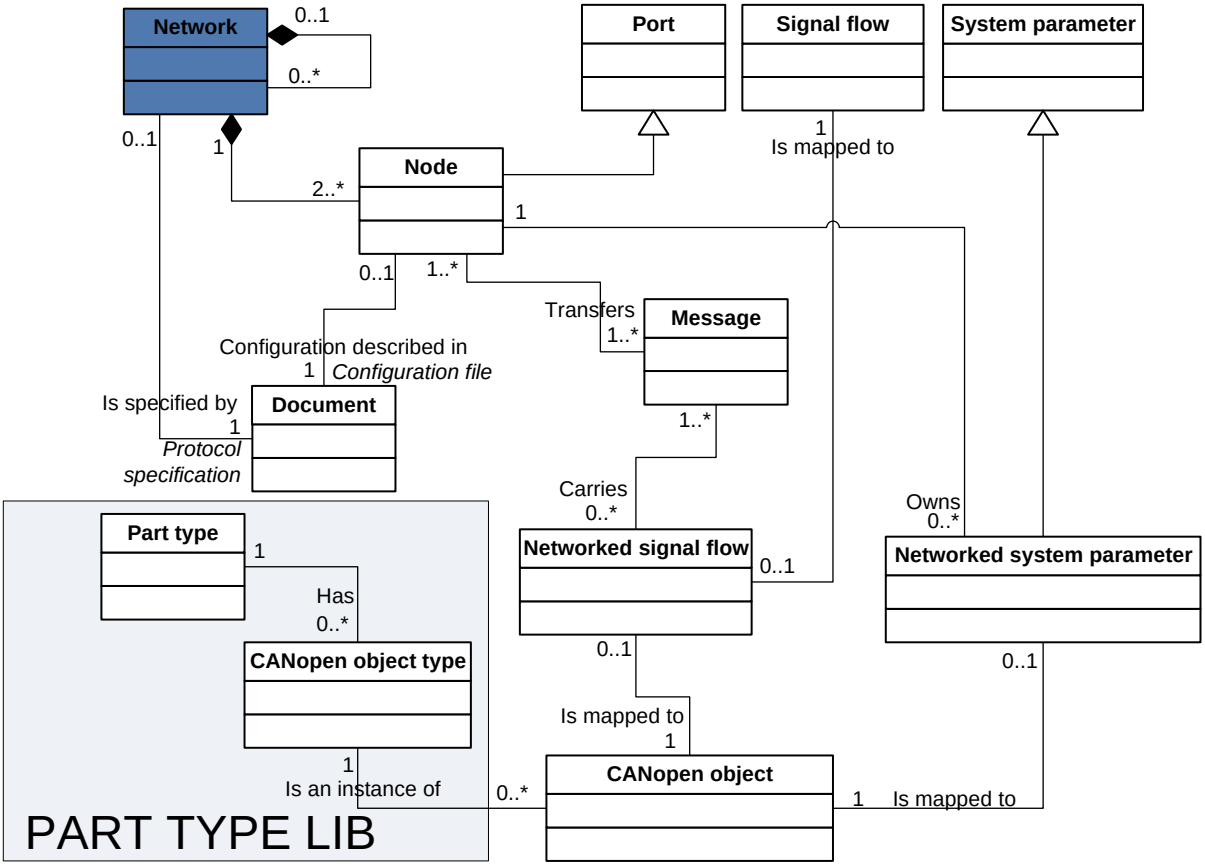


Figure 32. Network concept model (draft).

A draft of the network concept model is depicted in Figure 32. *Network* consists of *Nodes* and sub-networks. A node is a special case of *Port* (see description of *Ports* in Section 3.6). A network is specified by a protocol specification and by a message specification structured as a *Message* class. Messages are owned by *Nodes*. A message can carry one or more *Network signal flows*. A Network signal is mapped to *Signal flow* (see description of Signal flows in Section 3.6). A network signal flow is mapped to *CANopen object* (in the case of CANopen networks). A CANopen object is an instance of *CANopen object type* owned by *Part 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 parameters*. In the case of CANopen, access by such parameters is through the SDO service. The necessary parameters to do this can be found in *CANopen object* (actual value) and in *CANopen object type* (other parameters).

Note that this model is not complete and has neither been tested nor demonstrated due to the fact that network modelling was not part of this work. The model above was only created to ensure that the *Structure* model can be linked to the network model. The model above is CAN-bus- (especially CANopen) oriented and needs to be generalized to cover other communication protocols. The node and

protocol specification – and message specification – concept model is generic and should work with any communications system type, though the attributes may differ.

The attributes of the classes are defined and described in the context of the database model discussed in Section 4.1. A full list of attributes is not provided in this research note however.

4. IIDAbase implementation examples

The data model was tested and demonstrated in two environments: a MySQL-MS Access relational database environment (see Section 4.1) and an Application Lifecycle Management tool called Polarion ALM (see Section 4.2).

The MySQL-MS Access environment worked as a convenient and quick environment to create forms with sub-forms to test the concepts and relations between the tables, whereas Polarion ALM provided the version management, traceability feature, issue management and workflow feature necessary to create real-world implementations of the IIDAbase concept.

4.1 Database implementation

4.1.1 Introduction

A database model was created according to the IIDAbase concept model and implemented as a MySQL database. The database model was designed by the MySQL Workbench tool with its EER[18] notation. From the MySQL Workbench, the SQL scripts were generated to create the database on a MySQL database server. It is assumed that for each system under development, a dedicated schema is created in the database server.

The notation used in the database model diagrams applies the UML style for presenting multiplicity at the ends of the association lines. A solid association line means an identifying relationship, i.e., the related items cannot exist without a parent item while a dashed association line means a non-identifying relationship, i.e., the related item can exist without a parent item. See Figure 33 for the EER notation style.

[18] EER stands for Enhanced Entity-Relationship, but there are other EER notations that do not resemble the MySQL Workbench EER notation.

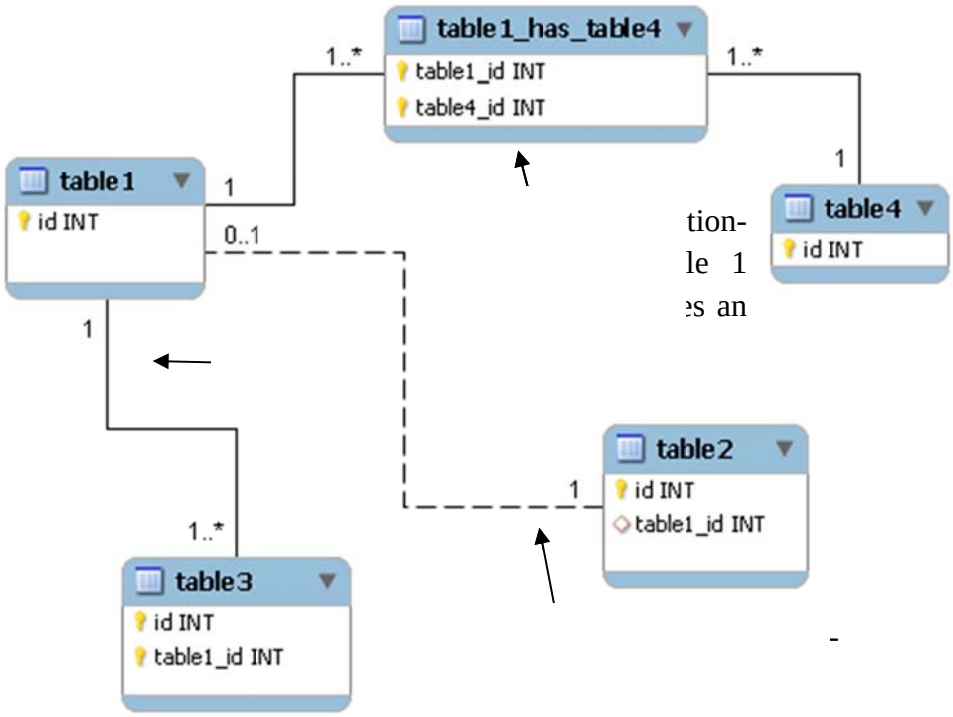


Figure 33. Guide to interpreting the EER model notation.

4.1.2 Core of the database model

Figure 34 illustrates the core of the database model.

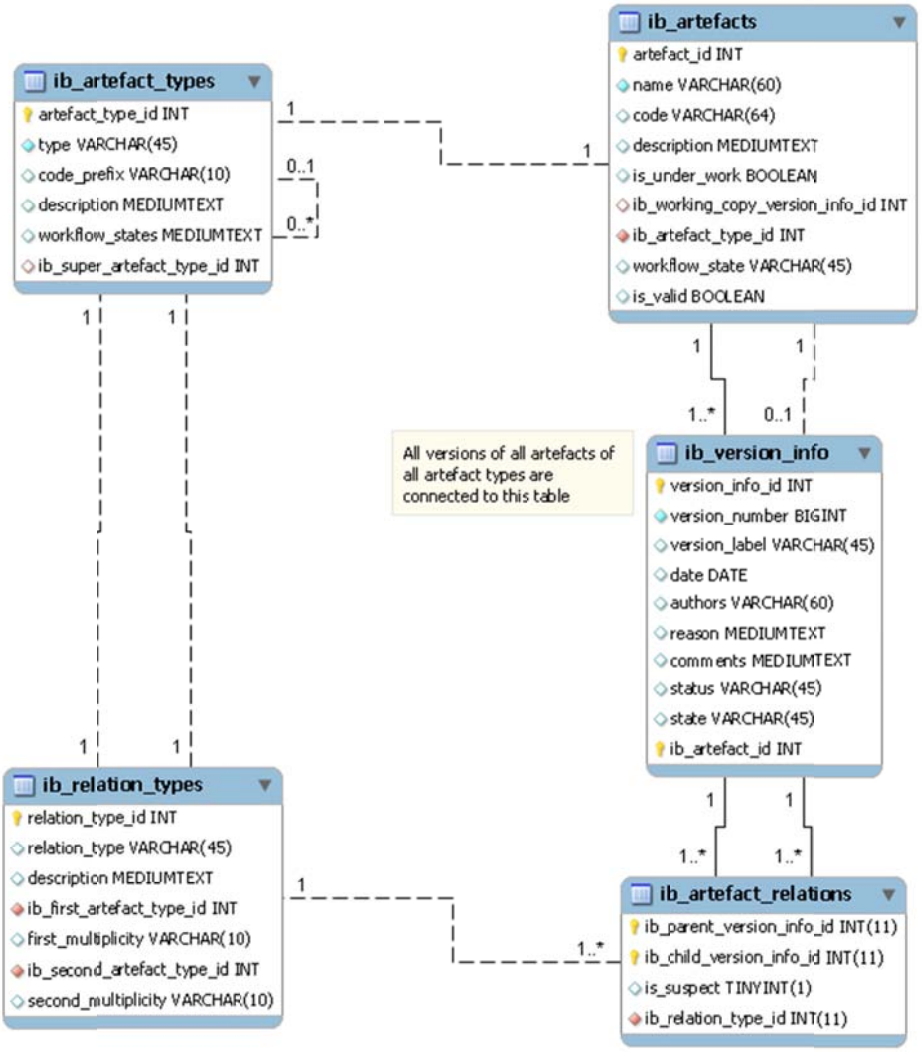


Figure 34. Core of the database model.

The *ib_artefact_types* and *ib_relation_types* tables store all the information in the concept model described in Chapter 0. Each artefact in the system must be linked to an artefact type provided by the concept model, and each artefact relation must be linked to a relation type provided by the concept model. This helps build a systems engineering tool that guides the creation and management of the links and their relations to each other. For example, when the system engineer wants to add an artefact to the IIDAbase repository, the tool can list the available artefact types. Once the system engineer has selected the artefact type he wants, the tool can list the artefact types that must or can be associated with this artefact type. In fact, this provides a guided workflow in the sense that if, for example, the system engineer starts his work by creating a *Use case act* artefact, according to the model there must

be a *System use case* artefact associated with a *Use case act* artefact; hence the tool insists on creating a *System use case* artefact, but as soon as he creates a *System use case* artefact, the tool insists on creating a *Behaviour* artefact and again while creating the *Behaviour* artefact, the tool insists on creating a *System* artefact. Hence, although the workflow is in a sense incorrect, the tool can be programmed with the help of this core database model to ensure inclusion of all the necessary artefacts needed for the integrity of the system artefacts.

The *ib_artefacts* table stores the generic part of each artefact, the *ib_version_info* table stores version information of each artefact version and the *ib_artefact_relations* table is used to make the relations between the artefacts according to the model. The artefact-specific part of an artefact type is linked to the *ib_version_info* table as depicted in Figure 35.

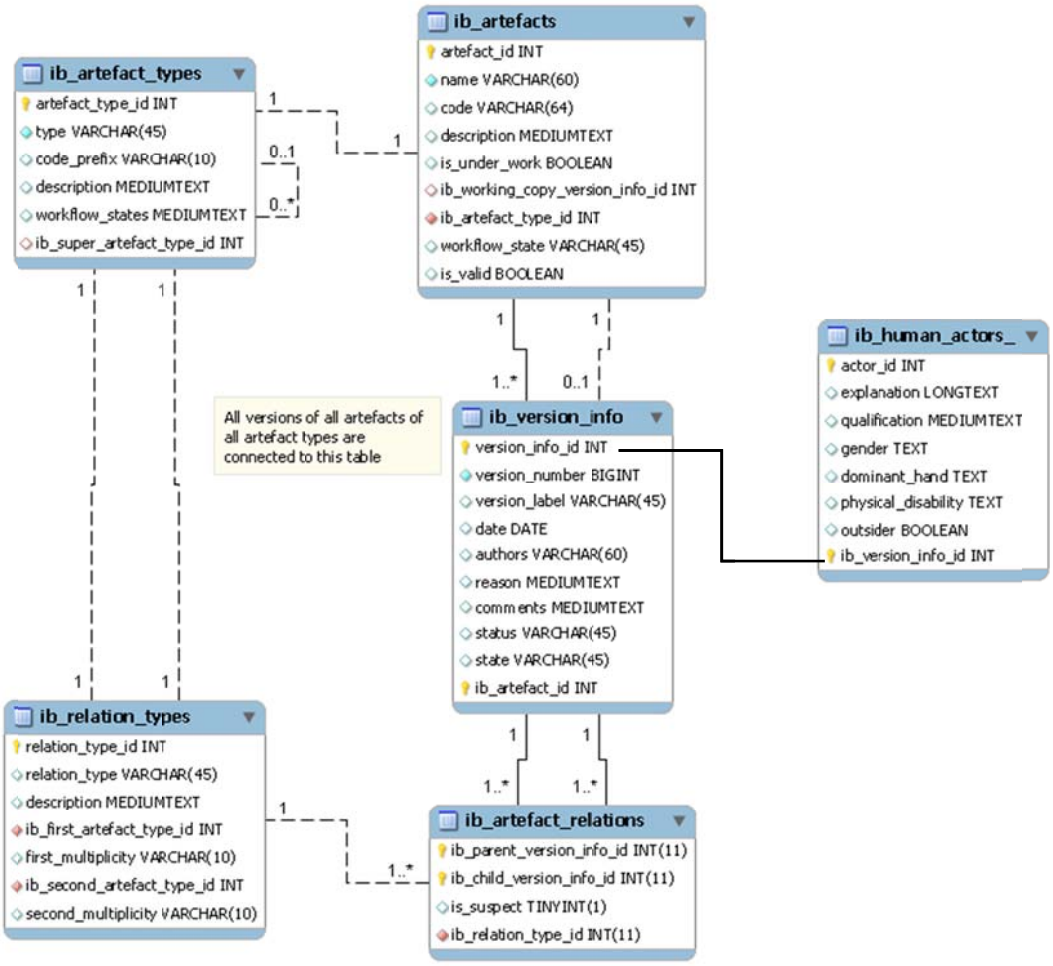


Figure 35. Linking the artefact types to the core model; example: *Human actors*-artefact

The core model presented above supports version management such that the versions of the artefacts are stored in the same database or the versions are stored in a version management tool like Subversion. In the latter case, the format to store a database record as a line of text into the Subversion repository needs to be defined. Such a format was not defined in this work.

In Table 3, one of the artefact tables, namely *ib_human_actors*, is described in detail.

Table 3. Description and attributes of the *Human actor* artefact.

Table name			***Table description***
ib_human_actors_			Lists all the human actors dealing with the system, whether they are users of the system or outsiders. This table fulfils the requirement of ISO 4121-1, 5.2 b), c) and d).
Attribute name	**Attribute type**	**Can be null**	**Column description**
actor_id	int(11)	NO	Database id of the actor
explanation	longtext	YES	A more detailed description of the actor role
qualification	mediumtext	YES	Description of the necessary training etc. for the actor to be qualified to perform the use case [ISO 4121-1, 5.2. c)]
gender	text	YES	If it matters whether this role is occupied by a male or a female, the description of such considerations is written here [ISO 4121-1, 5.2. b)]
dominant_hand	text	YES	If it matters whether this role is occupied by a left-handed or a right-handed person, the description of such considerations is written here [ISO 4121-1, 5.2. b)]
physical_disability	text	YES	If it matters whether this role is occupied by a person with a physical disability (like a visual or hearing impairment, size or strength), the description of such considerations is written here [ISO 4121-1, 5.2. b)]
outsider	tinyint(1)	YES	TRUE if this actor role is not a user of the system, e.g., operators of adjacent (possibly similar) machinery, non-user employees in the vicinity, non-employees in the vicinity; the description shall be given in the description field. [ISO 14121-1, 5.2 d)]
ib_version_info_id	int(11)	NO	Link to the version of the Human actor artefact to which this content belongs

The attributes of all the artefact types presented in Chapter 0 were defined in the same way using the MySQL Workbench tool.

4.1.3 MySQL-MS Access demonstration

The data model was tested and demonstrated with an MS Access 2007 database application called IIDAdesktop created in the project. The MySQL tables were linked as external tables to IIDAdesktop. The main user interface of IIDAdesktop is illustrated in Figure 36.

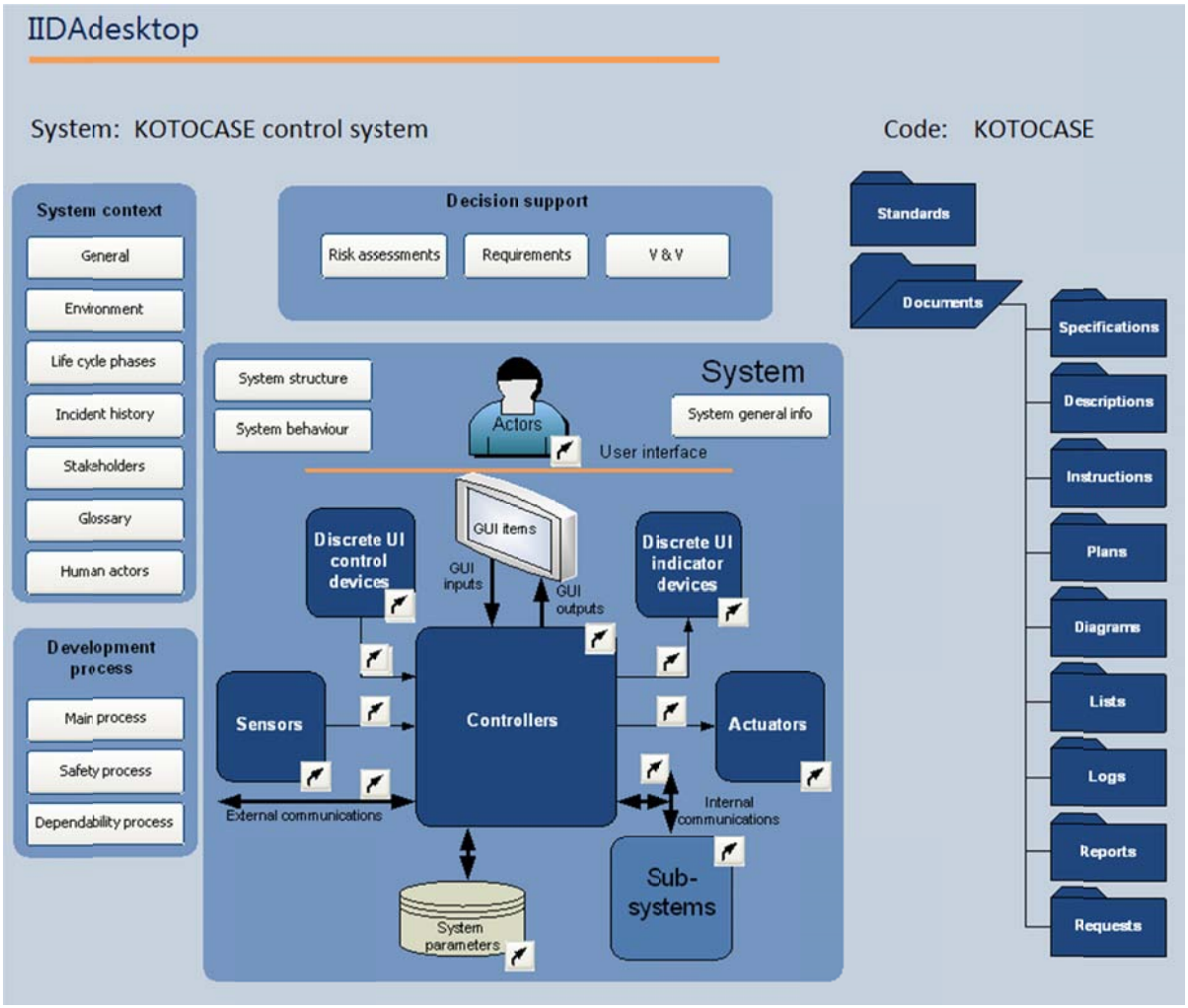


Figure 36. IIDAdesktop user interface.

IIDAdesktop allows browsing of the system engineering artefacts. The upper left corner of the user interface collects the system context information needed for, e.g., risk assessment. By pressing, for example, the Environment button, the following form is shown (Figure 37):

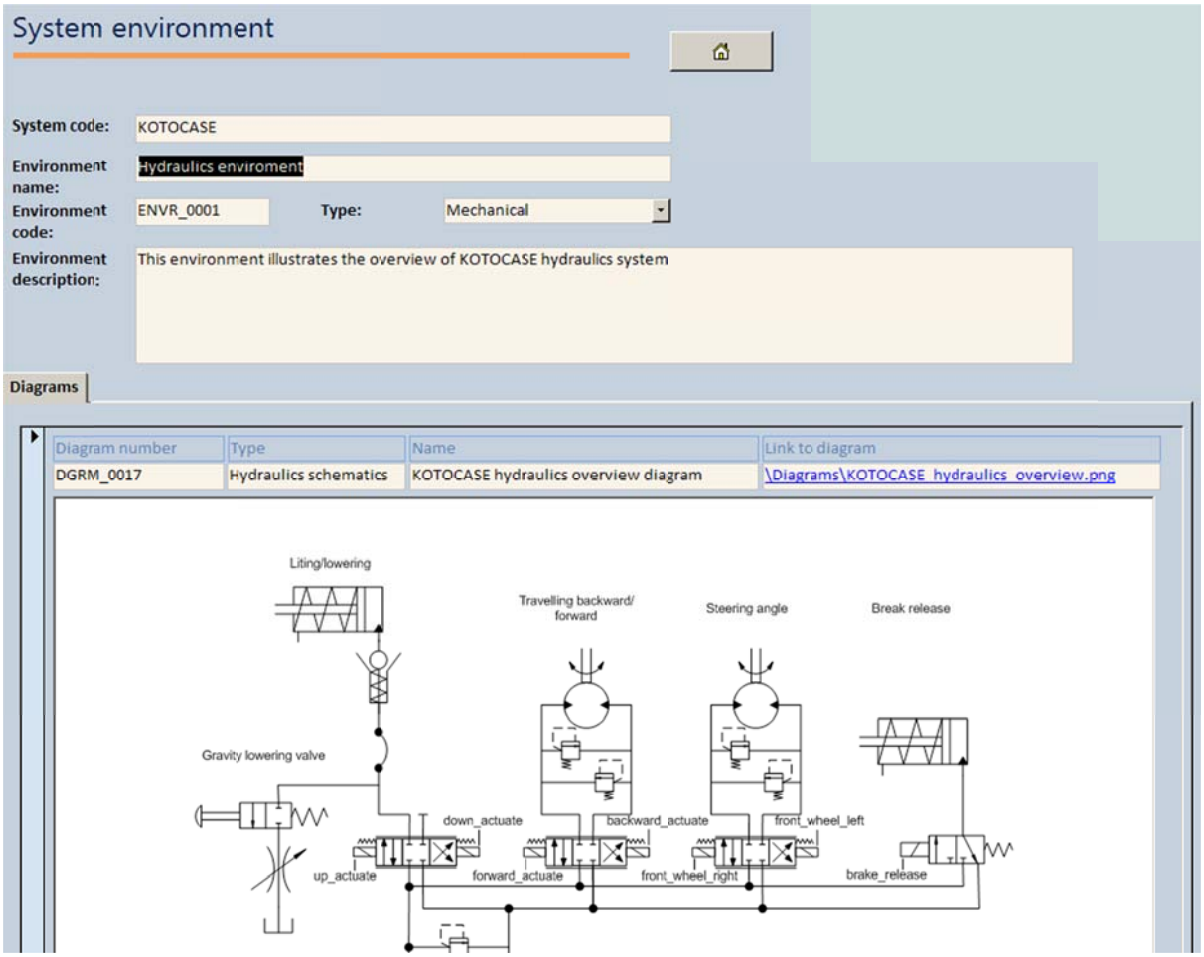


Figure 37. System environment form with Diagrams sub-form (only one environment is shown).

The system artefacts can be browsed from the middle of the IIDAdesktop user interface. For example, by pressing the System structure button, the following form is opened (Figure 38 and Figure 39).

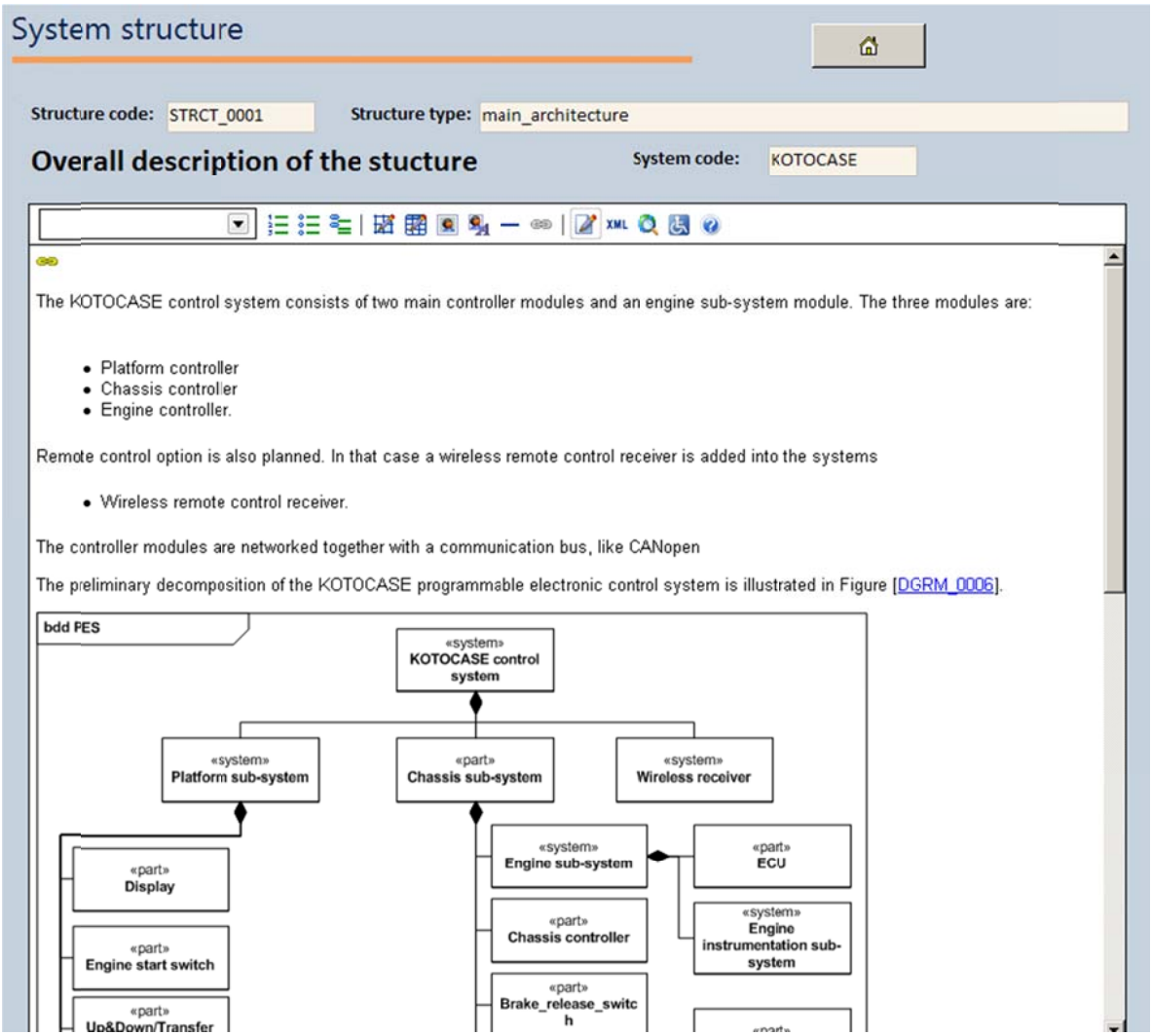


Figure 38. System structure form, part 1.

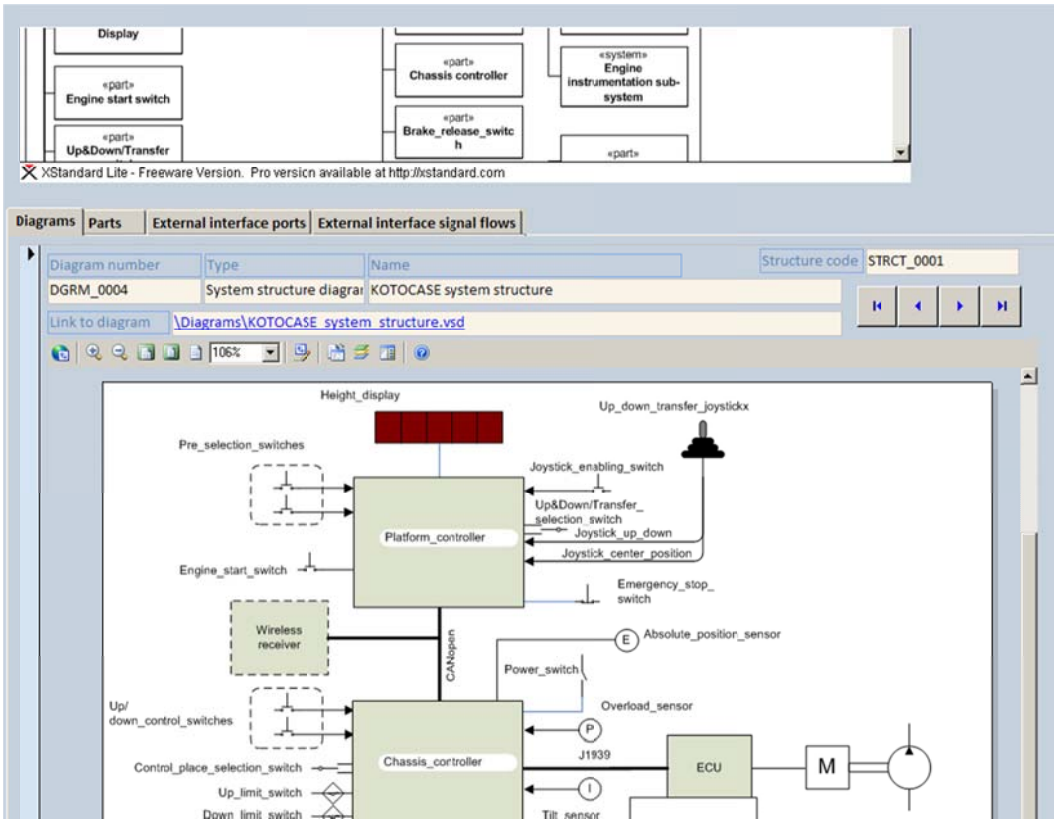


Figure 39. System structure form with Diagrams sub-form, part 2.

In Figure 39, the signal flow names of the schematic drawing can be fetched from the database, e.g., the *Joystick_up_down* signal is attached to IIDAbase, and hence if the signal name is changed in the database the figure is updated accordingly. MS Visio was used as the drawing tool. Furthermore, by clicking the *Joystick_up_down* signal a list of the signal attributes is opened as shown in Figure 40.

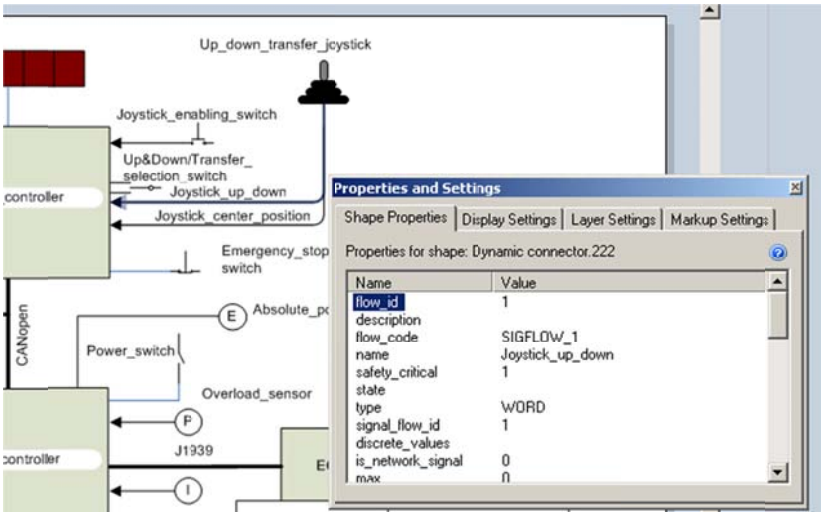


Figure 40. Browsing the *Signal flow* attributes.

The attributes of the parts and the signal flows of the structure can also be browsed by pressing the appropriate arrow button in the IIDAdesktop user interface, e.g., the *Discrete UI control devices* arrow produces the following view (Figure 41):

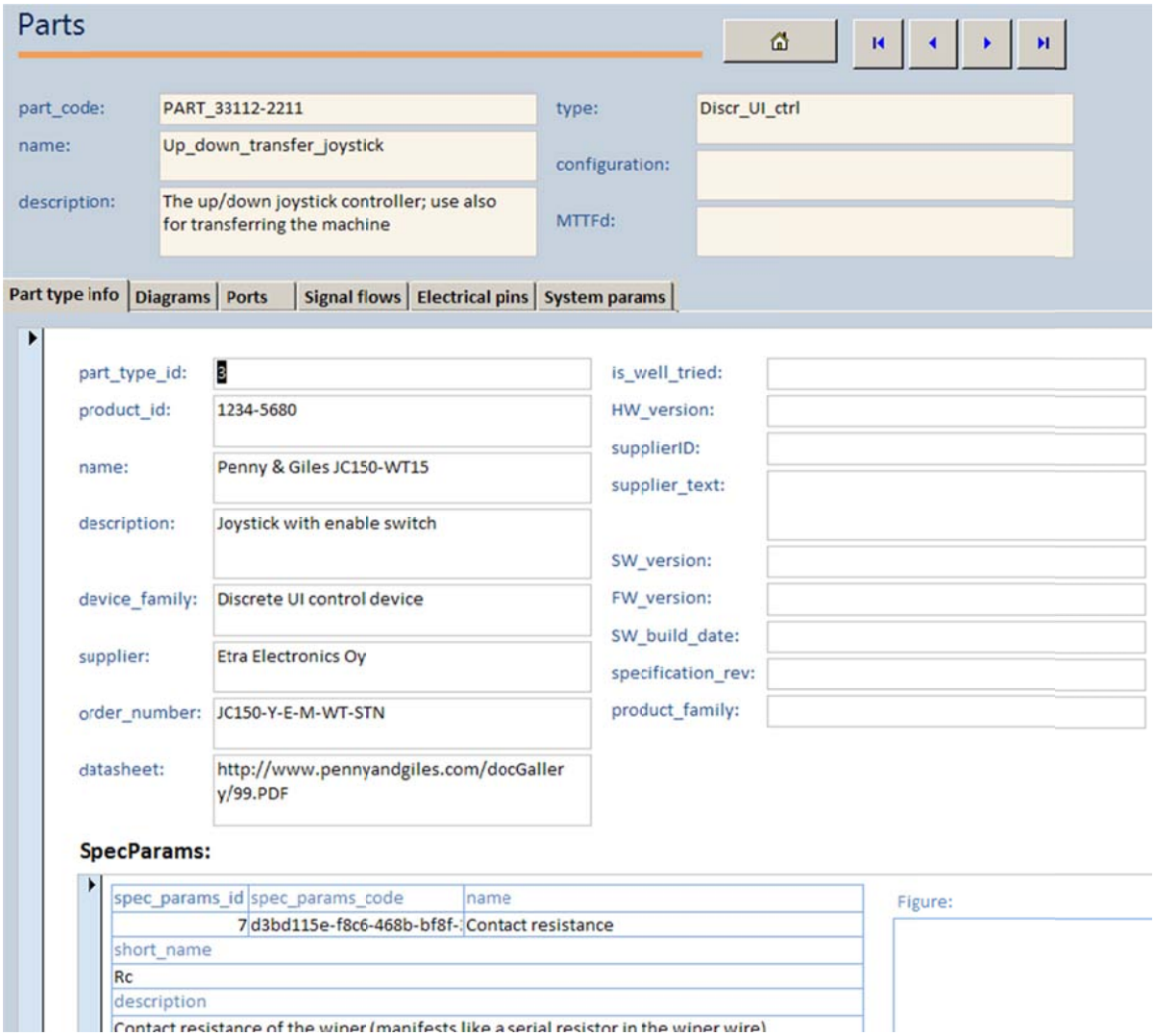


Figure 41. Form for browsing parts and their specifications.

System parameters and subsystems can be browsed from the respective arrow buttons in the IIDAdesktop user interface.

By pressing the Behaviour button in the user interface, a form is opened that provides access to the system use cases (see Figure 42 and Figure 43).

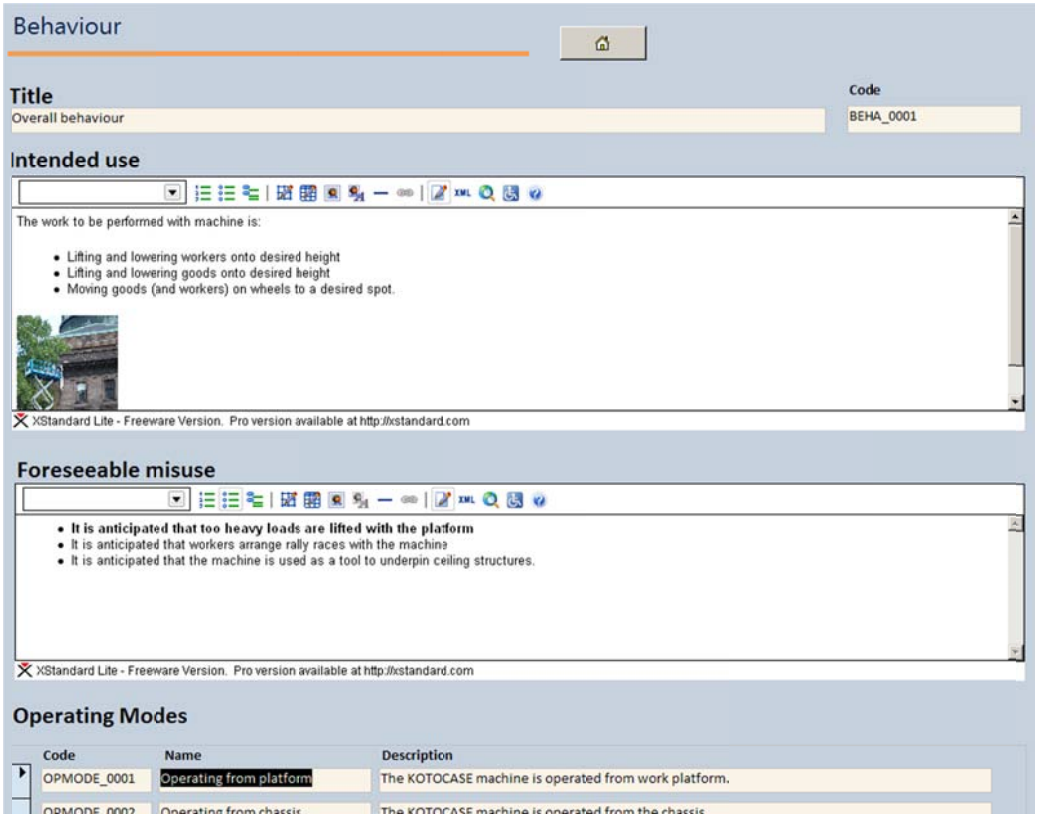


Figure 42. System behaviour form, part 1.

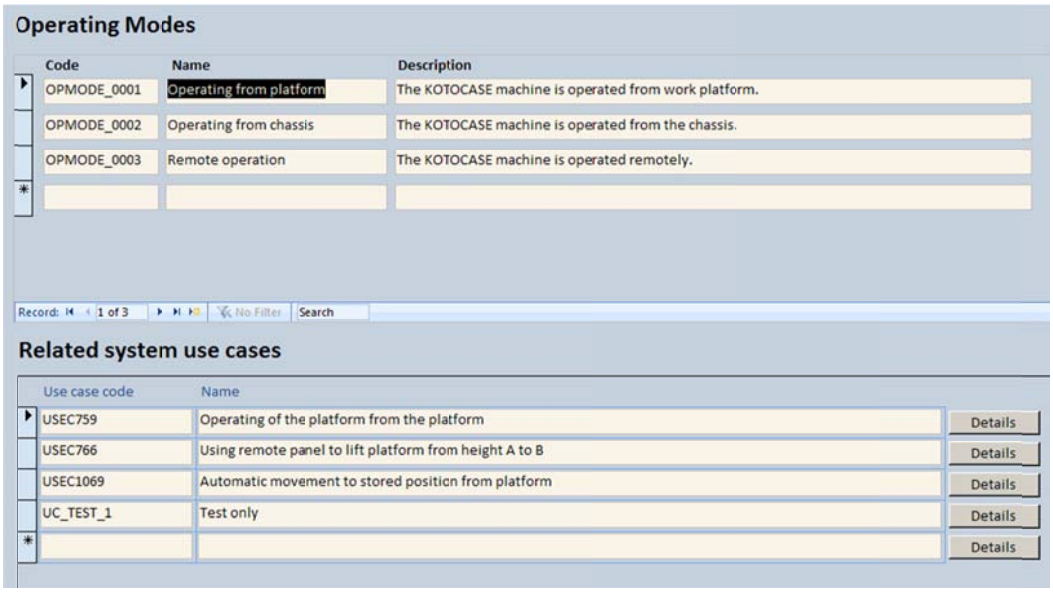


Figure 43. System behaviour form, part 2.

When the Use Case details button is pressed in the General behaviour form, the System use case form is shown as in Figure 44.

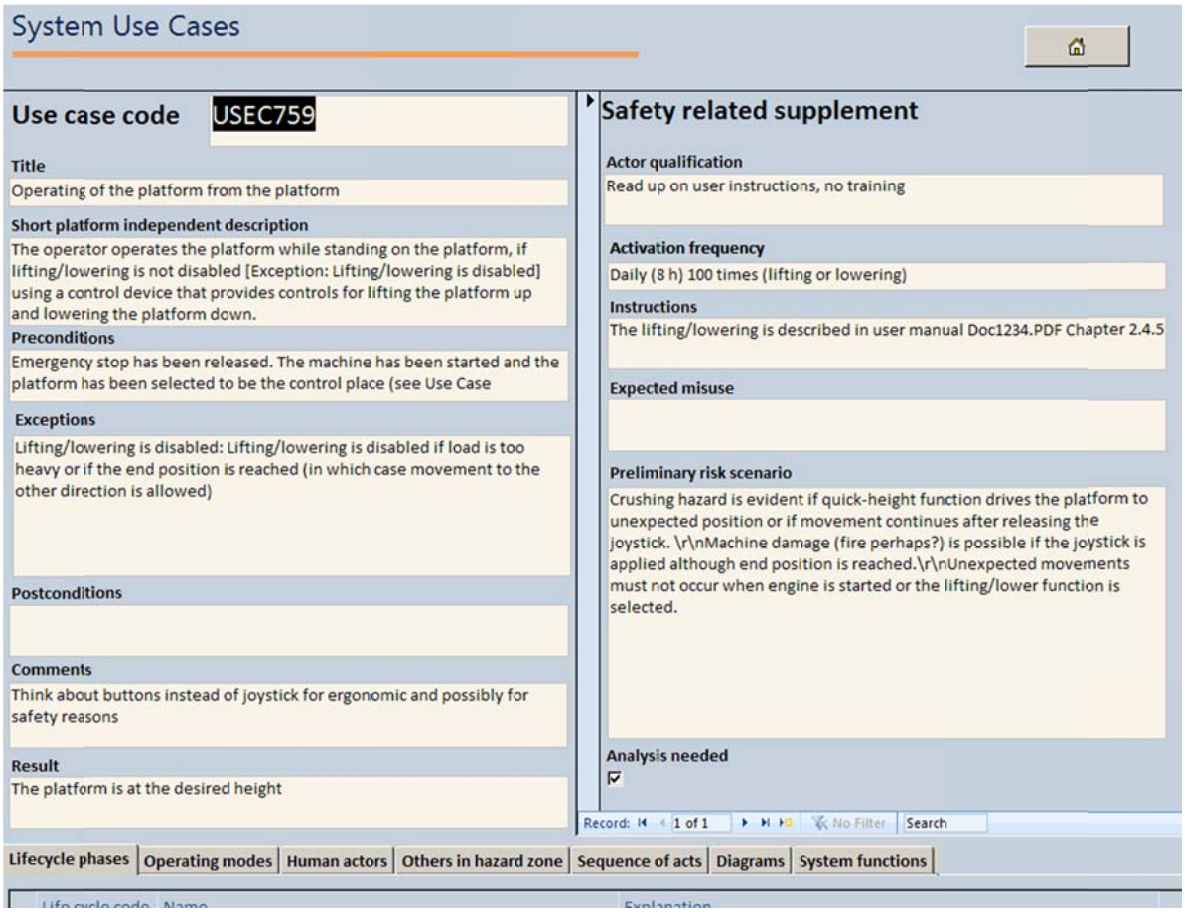


Figure 44. System use case form.

The bottom part in Figure 44 shows the artefacts related to the particular use case. They can be browsed by opening the corresponding sheet, e.g., the *Sequence of acts* sheet as shown in Figure 45.

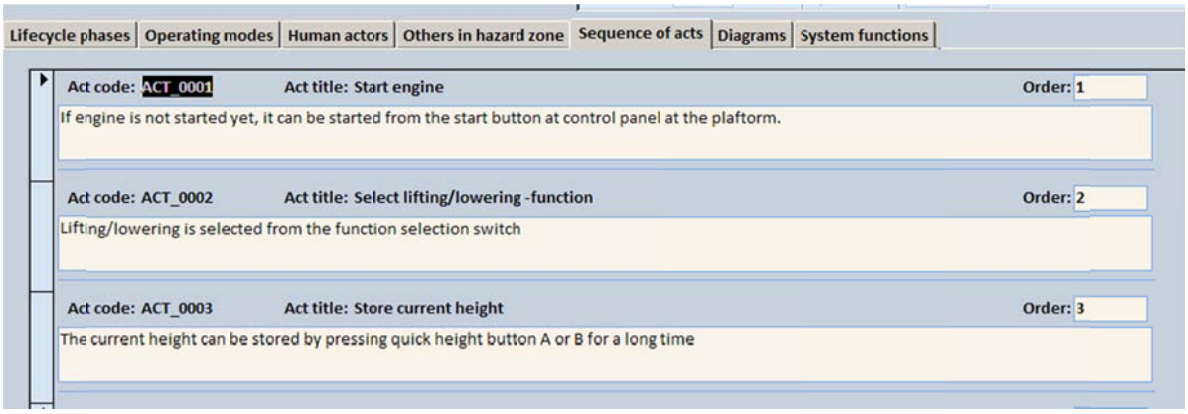


Figure 45. Sequence of acts sheet.

The *Decision support* part of the IIDAdesktop user interface collects the entry points to the risk assessments, the requirements management and the verification and validation. The Risk assessment button opens the following form (Figure 46):

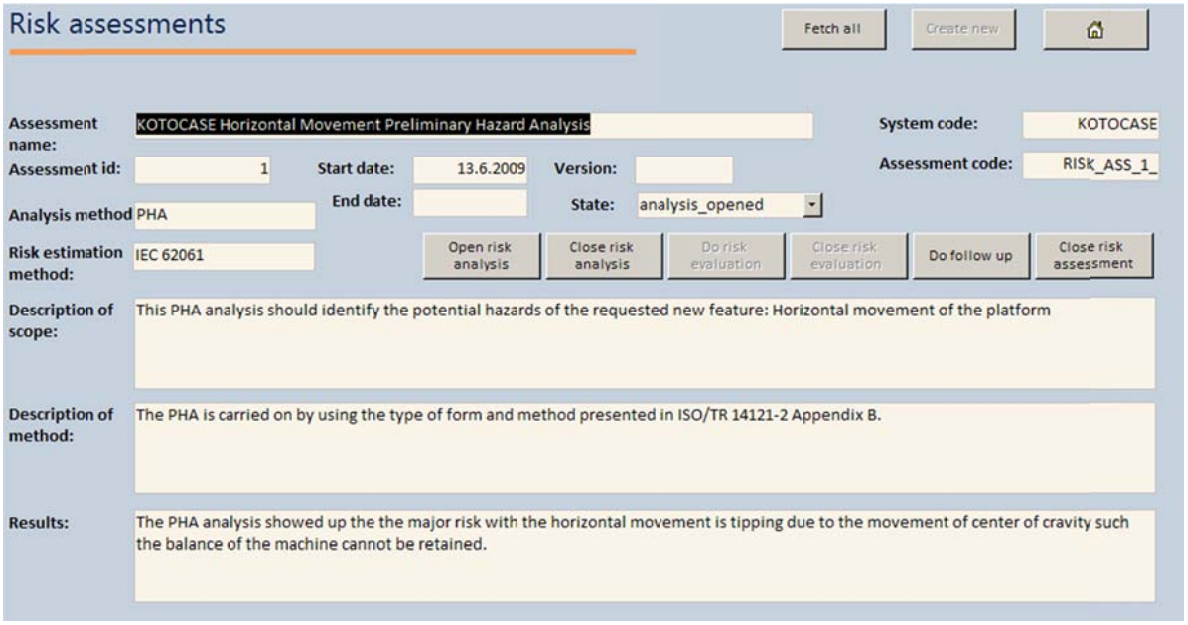


Figure 46. Risk assessment form.

By pressing the Open risk analysis button in the form above, the corresponding analysis tool is opened (in this case the PHA tool) (see Figure 47).

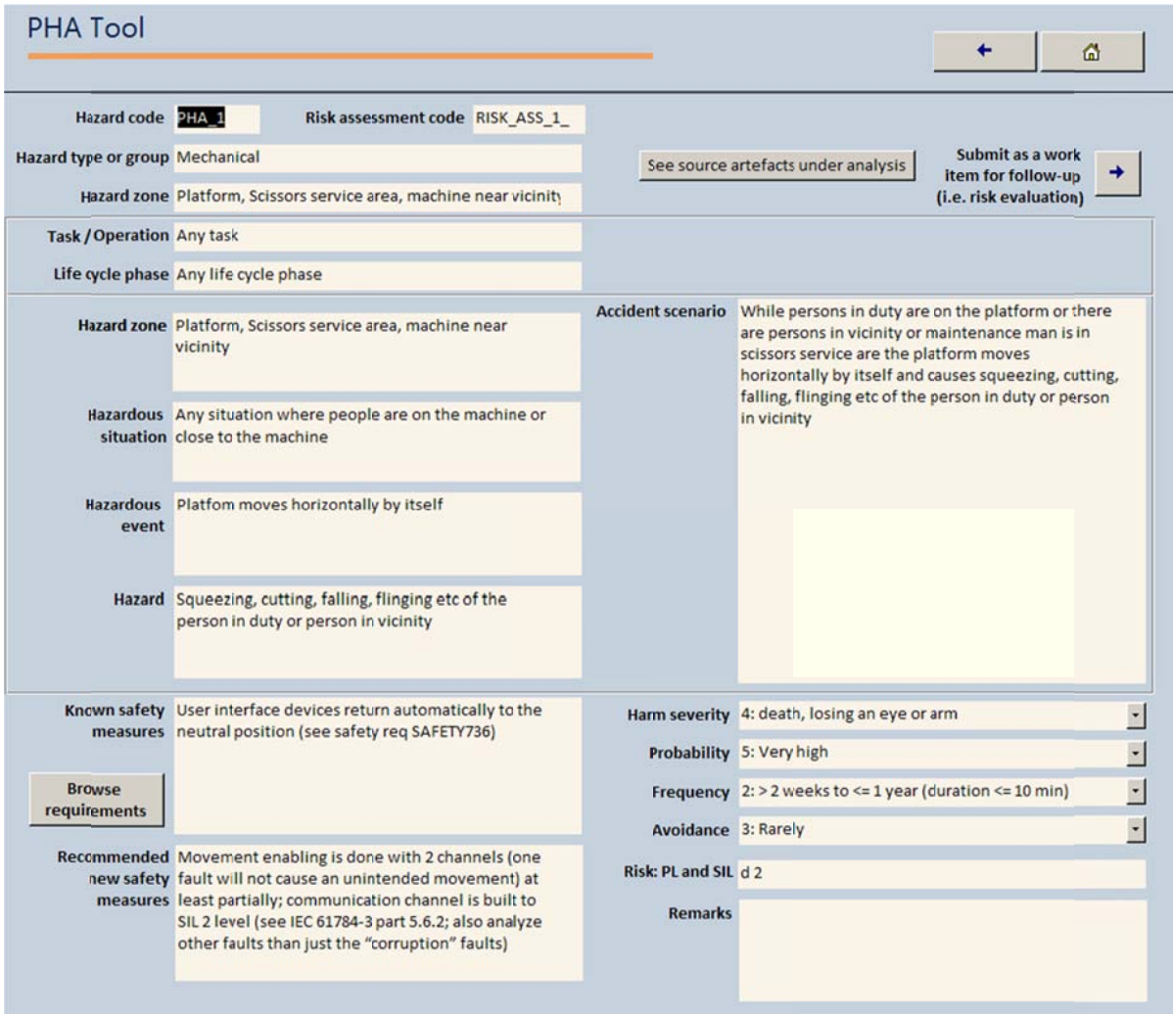


Figure 47. PHA tool.

This tool can also be opened by first pressing the Safety process button on the IIDAdesktop user interface, which opens the safety process outline figure (see Figure 48), and then pressing the Preliminary Hazard Analysis (PHA) box, which produces the PHA process illustration in Figure 49. From there the PHA tool is opened from the tools button in the middle of the figure.

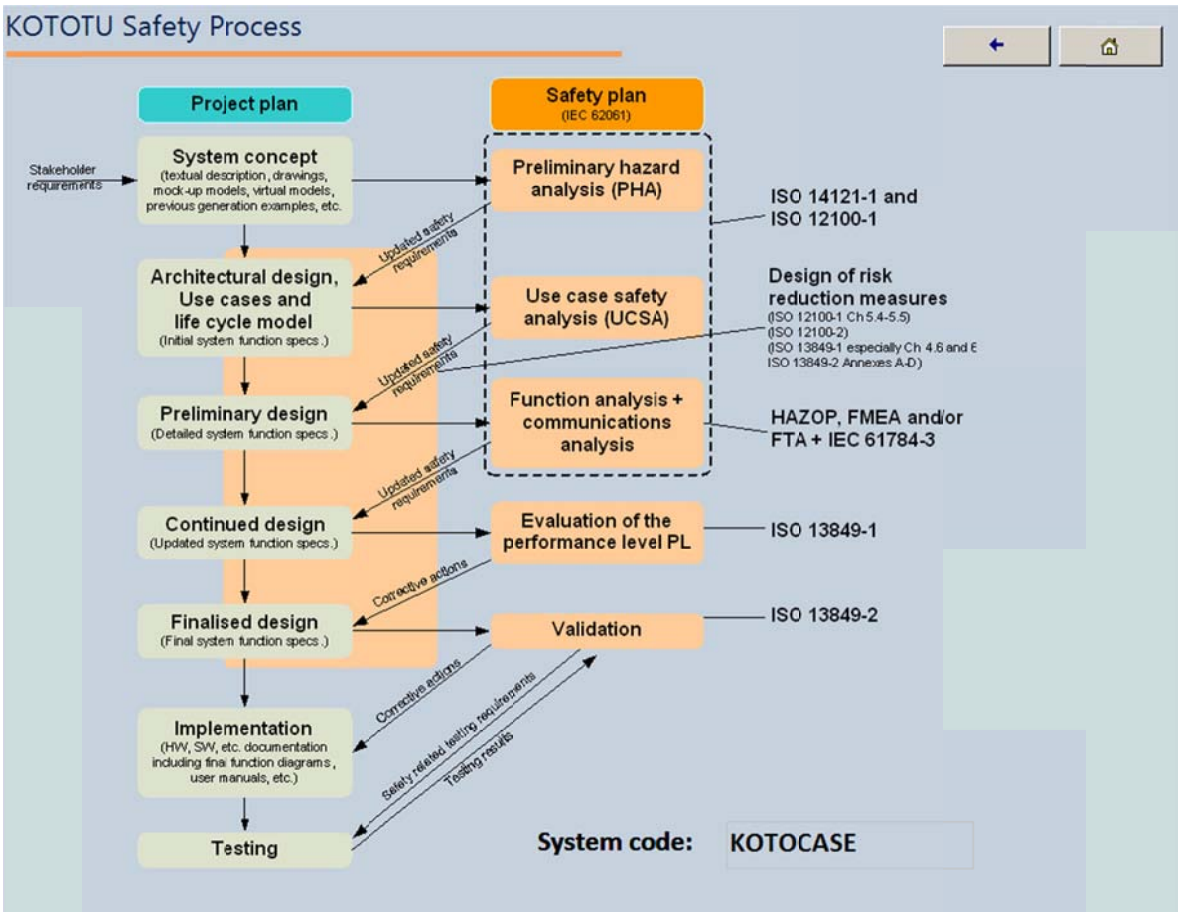


Figure 48. Safety process outline view.

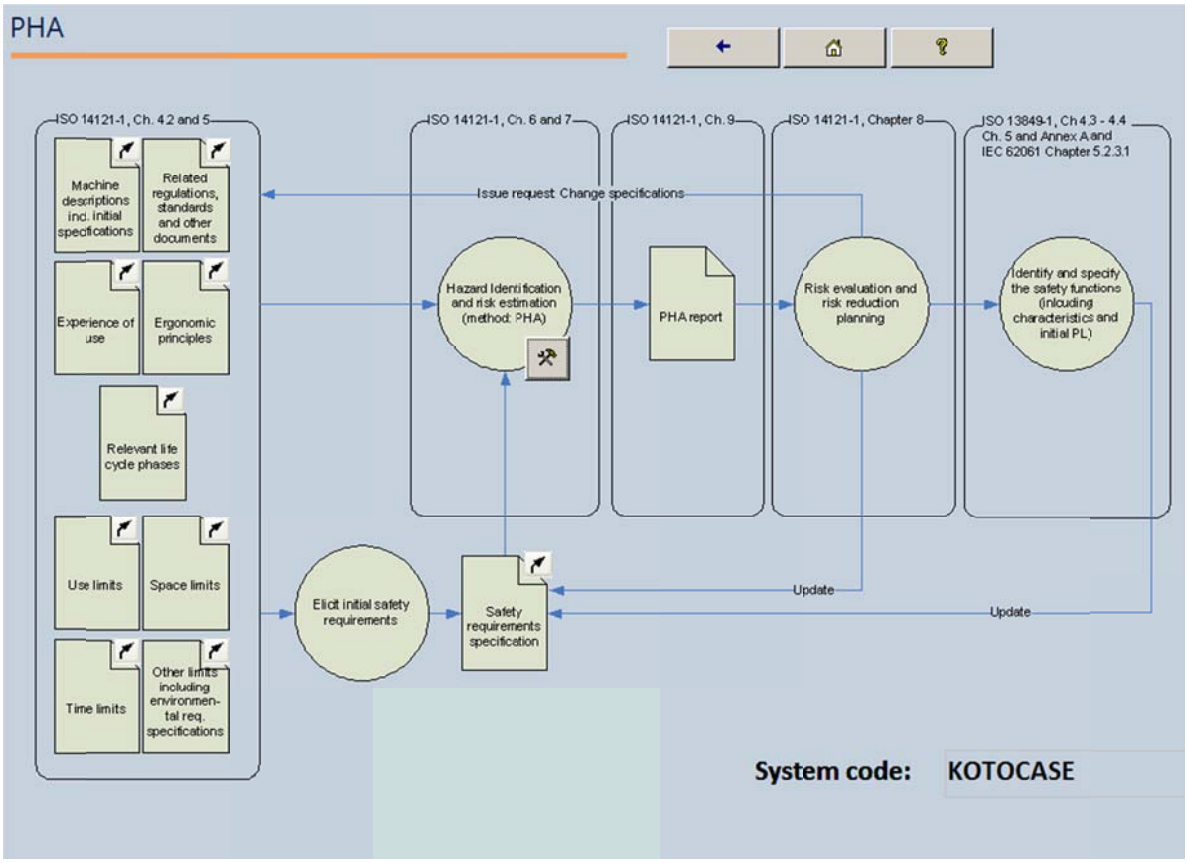


Figure 49. PHA process illustration.

In the IIDAdesktop main window, the Requirements button has a twofold function: pressing the left side of the button opens the Polarion ALM tool and shows the requirements, and pressing the right side of the button opens a form to browse the requirements from the MySQL database, if the requirements are mirrored from the ALM tool in MySQL.

The documents for the system can be browsed through the folder list on the right side of the window (see Figure 36). By pressing the folder icon, the document management tool that is used is opened. In this case, pressing, for example, the Standards folder opens the Polarion documents repository (see Figure 50).

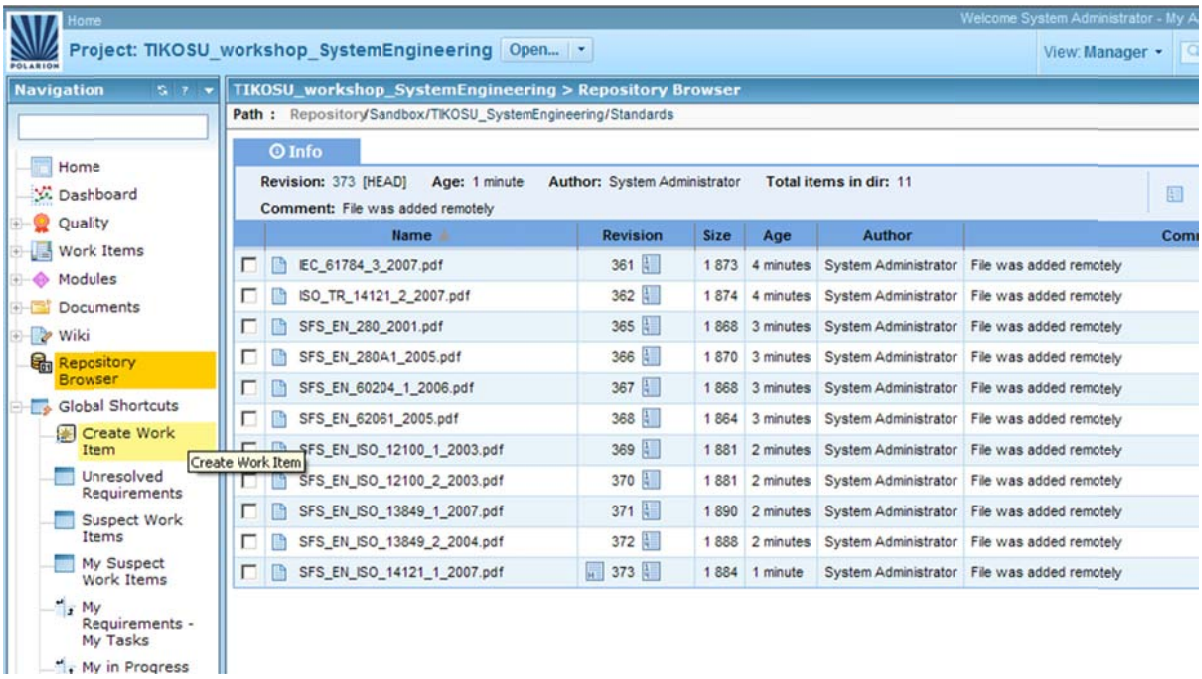


Figure 50. Polarion repository: Standards folder.

The Diagrams folder opens an internal diagrams browser however (see Figure 51).

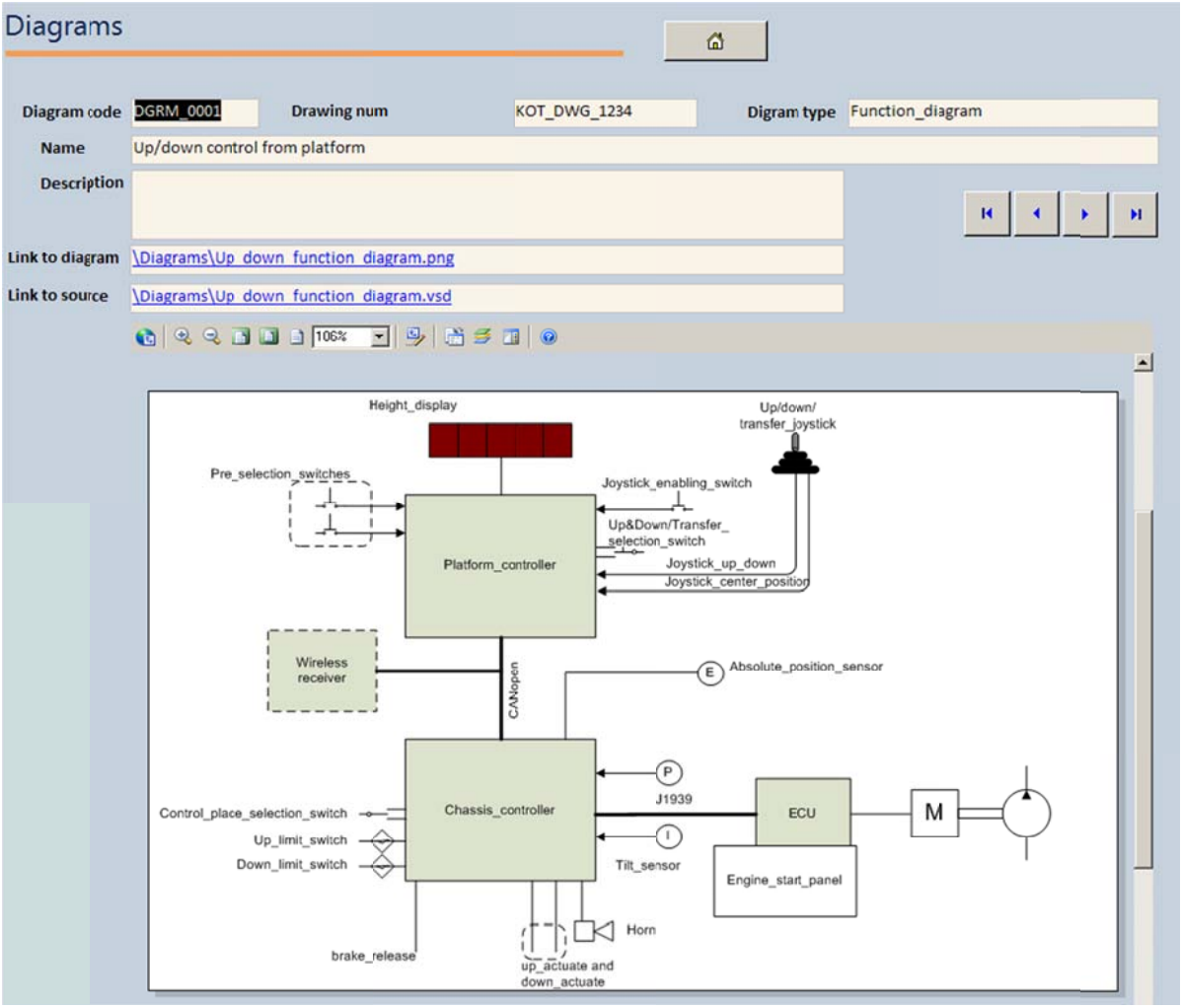


Figure 51. Diagrams browser.

4.1.4 Discussion

The MySQL/Access demonstration proved that the model could be implemented. The demonstration showed the power well of an integrated systems engineering environment based on the IIDAbase integrated artefacts repository.

It was noted that commercial systems engineering 'composer' tools for drawing architecture diagrams to connect to IIDAbase were not available[19]. If we compare the situation with one of designing electronics, it is like an electronics designer drawing the circuit schematics with ad-hoc drawing tools like Paintbrush etc. and the circuit board designer then starting to design the circuit board based on the

[19] There are some SysML tools that connect to a database. These could be used with appropriate integration to IIDAbase. The SysML tools are restricted to SysML notation however.

drawing, without any supporting digital data on the components and interconnections. Hence, such a systems engineering composing tool is needed for drawing the system architecture diagrams to allow the electrical CAD designer, safety engineer and software engineers to use the digital data from the data repository to perform their part of the system design.

In this case, the Access demonstrator worked as a prototype of the IIDAdesktop tool. In practice, it may be more convenient to create the user interface on top of the chosen Application Life Cycle Management (ALM) tool or Product Life Cycle Management (PLM) tool with, e.g., Java programming, thus providing an IIDAbase profile on top of the ALM or PLM tool. The Access demonstrator provides an example of such an IIDAbase profile. In the project, the Polarion ALM tool was demonstrated but no graphical user interface similar to the IIDAdesktop demonstration tool was programmed; instead, the generic interface of Polarion ALM was used. The Polarion ALM implementation is presented in Section 4.2.

4.2 Application Lifecycle Management tool implementation

Polarion is an Application Life Cycle Management tool based on a repository created using the Subversion versioning system, and it provides a web-based graphical user interface; see Figure 52.

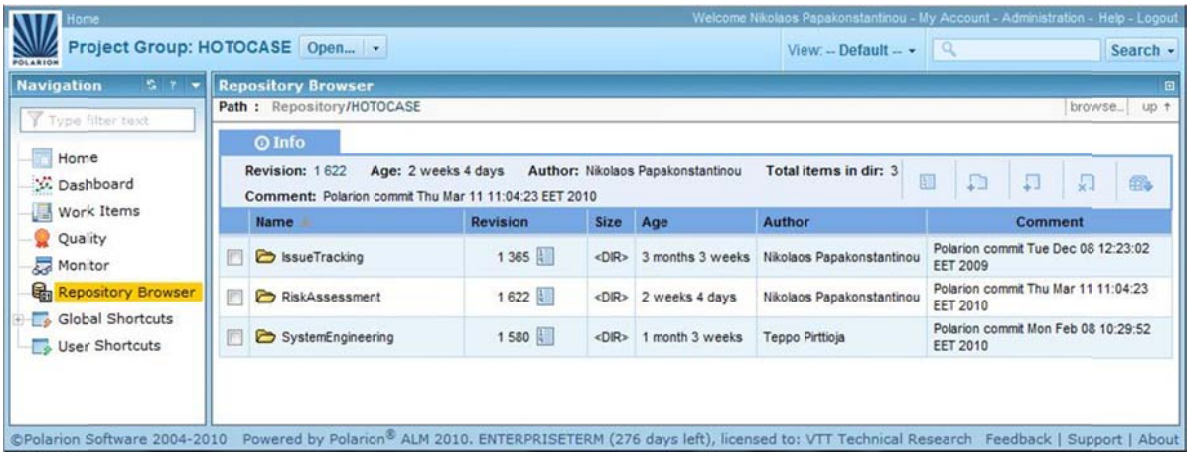


Figure 52. Polarion's repository browser.

In Polarion, artefacts are called Work Items (WIs) and are stored as flat, structured XML files. The WIs can have custom names and fields. Each WI can follow a customized state machine (e.g., 'created', 'open', 'closed'); see Figure 53.

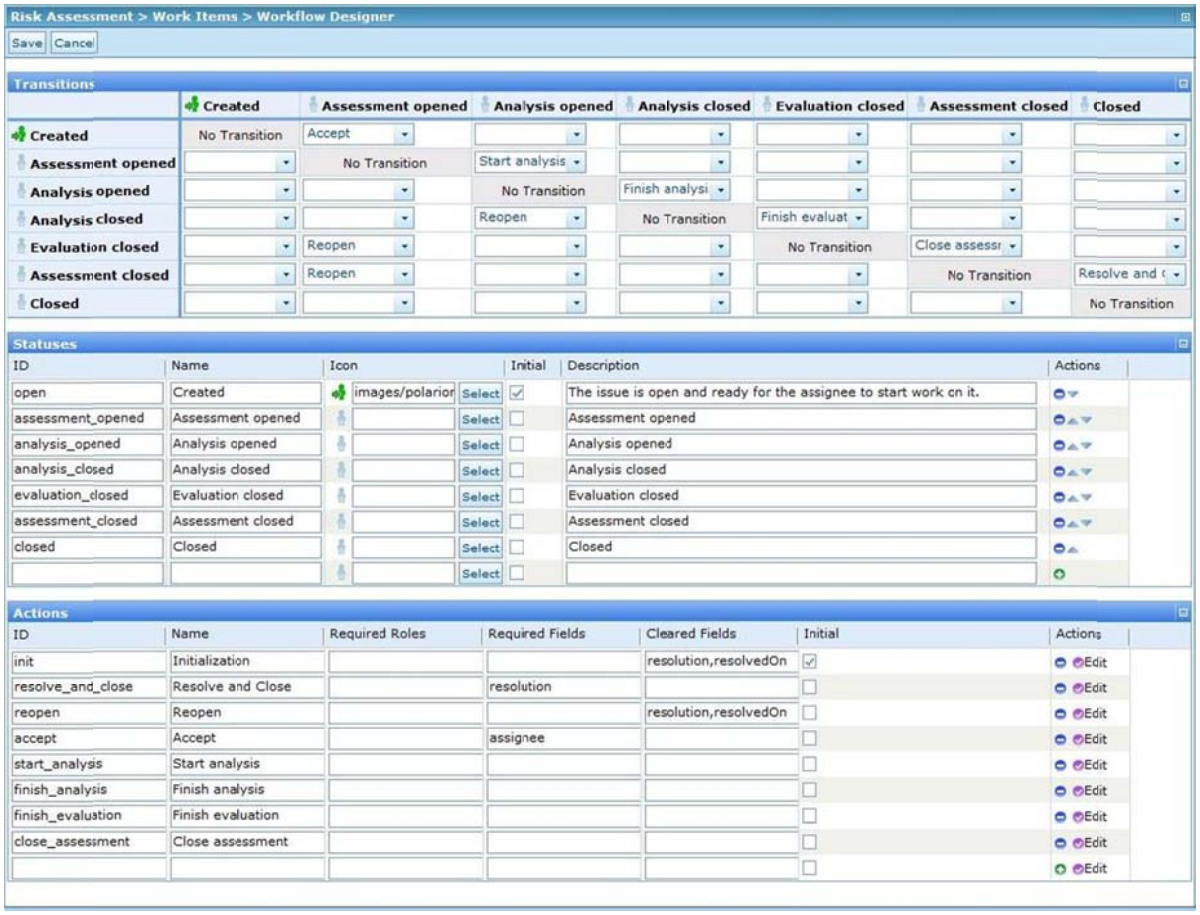


Figure 53. Polarion's workflow designer.

The WIs can be linked so a hierarchy can be created for traceability purposes and displayed as a tree; see Figure 54.

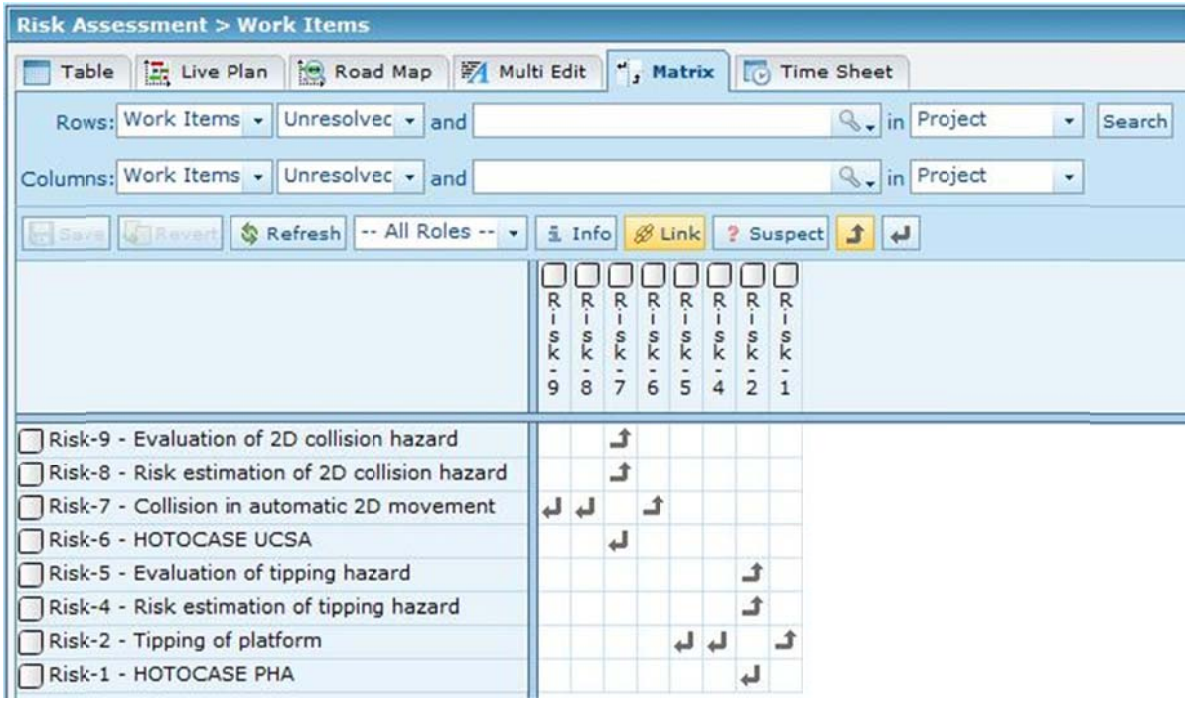


Figure 54. Viewing and editing links using the matrix view.

4.2.1 Polarion demonstration

To support the data model in Polarion, we created customized work items with custom workflows when necessary, related to issue tracking, system engineering and risk assessment. To demonstrate the implementation of the data model in Polarion, a case study presents the system engineering workflow of adding a feature to a machine's design (the HOTOCASE machine, a virtual scissors lift platform) and the required safety workflow that proceeds in parallel. The KOTOTU safety process will be followed (see Figure 21).

The new feature that will be added is the horizontal movement of the platform of the mobile elevating platform machine. The machine supports manual movement and automatic movement to a stored position, so the new feature should also support these two modes of operation.

A project group called HOTOCASE was created and within it three projects related to issue tracking, the system engineering and the risk assessment aspects of HOTOCASE. The first step is the creation of a *Feature Request* work item, as presented in Figure 55.

4. IIDAbase implementation examples

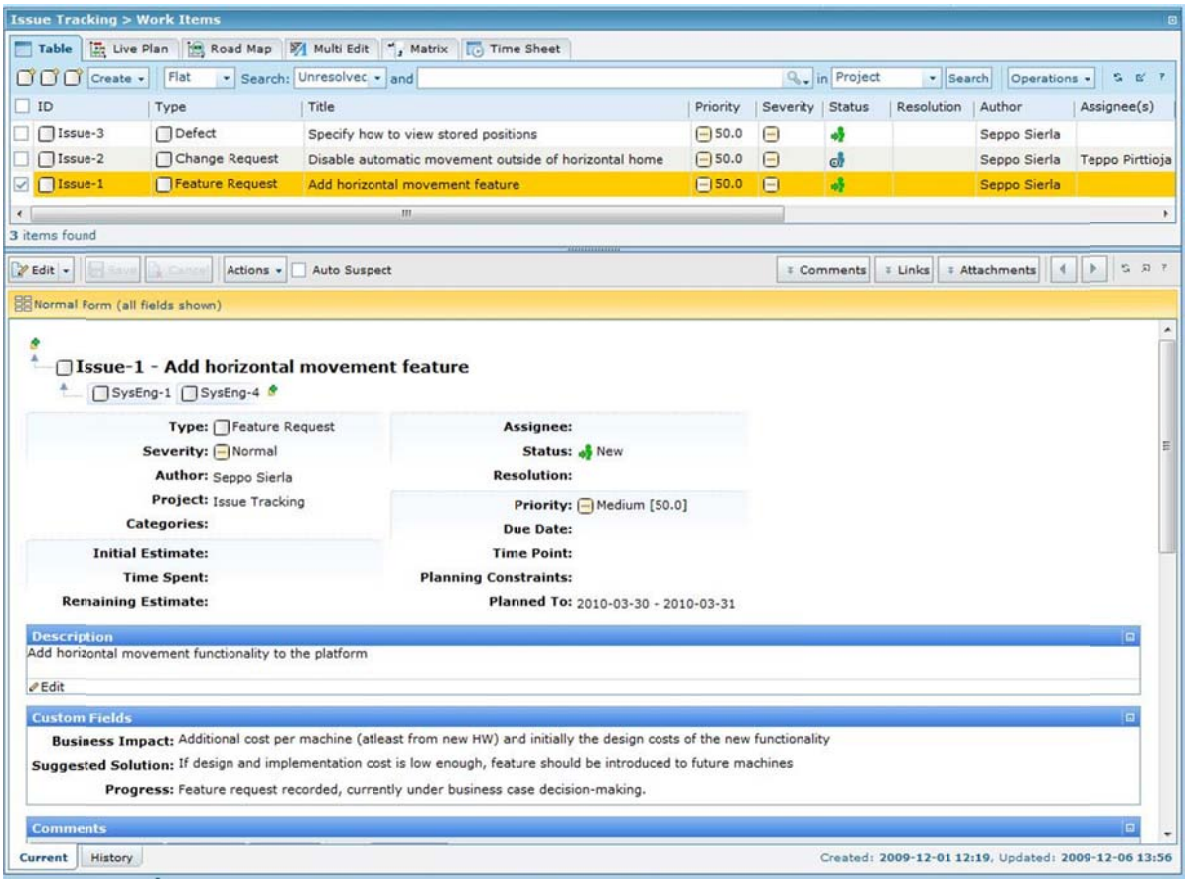


Figure 55. Feature request for the addition of horizontal movement.

As a response to this feature request, two *System Use Case* work items are created in the System Engineering project to support horizontal movement in manual and automatic modes of operation; see Figure 56 (manual movement) and Figure 57 (automatic movement).

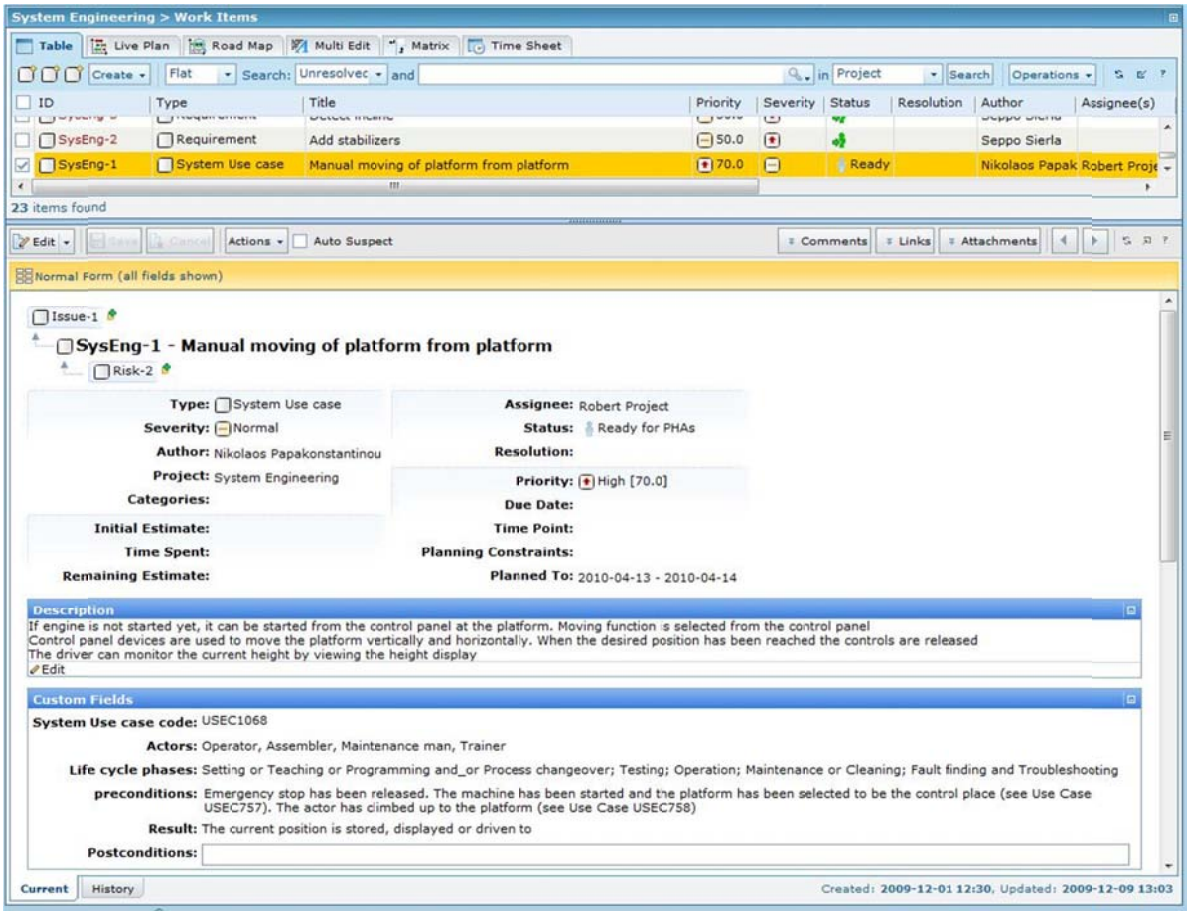


Figure 56. System use case WI for the manual movement of the platform.

4. IIDAbase implementation examples

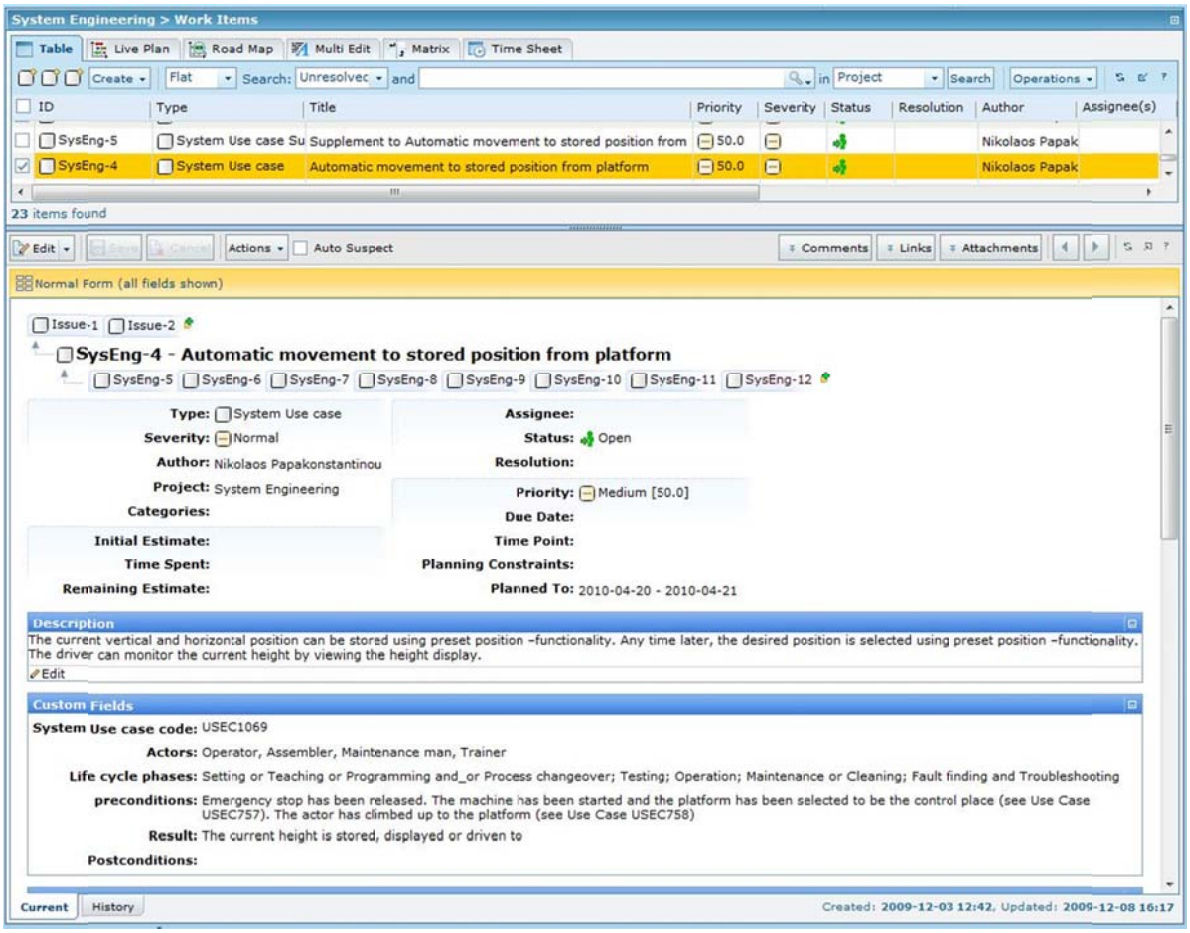


Figure 57. System use case WI for the automatic movement of the platform.

After this step, and in accordance with the KOTOTU safety process, a *Risk assessment* work item can be created as a request by the safety engineer for an early risk assessment of the system use cases that are part of the project so far; see Figure 58. The risk assessment method that can be applied at this stage of system development is the Preliminary Hazard Analysis (PHA); the estimation of the risk levels of the potential hazards is defined to be done according to the IEC 62061 standard.

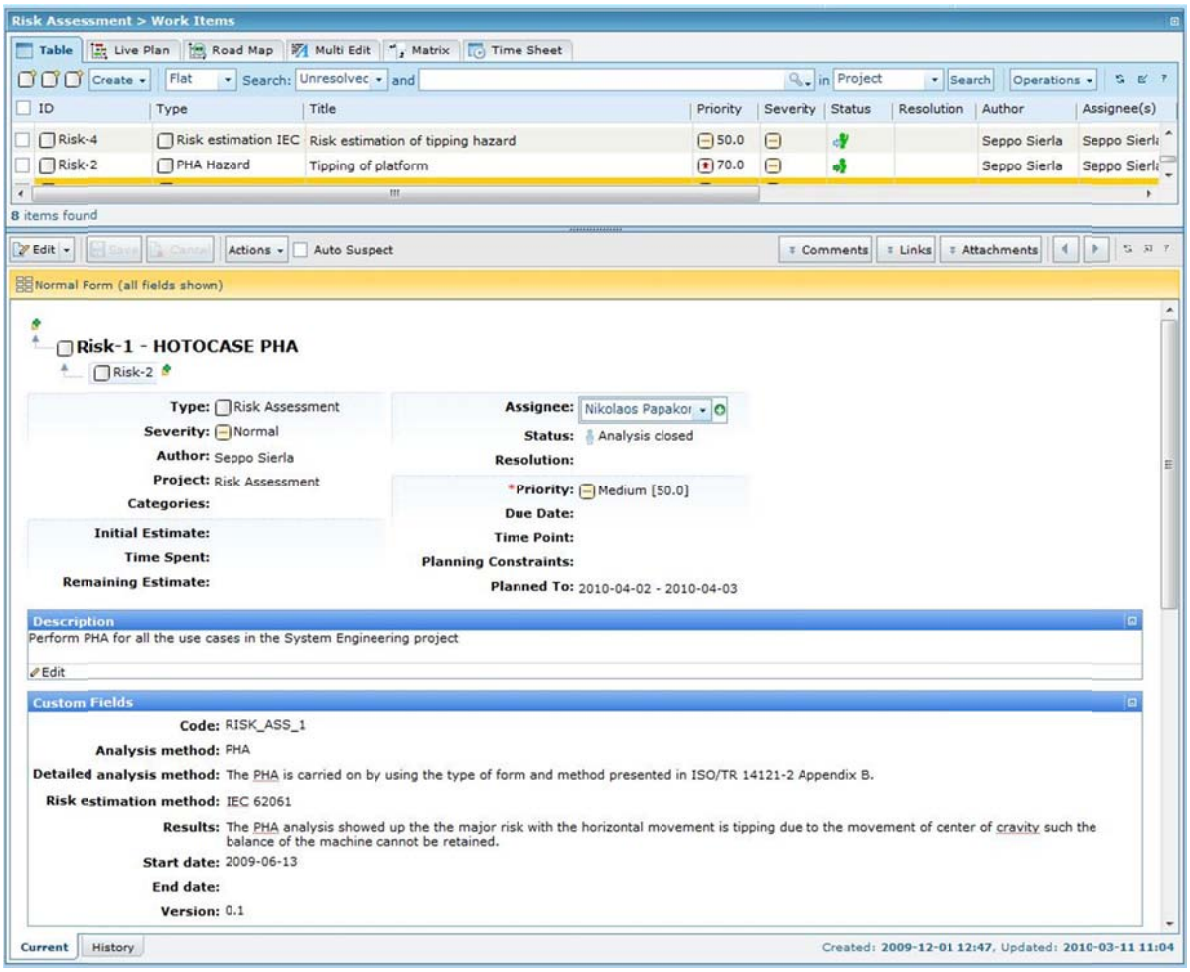


Figure 58. Risk assessment using the Preliminary Hazard Analysis method (note that the Results field is empty prior to performing the risk assessment).

A hazard is identified related to the system use case describing the manual movement of the platform. If the machine is placed on an inclined surface then there is a significant possibility of the horizontal movement leading it out of balance and thus to it tipping over. A *PHA hazard* (Figure 59) and a *Risk estimation* (Figure 60) work item are created to describe and estimate this hazard. The recommended safety measure proposed to avoid this hazard is the addition of stabilizers to the machine.

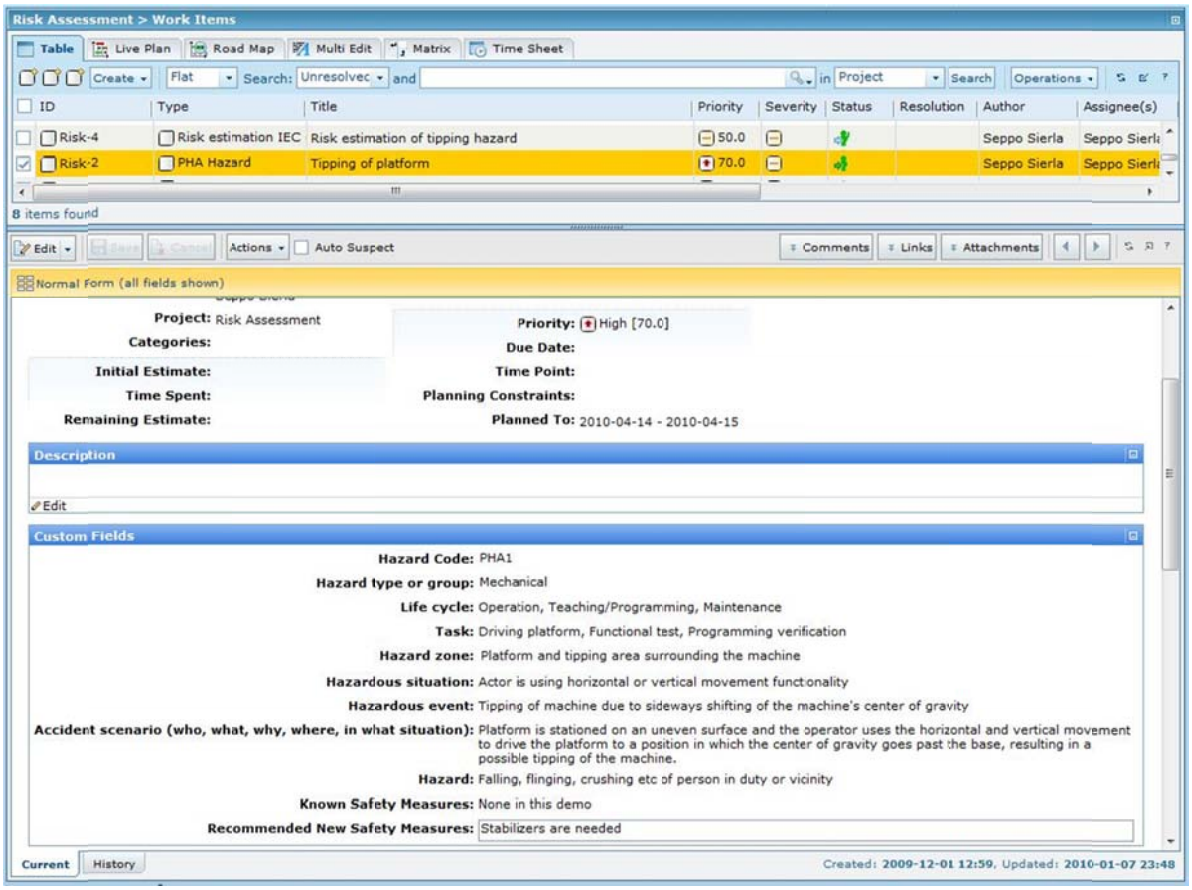


Figure 59. Tipping hazard identified using the Preliminary Hazard Analysis method.

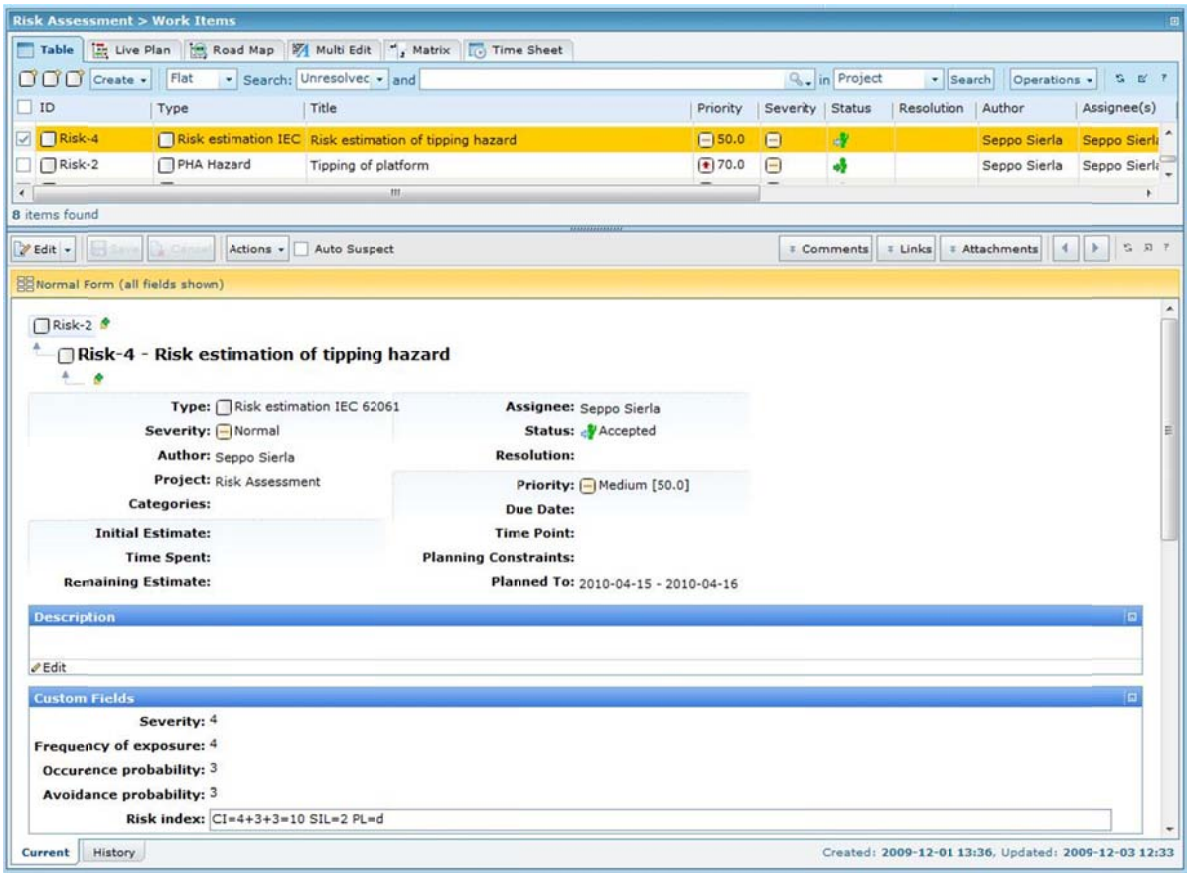


Figure 60. Risk estimation of tipping hazard using the IEC 62061 standard.

A *Risk evaluation* work item is now created by the system engineer to evaluate this risk and the proposed safety measures by the safety engineer; see Figure 61. The proposed safety measures are accepted and two *Requirement* work items for the implementation of the related safety measure, the 'Detect incline' and the 'Add stabilizers', are created and the risk assessment completed.

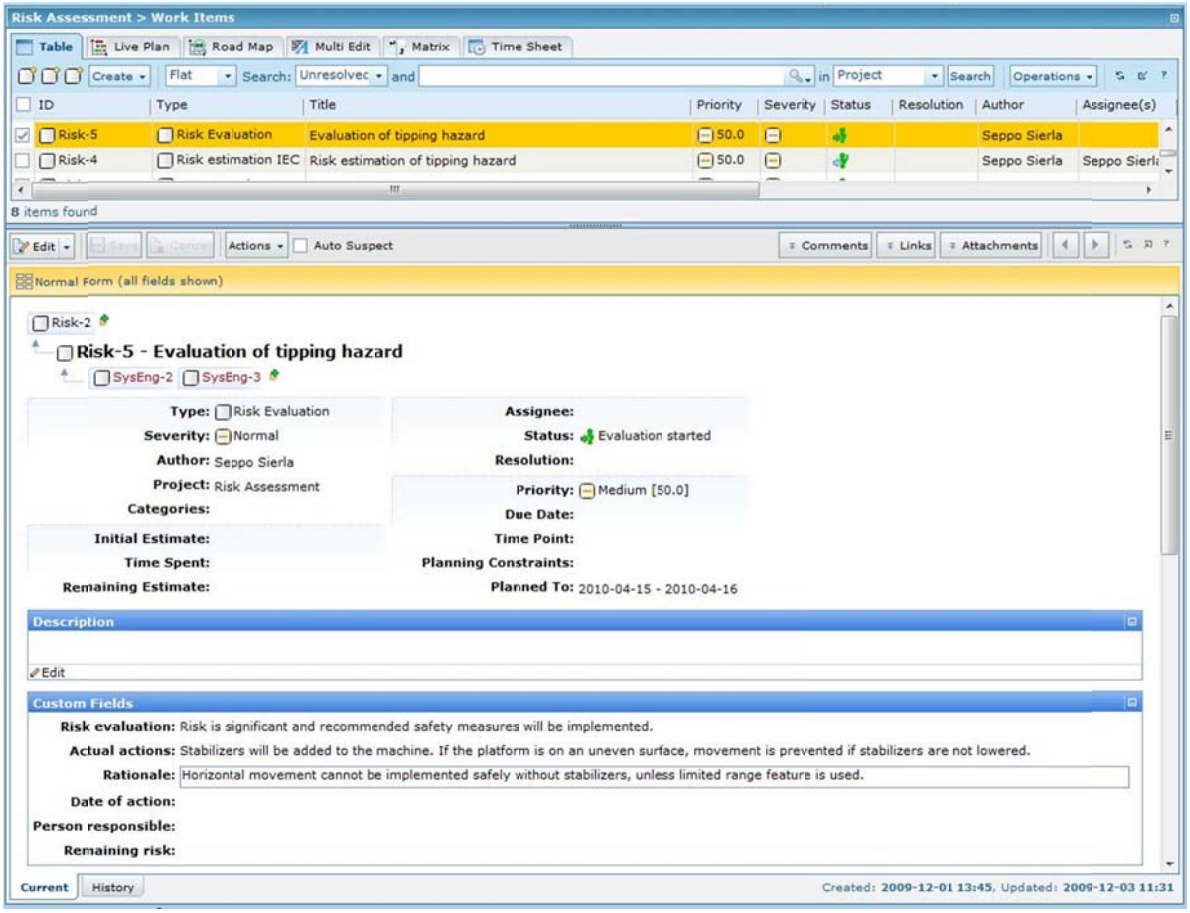


Figure 61. Risk evaluation of the tipping hazard.

Following the KOTOTU safety process, the system design can continue by extending the system use case describing the automatic movement to a stored position with the creation of a *System use case supplement* work item containing additional safety-related information (Figure 62), a *System function* (Figure 63) and *Use case acts* (Figure 64) work items.

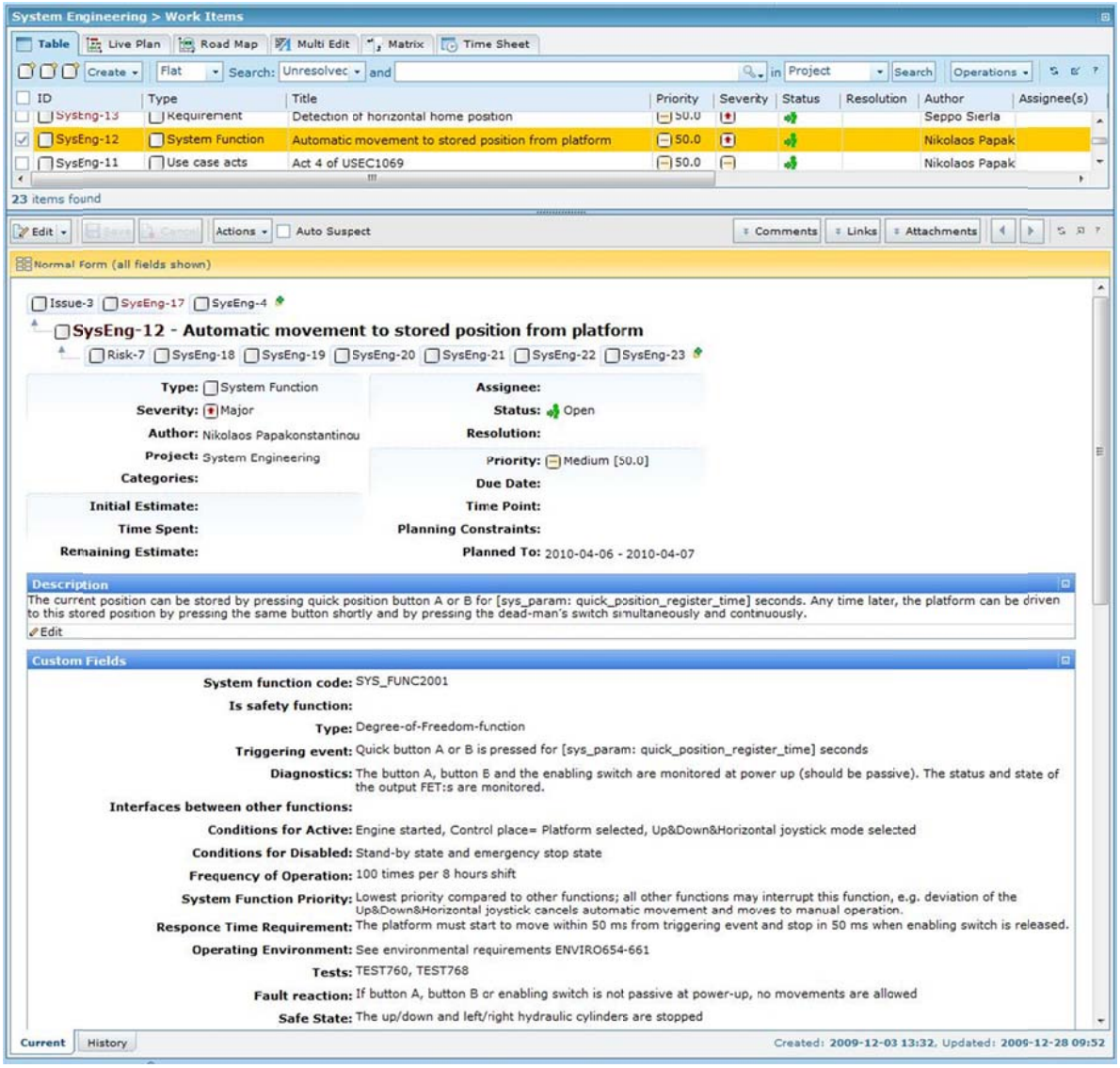


Figure 62. System function of the automatic movement of the platform.

4. IIDAbase implementation examples

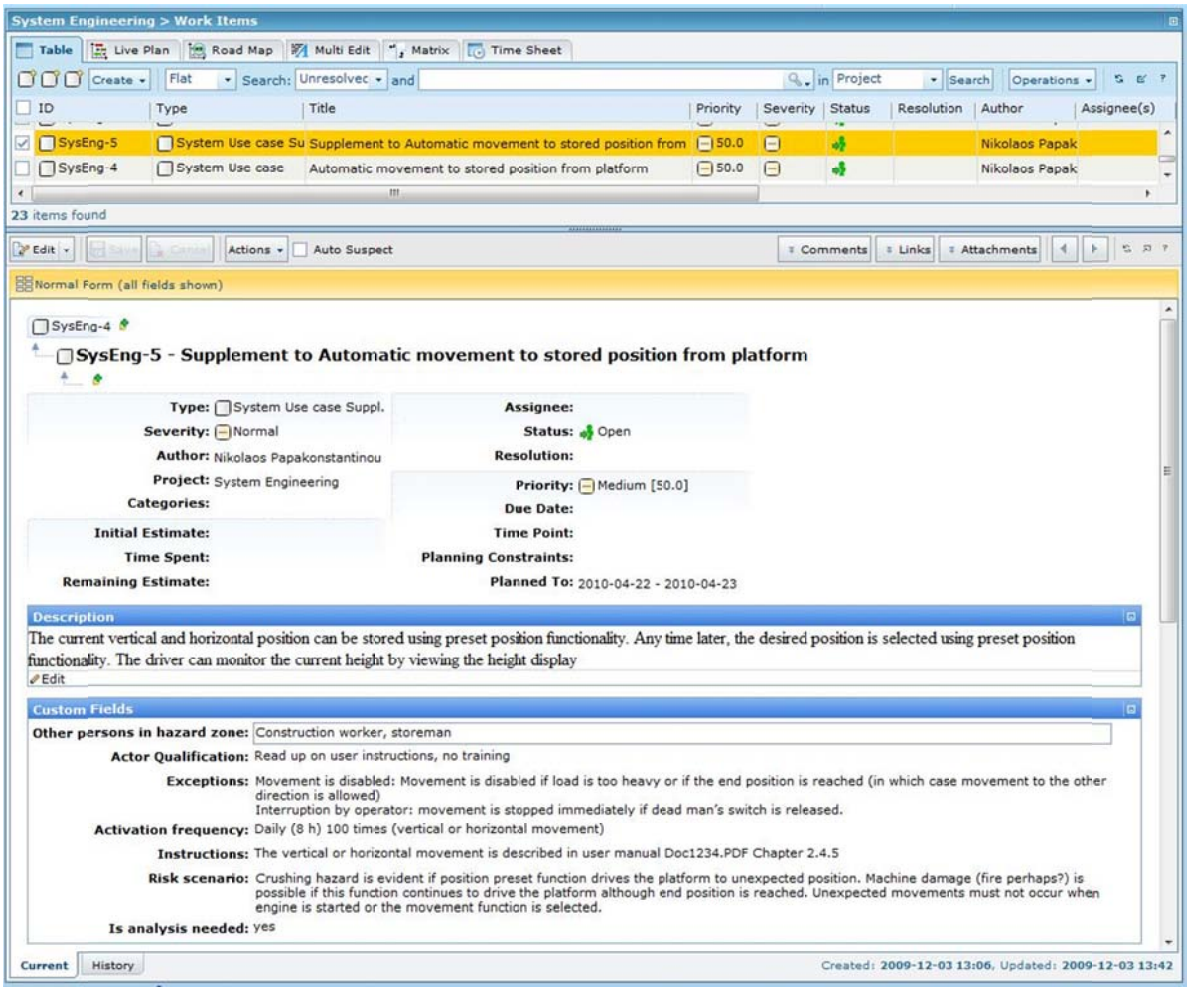


Figure 63. Automatic movement system use case safety supplement.

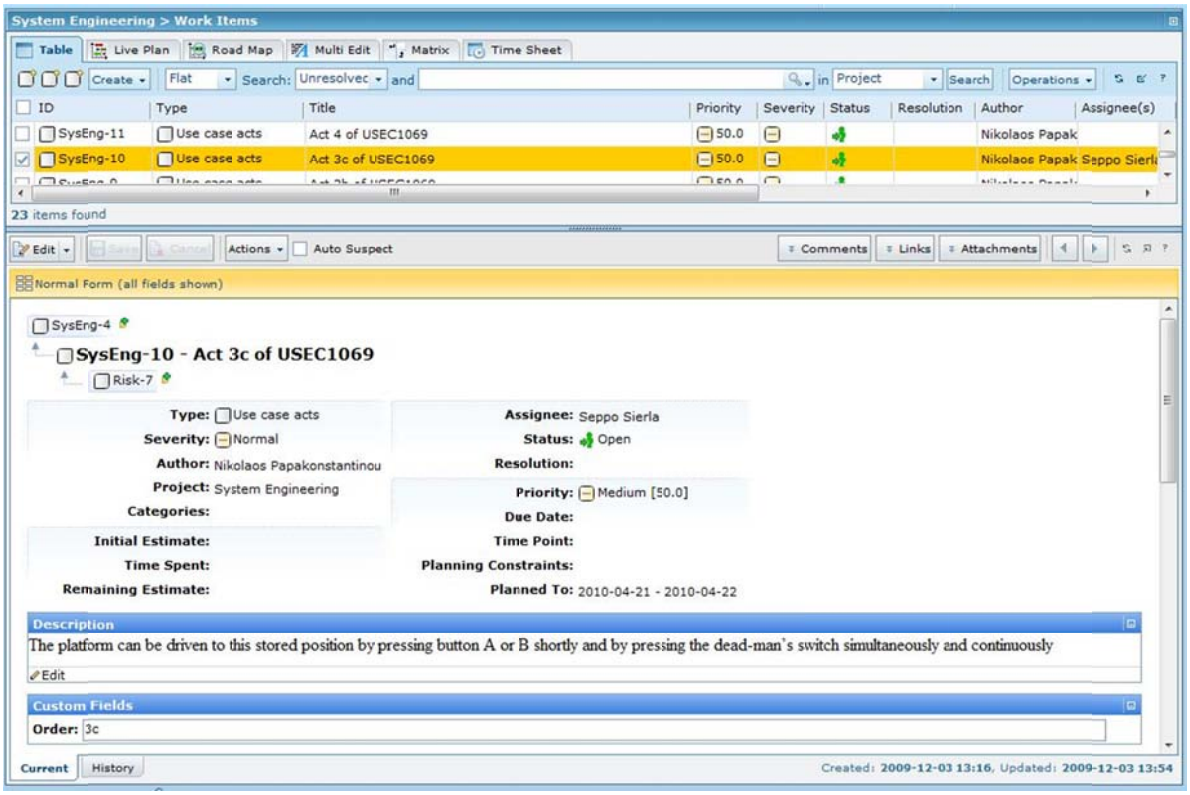


Figure 64. Use case act: part of the use case for automatic movement of the platform.

A second *Risk Assessment* work item is now created, as the source system engineering information is present to support the Use Case Safety Analysis (UCSA) risk assessment method. The safety engineer should go through all the system use cases present in the project and for each one she should iterate through the use case acts and try to see if their descriptions combined with each one of a set of safety guide words (like "too close", "too early") can lead to the identification of a potential hazard. In this case, the safety engineer identifies a hazard when she combines the description of use case act 3c, "The platform can be driven to this stored position by pressing button A or B shortly and by pressing the dead-man's switch simultaneously and continuously" with the guide word "too close". A *UCSA hazard* work item (Figure 65) is created for the hazard that describes this using the automatic movement to a stored position feature; the platform can be driven to a position that will cause it to collide with an object in its environment. A *Risk estimation* work item can then be created to estimate this hazard using the IEC 62061 standard, and the safety engineer can propose the new safety measure of the addition of an obstacle avoidance system for the machine.

4. IIDAbase implementation examples

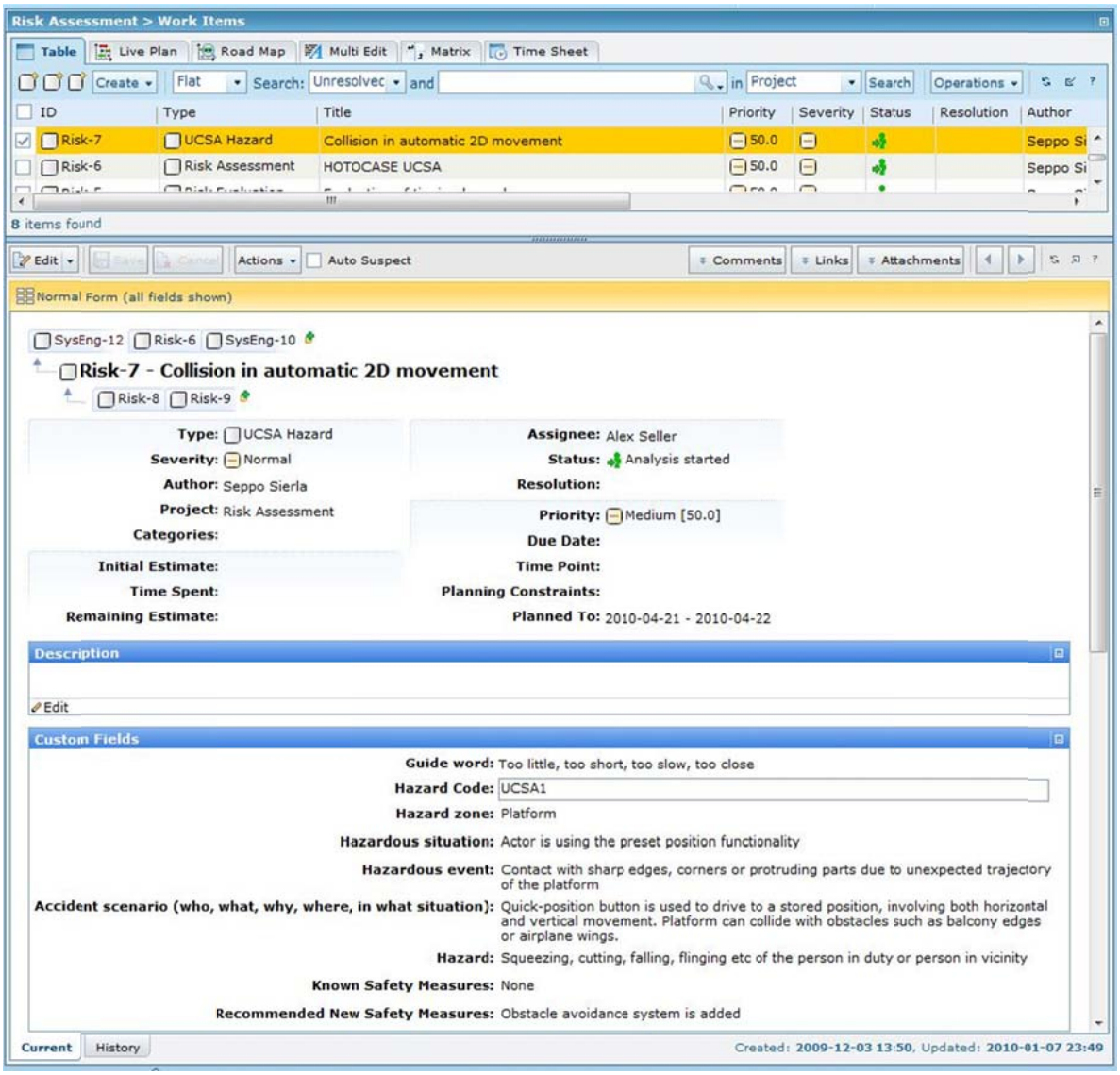


Figure 65. Use Case Safety Analysis hazard describing the danger of a collision in the automatic movement of the platform.

The system engineer reviews the work items created by the safety engineer and the proposed new safety measure, but in his *Risk evaluation* (Figure 66) work item he rejects the measure as too expensive and decides to create an *Issue* artefact, i.e., a *Change request* (Figure 67) work item describing the prohibition of automatic movement if the platform is out of its horizontal home position. He also creates three new *Requirement* work items on the prevention of automatic movement outside the horizontal home position, the limits of the horizontal home position and the display of the home position status to the machine's user.

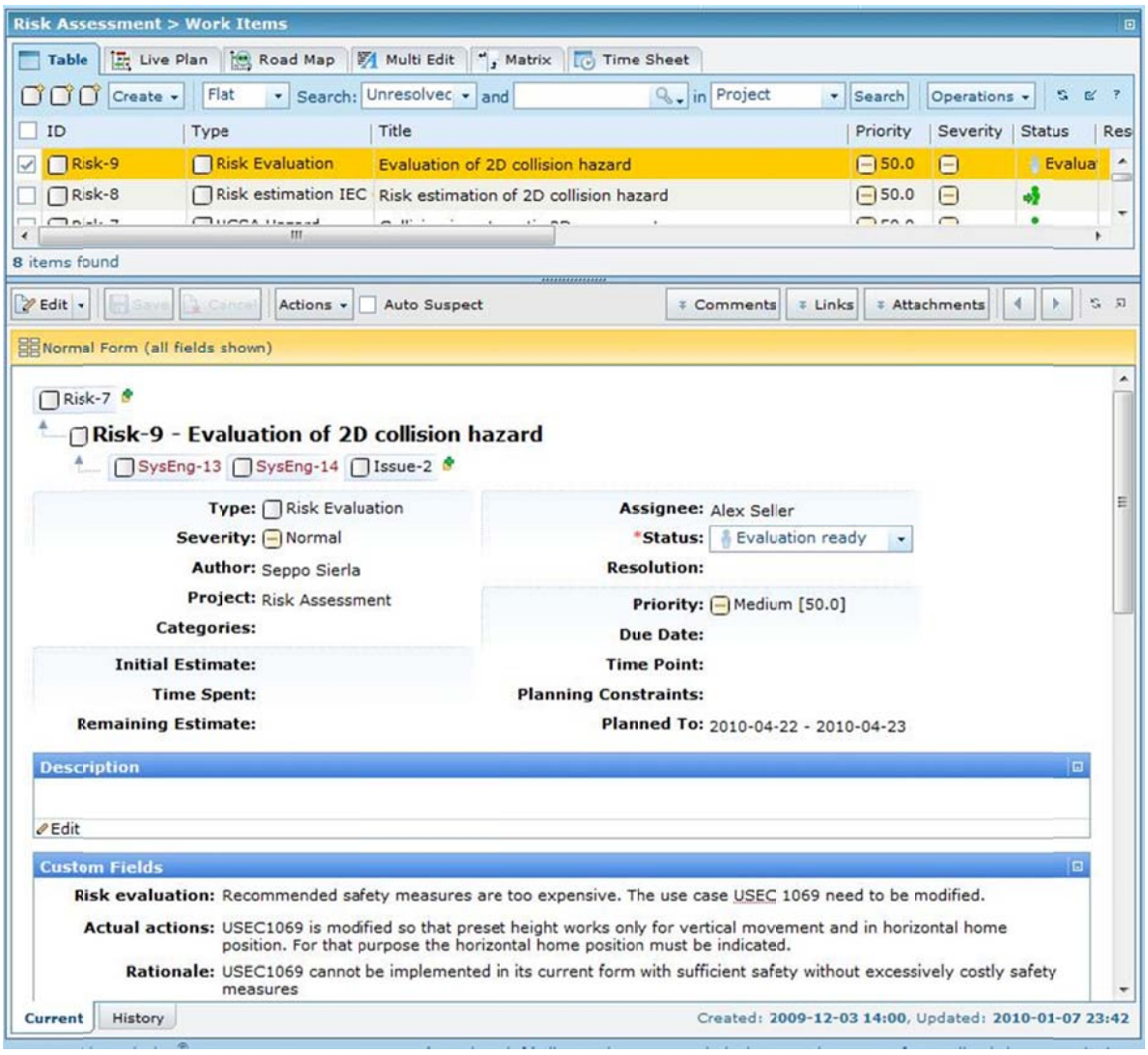


Figure 66. Risk Evaluation for a 2D collision hazard in automatic mode.

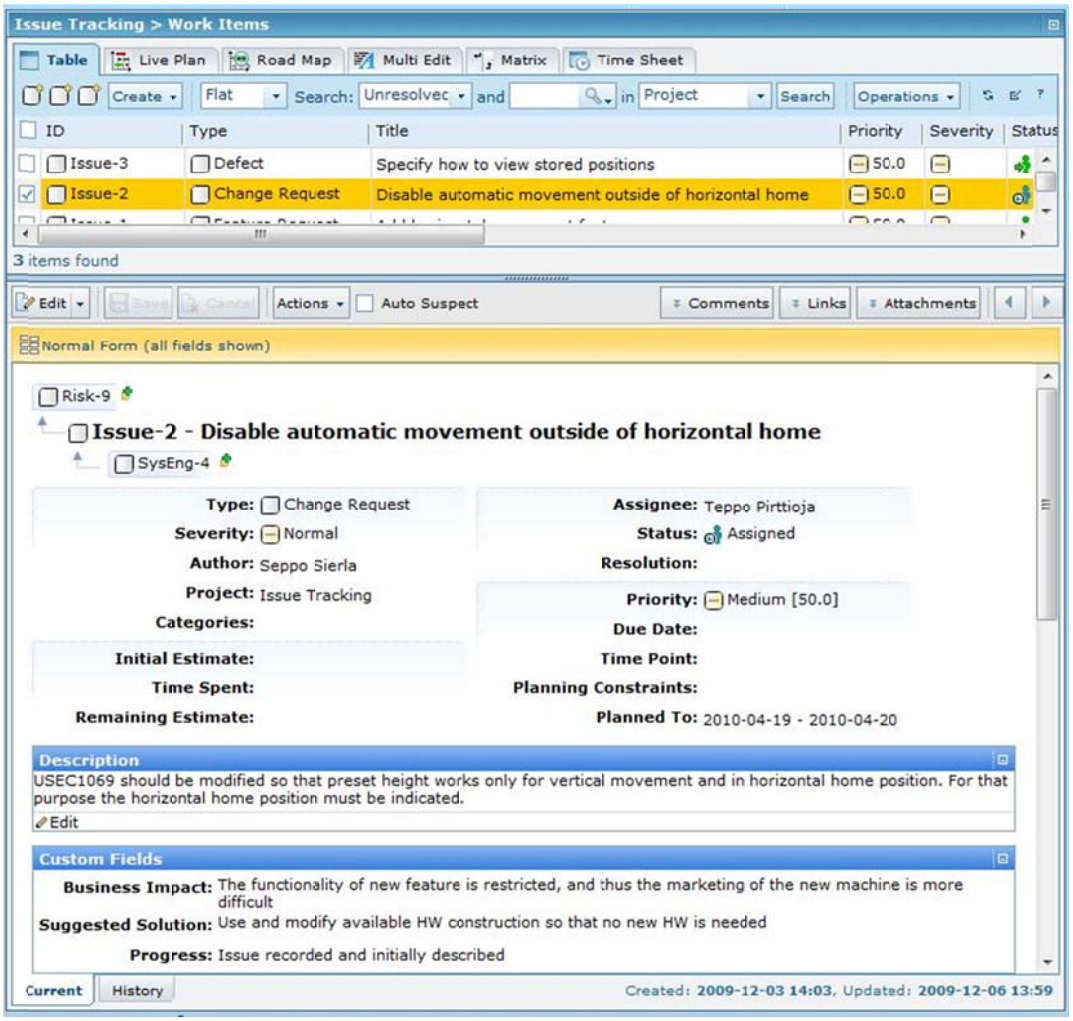


Figure 67. Change request to change the system use case describing the automatic movement of the platform.

4.2.2 Discussion

The demonstration proved that the workflow and data models, as presented through an Application Lifecycle Management tool such as Polarion, can help the integration of system development with safety risk assessments. By following the KOTOTU safety process and thus performing risk assessments starting with the use of methods that can be applied with early system design artefacts as sources, potential hazards can be spotted early and the amount of rework due to system redesign caused by safety concerns can be reduced.

The safety engineering workflow presented in Section 4.2.1 above was tested by a group of engineers from industrial companies in a workshop. The concept and the systematic flow of work were well accepted, but the workshop participants called for improvements to the tool user interface to illustrate better the safety process and to hide the complexity of the model. This motivates the building of an IIDAbase profile on top of Polarion ALM as suggested in Section 4.1.4.

5. Automatic creation of documents

5.1 Introduction

The current prevalent approach of using word processing and spreadsheet tools in systems engineering makes the information prone to becoming duplicated, inconsistent and obsolete. With IIDAbase, the centralized data repository can support the automated creation of documents based on demand. For example, the system requirements specification and the relevant parts of the technical file, which are required by the Machinery Directive, can be generated from the database.

One of the TIKOSU project tasks was to study the possibilities of creating documents from IIDAbase. The vision was to generate documents similar to the current style of word processing documents with free-flow text but such that the artefacts data are not written manually into the document but are fetched from the IIDAbase into the text flow.

5.2 Possible tools

The main aim was to find documenting tools that provided database connectivity. As IIDAbase was also demonstrated on the XML-based Polarion ALM tool, the facilities of the Polarion tool to generate documentation were also surveyed.

5.2.1 MS Word

Microsoft Word provides database connectivity through an ODBC connection. With the help of the ODBC connection, any database with an ODBC driver can be accessed. The access uses a special field code script that allows a SQL query to be issued to the database. For example, the following field code script (Program 3) produces Table 4.

Program 3. Example of a SQL query from within MS Word.

```
{ DATABASE  \d "C:\\Documents and Settings\\user\\My Documents\\My Data
Sources\\kotocase4 ib_requirements.odc" \c "Provider=MSDASQL.1;Persist
Security Info=True;Extended
Properties=\"DATABASE=kotocase4;DSN=RMMSSQL;OPTION=0;PWD=XXXXXXXX;PORT=XXXX;SE
RVER=XXX.XXX.XXX.XXX;UID=root;\";Initial Catalog=kotocase4" \s "SELECT `code`
AS `Code`, `name` AS `Name`, `explanation` AS `Description`, `type` AS `Type`
FROM `ib_parts`" \l "23" \b "63" \h }
```

Table 4. An example table, the data of which is retrieved from IIDAbase (in this case, a MySQL database).

Code	Name	Description	Type
PART_12211-1221	Chassis_controller	Controller on the chassis that is mainly used for controlling the hydraulics and the engine	Controller
PART_12312-2121	Platform_controller	Controller on the platform that is mainly used for inputting user interface devices but also for displaying the height of the platform on a small display	Controller
PART_33112-2211	Up_down_transfer_joystick	The up/down joystick controller; also used for transferring the machine	Discr_UI_ctrl
PART_31231-1211	Platform_position_encoder	Platform position encoder	Sensor

The database connectivity facility of MS Word works well with simple tables like Table 4, but as soon as the tables become big and the layout of the table needs formatting, e.g., by adjusting the widths of the columns and changing the fonts, MS Word does not retain the changes but reapplies the original column widths and font types as soon as the field codes are updated.

Furthermore, only tables or lists are practical means of rendering information from the database. Random layout of data inside the text is difficult and in some cases impossible. Hence, MS Word is not considered the optimum tool for generating complex documents out of the database.

5.2.2 MS Access reports

MS Access was used as a demonstration and test tool in the TIKOSU project. MS Access only worked as a front-end to the MySQL implementation of the IIDAbase model. The MySQL tables were linked to MS Access using the MS Access external data source facility. As all the MySQL data were thus visible in MS Access, it was relevant to consider the possibility of using the MS Access reporting facility to generate documents.

The report generation facility of the database tools are typically such that for a blank report template, the source of the data is selected and the data fields from the selected table are dropped into the blank sheet. Additional text can be added and all text can be formatted, but, e.g., single words within a

chapter of text cannot be formatted differently to the format of the chapter. Figure 68 illustrates the design phase of such a report. Figure 69 displays the output of the design in Figure 68.

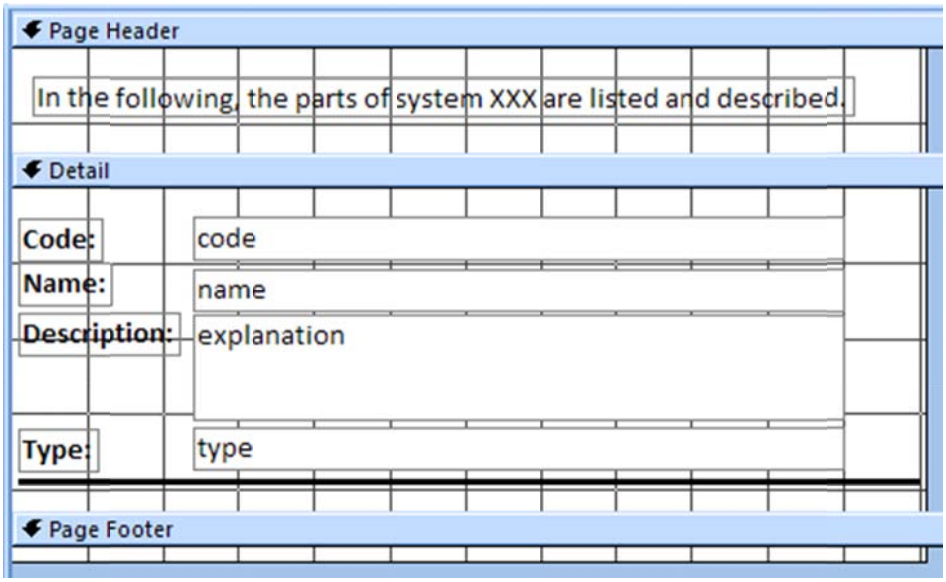
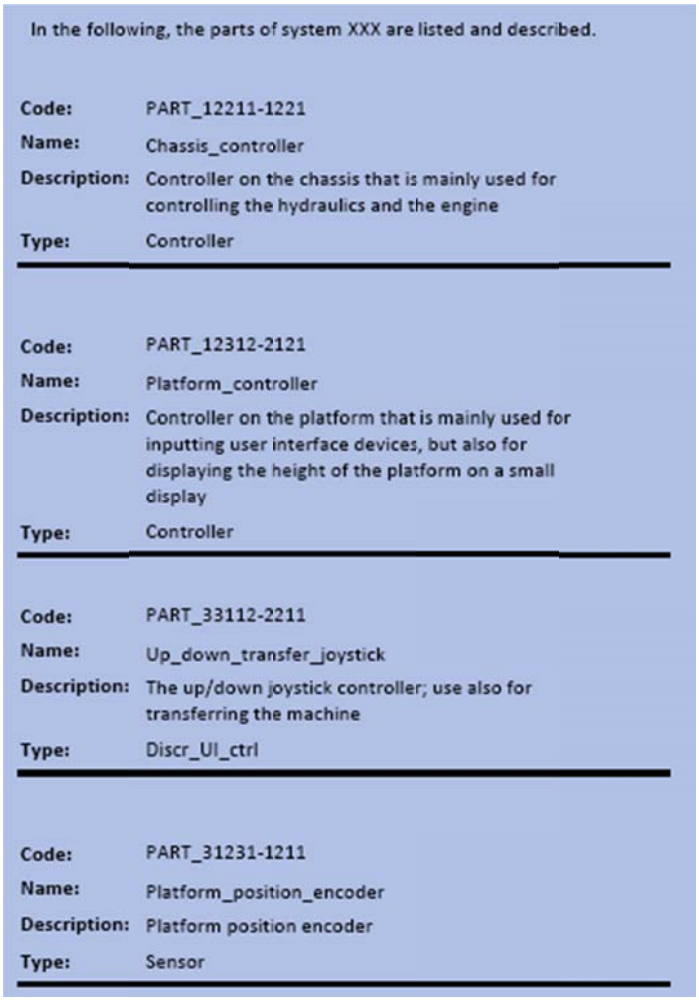


Figure 68. Designing a report in MS Access.

The output format can be selected from the following: PDF, XPS, RTF, TXT, XML and HTML. Of these, only PDF and XPS preserve the layout as designed in MS Access, e.g., the separating line below the 'type' field is only visible in PDF and XPS outputs.

It is not possible to create free-flowing documents with the MS Access report generator, e.g., data fields cannot be taken from arbitrary data tables if there is no association between the source table that has already been defined and the new one. Furthermore, it is not possible to add pieces of descriptive texts freely to the documents, e.g., in Figure 68 it is not possible to write a chapter of text after the Detail section and start to render data from another table with no association to the first one. Hence, the MS Access report generation is not good for free-flow documents and can only be used to show a compound part of the data in a structured format.


In the following, the parts of system XXX are listed and described.

Code: PART_12211-1221
Name: Chassis_controller
Description: Controller on the chassis that is mainly used for controlling the hydraulics and the engine
Type: Controller

Code: PART_12312-2121
Name: Platform_controller
Description: Controller on the platform that is mainly used for inputting user interface devices, but also for displaying the height of the platform on a small display
Type: Controller

Code: PART_33112-2211
Name: Up_down_transfer_joystick
Description: The up/down joystick controller; use also for transferring the machine
Type: Discr_UI_ctrl

Code: PART_31231-1211
Name: Platform_position_encoder
Description: Platform position encoder
Type: Sensor

Figure 69. Output from the report template presented in Figure 68.

5.2.3 MS Excel

MS Excel also provides database connectivity. MS Excel is only suitable for displaying lists and tables as shown below in Table 5.

Table 5. A table created by MS Excel by retrieving the data from IIDAbase.

Code	Name	Description	Type
PART_12211-1221	Chassis_controller	Controller on the chassis that is mainly used for controlling the hydraulics and the engine	Controller
PART_12312-2121	Platform_controller	Controller on the platform that is mainly used for inputting user interface devices, but also for displaying the height of the platform on a small display	Controller
PART_33112-2211	Up_down_transfer_joystick	The up/down joystick controller; use also for transferring the machine	Discr_UI_ctrl
PART_31231-1211	Platform_position_encoder	Platform position encoder	Sensor

MS Excel allows formatting of the visual look of the table but does not maintain the width of a column if it is changed. Free-flow documents cannot be made, but MS Word can link to an Excel file that con-

nects to IIDAbase. In this case, however, the data are not updated in the Word document if the Excel document is not first opened and refreshed.

5.2.4 Others

MS Publisher provides database connectivity but is restricted to creating lists and mail merge.

Jaspersoft's *iReport* is similar to the MS Access report generator and has the same limitations.

OpenOffice provides database connectivity but is clumsy and only suitable for creating tables and lists within a text. Oracle Report Builder, a similar facility to iReport and MS Access report generator, is provided with a special plug-in.

5.2.5 Altova StyleVision

The XSLT (Extensible Stylesheet Language Transformations) specification defines a way to create human-readable output, like HTML, from XML files. Altova's StyleVision provides a visual style sheet designer to create an XSLT file. StyleVision supports relational databases by first creating a XSD schema from the database.

In the visual style sheet designer, special tags can be used to denote the particular pieces of information to be rendered into the output document from the database, e.g., the following excerpt (Figure 70) from a Systems Requirements Specification template produces the output shown in Figure 71.

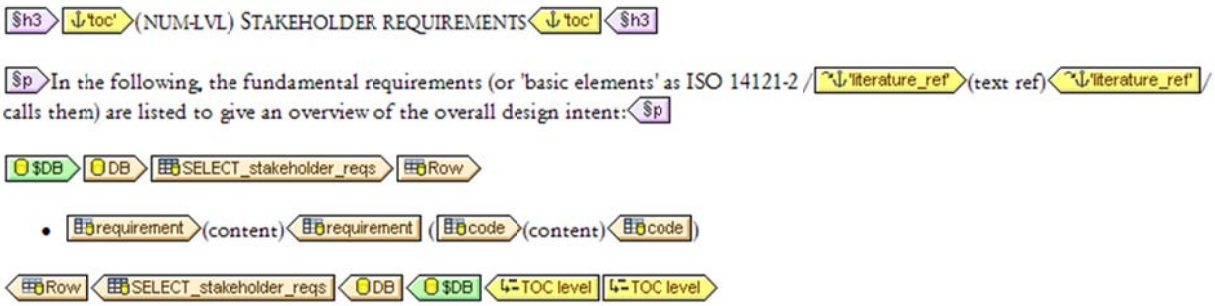


Figure 70. An example of the StyleVision visual designer usage.

2.1.1 STAKEHOLDER REQUIREMENTS

In the following, the fundamental requirements (or 'basic elements' as ISO 14121-2 /2/ calls them) are listed to give an overview of the overall design intent:

- Machinery directive 2006/42/EC must be followed (SAFETY22222)
- A mobile elevating platform hoisted by scissors for buliding construction and for warehouse use (GENERAL11111)

Figure 71. Output rendered by the tag script in Figure 70.

As can be seen in Figure 71, free formatting of data (in this case a bulleted list with additional metadata in parentheses) can be used. Furthermore, the text can continue to flow freely as in a word pro-

cessing tool and again a piece of data can be imported from another table in the database. Hence, any data from the database can be incorporated into any part of the document.

StyleVision also allows conditional rendering of data, e.g., the tag script in Figure 72 produces the output in Figure 73.

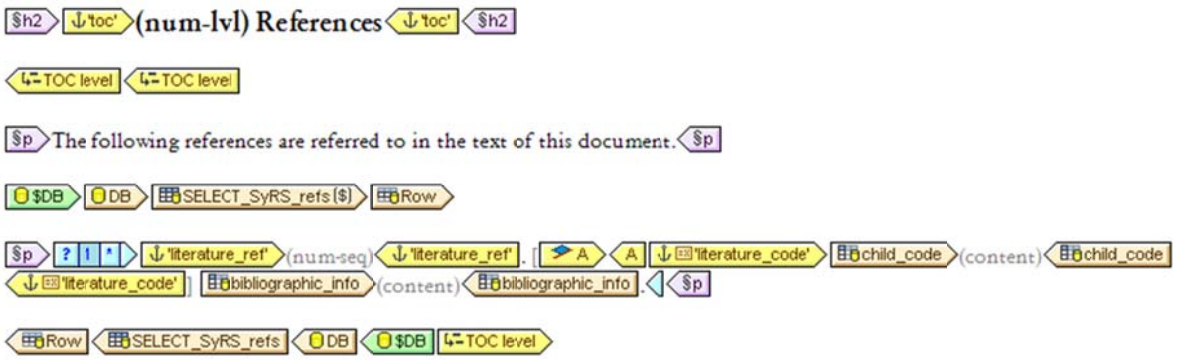


Figure 72. Another example of StyleVision visual designer usage.

1.3 References

The following references are referred to in the text of this document.

1. [STAN_0002] ISO 14121-1, 2007. Safety of machinery – Risk assessment – Part 1: Principles. 2007-09-01 edn. Geneva: International Organisation for Standardization.

2. [STAN_0003] ISO/TR 14121-2, 2007. Safety of machinery – Risk assessment – Part 2: Practical guidance and examples of methods. 2007-12-03 edn. Geneva: International Organisation for Standardization.

3. [STAN_0001] SFS-EN 280 + A2, 2009. Mobile elevating work platforms. Design calculations. Stability criteria. Construction. Safety. Examinations and tests. Helsinki: Suomen Standardoimisliitto SFS. 146 pages..

Figure 73. Output rendered by the tag script in Figure 72.

In Figure 73, the last reference is shown in grey: the question mark tag in Figure 72 is a conditional tag that is configured to render the text in grey if the suspect flag (i.e., the traceability flag) is TRUE. In this case, the last standard in the list is updated and hence the trace becomes suspect and needs to be checked by the system engineer.

Furthermore, StyleVision allows for flexible layout of data and navigation in the document using the artefact database codes as links, as shown in Figure 74 and Figure 75.

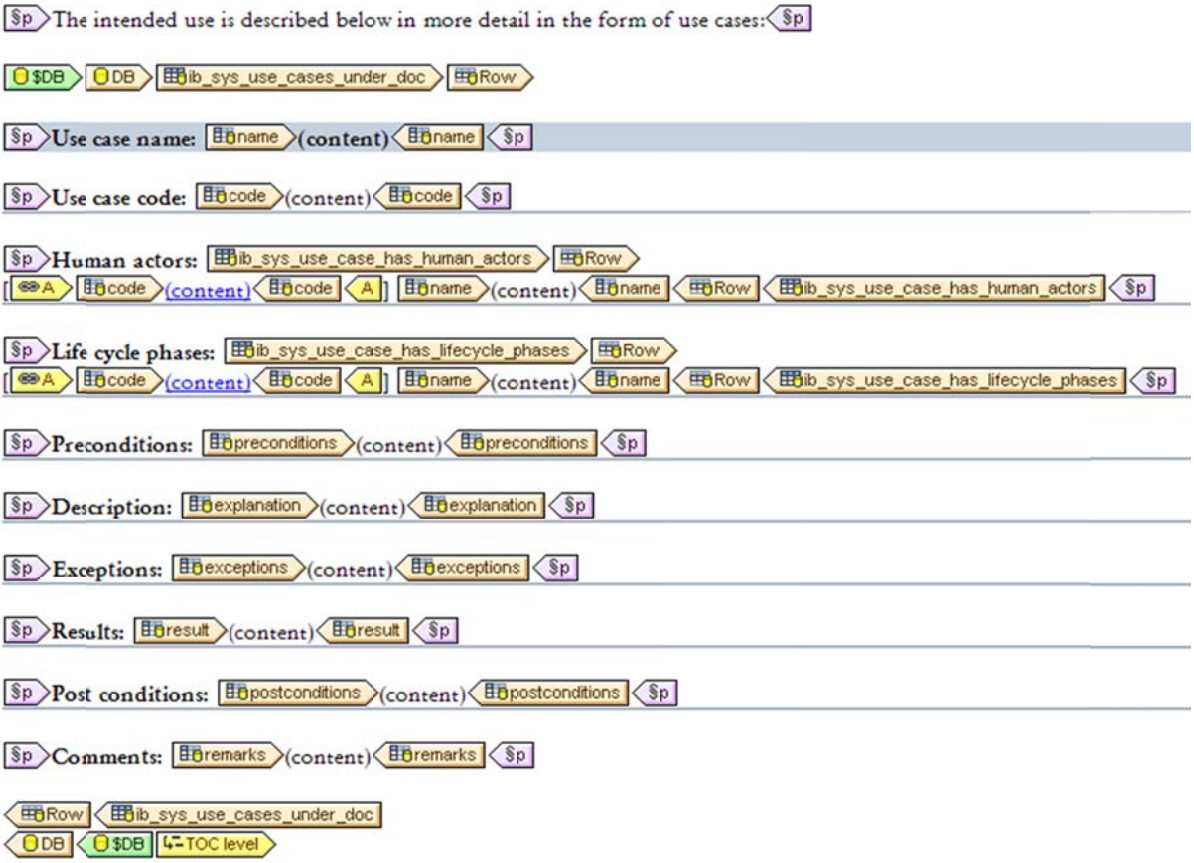
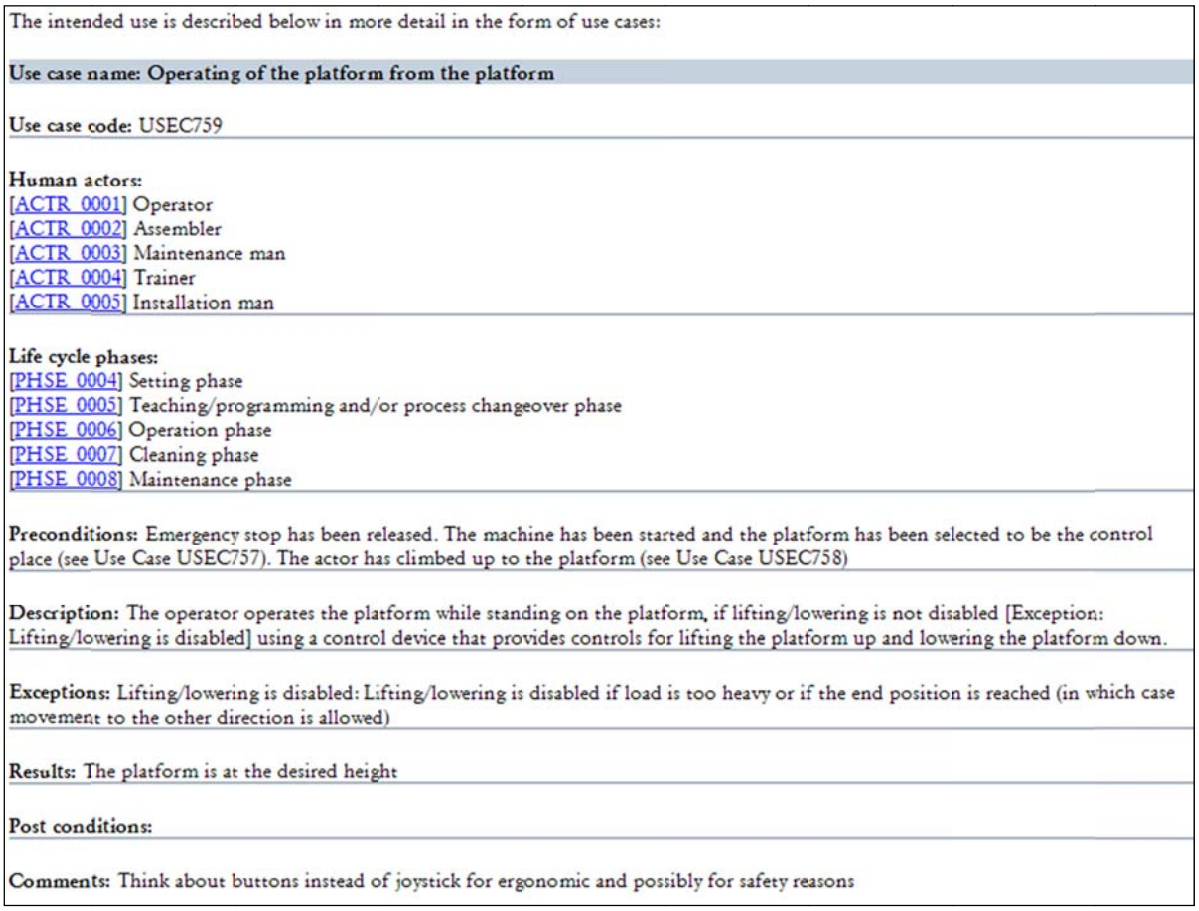


Figure 74. Another example of StyleVision visual designer usage.



The intended use is described below in more detail in the form of use cases:

Use case name: Operating of the platform from the platform

Use case code: USEC759

Human actors:
[ACTR_0001] Operator
[ACTR_0002] Assembler
[ACTR_0003] Maintenance man
[ACTR_0004] Trainer
[ACTR_0005] Installation man

Life cycle phases:
[PHSE_0004] Setting phase
[PHSE_0005] Teaching/programming and/or process changeover phase
[PHSE_0006] Operation phase
[PHSE_0007] Cleaning phase
[PHSE_0008] Maintenance phase

Preconditions: Emergency stop has been released. The machine has been started and the platform has been selected to be the control place (see Use Case USEC757). The actor has climbed up to the platform (see Use Case USEC758)

Description: The operator operates the platform while standing on the platform, if lifting/lowering is not disabled [Exception: Lifting/lowering is disabled] using a control device that provides controls for lifting the platform up and lowering the platform down.

Exceptions: Lifting/lowering is disabled: Lifting/lowering is disabled if load is too heavy or if the end position is reached (in which case movement to the other direction is allowed)

Results: The platform is at the desired height

Post conditions:

Comments: Think about buttons instead of joystick for ergonomic and possibly for safety reasons

Figure 75. Output rendered by the tag script in Figure 74.

By pressing the links, like *ACTR_0001*, in the document in Figure 75 above, the focus jumps to the position in the document at which the particular artefact is presented.

The text can be output in HTML, PDF, RTF or DOC formats as well as in the Altova's proprietary Authentic format, which is similar to HTML, with the exception that Authentic allows data to be written to the database through the generated document. This requires a special Authentic client (both stand-alone and browser plug-ins are supported).

StyleVision thus matches the needs for free-flow document generation.

5.2.6 Polarion document generation capabilities

The Polarion ALM tool (version 2010) provides three possibilities for document generation: the Modules concept, Wiki pages and Word document interfacing. None of these facilities provides the necessary mixed usage of artefacts in a single document. The new version (2011) of Polarion has a completely new concept related to interfacing with free-flow documents. Much of the Polarion basic functionality in the 2011 version is also built to be interactive and editable through document-type views, called LiveDocs, through the Polarion user interface. Polarion 2011 was not available for testing during the project, but the manuals of the 2011 version presented a possibility of exporting hierarchical

trees of the work items into the live documents. The exporting feature allows users to control which of the fields are rendered to a document, how the work items are sorted in the lists and how many levels of child items are shown in the lists. Users can edit the order and hierarchy of the work items in the lists and tables in the document view, and these modifications are mirrored in the Polarion repository that stores and manages the work items. Traces to other work items can also be browsed.

The 2011 version also has a feature in which the document that was originally generated and edited with MS Word can be imported to Polarion to generate work items and their relations. The 2011 version also has a Word round-trip editing feature with which the work items created in Polarion can be sent in Word format to an external person for editing and imported back to Polarion when the updates are ready. The earlier version (version 2010) has similar but much more limited Word file export and import functionality. In version 2010, the document template is fixed and work items are rendered one after another as individual items, and the produced output as such is not that useful for document generation purposes.

Version 2011 of the LiveDocs facility seems to be a more flexibly editable version of the Module facility in version 2010. Without the ability to execute document generation tests with the Polarion version 2011, its LiveDocs feature seems to offer much of the functionality needed in the TIKOSU document generation context, including presenting multiple work item types in a single document. If Polarion is used to generate, modify and store the safety-related artefacts in work item form, the LiveDocs feature can be used to generate, e.g., parts of the technical file required by the Machinery Directive.

5.3 Other issues

5.3.1 IIDAbase Document concept model

The IIDAbase concept model for documents was presented in Figure 31 in Section 3.9.

The model in Figure 31 is only useful in cases in which the document is like a book with only simple text and figures. As soon as other artefacts need to be accommodated within the text, however, the model does not work, e.g., to produce the piece of the document in Figure 71, the paragraph should have a database relation to the stakeholder requirements. This would be easy to fix by adding the *Requirement* artefact to the model in Figure 31 and relating it to the *Paragraph* artefact, but the *Requirement* artefact is not the only piece of database information that may be needed in the document. Hence, practically all the artefacts should be linked to the *Paragraph* artefact in the *Document* model. Instead, it was decided to adhere to the concept of generating the documents such that the generic pieces of text reside in the StyleVision template (not in the *Chapter*/*Paragraph* artefacts) and the application-specific data are retrieved from the database by the StyleVision tool and put in appropriate places within the text as shown in Figure 70 to Figure 75. The StyleVision template is then introduced as a *Document* artefact (see Figure 76).

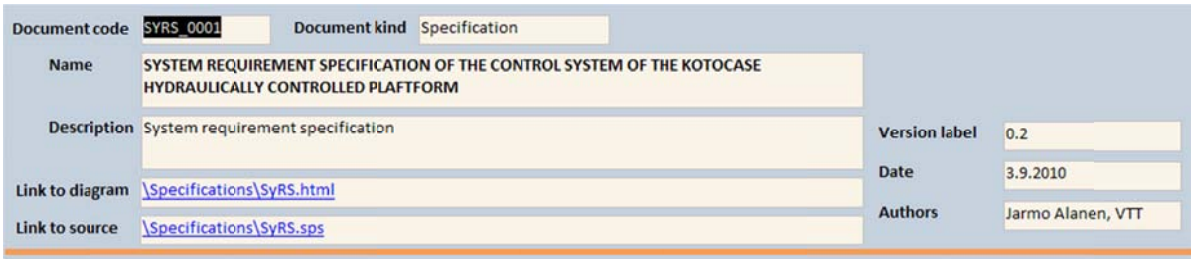


Figure 76. The StyleVision template introduced as a Document artefact.

The *Diagram* artefact can be used with the *Document* artefact as suggested in Figure 31, e.g., to associate a cover image with a document.

5.3.2 Formatted text

In many cases there is a need for formatted text and text with figures and tables etc. In practice, this means that some of the fields of an artefact need to support formatted text instead of plain text. A typical example is the description field of a *Requirement* artefact.

The RTF (Rich Text Format), HTML and Wiki formats are the practical candidates for formatted text. HTML was selected for both the formatted pieces of text and as the output format of the generated documents for the following reasons:

- It was easy to find an ActiveX control for the MS Access demonstrator tool that allows HTML text to be input into a database field. StyleVision, on the other hand, allows bypassing of the unnecessary rendering of HTML text again into HTML text and thus directly outputs the HTML text from a database field to the HTML file.
- With the RTF and MS Word file formats, the engineers are tempted to update the documents when they want to make changes to the artefacts shown in the document even though they should update the artefacts in their original place, i.e., in the IIDAbase repository (through a systems engineering tool). With HTML documents, it is very unlikely that the engineers will try to do this.

5.4 Discussion

In the project, a system requirements specification for a demonstration system was created with the StyleVision tool. It was noted that it is possible to make the generated documents look very similar to the conventional word processing documents with which the engineers are familiar. The possibility to embed mathematical equations into the text is still missing however. The equations must now be presented as images. This is true both in the case when the equations are written into the StyleVision template and in the case when the text is written via the ActiveX control in the MS Access tool. In fact, the problem with equations in HTML documents is a general one.

The Polarion ALM version 2011 tool, which was not available during the project, also seems to provide much of the reporting functionality striven for in the TIKOSU project. The optimum is of

course for the ALM or PLM tool used for the systems engineering to be capable of generating the documents from the artefacts it stores, in which case, no special report generator tool is needed.

As a conclusion, we can say that a typical word processing file is not the optimum means for storing, transferring or presenting systems engineering data. Instead, the use of a central repository to store the systems engineering information and presenting it through automatic generation of documents is encouraged. If word processing files are used due to their familiarity, e.g., to transfer requirements from customers to engineers, the word processing files must be well structured, and the transfer of requirements thereupon from engineers to other engineers should preferably occur using the central artefact repository tool.

6. Integration examples

The purpose of the IIDAbase model is not to replace all the systems engineering tools but to define the core set of artefacts that are needed in every systems engineering project of mobile machinery. In this sense, the set of core artefact types is homogenous whereas the off-the-shelf tools are heterogeneous. ISO AP233 was created to solve the integration problem of the tools, but the standard is still in the draft phase (and has been in preparation for years) and is massive to implement. The tool support is thus poor. The system engineering tools currently available do of course not support IIDAbase either. Hence, a remarkable effort is needed to integrate the tools into IIDAbase and each other via IIDAbase.

The basic model for tool integration is depicted in Figure 77.

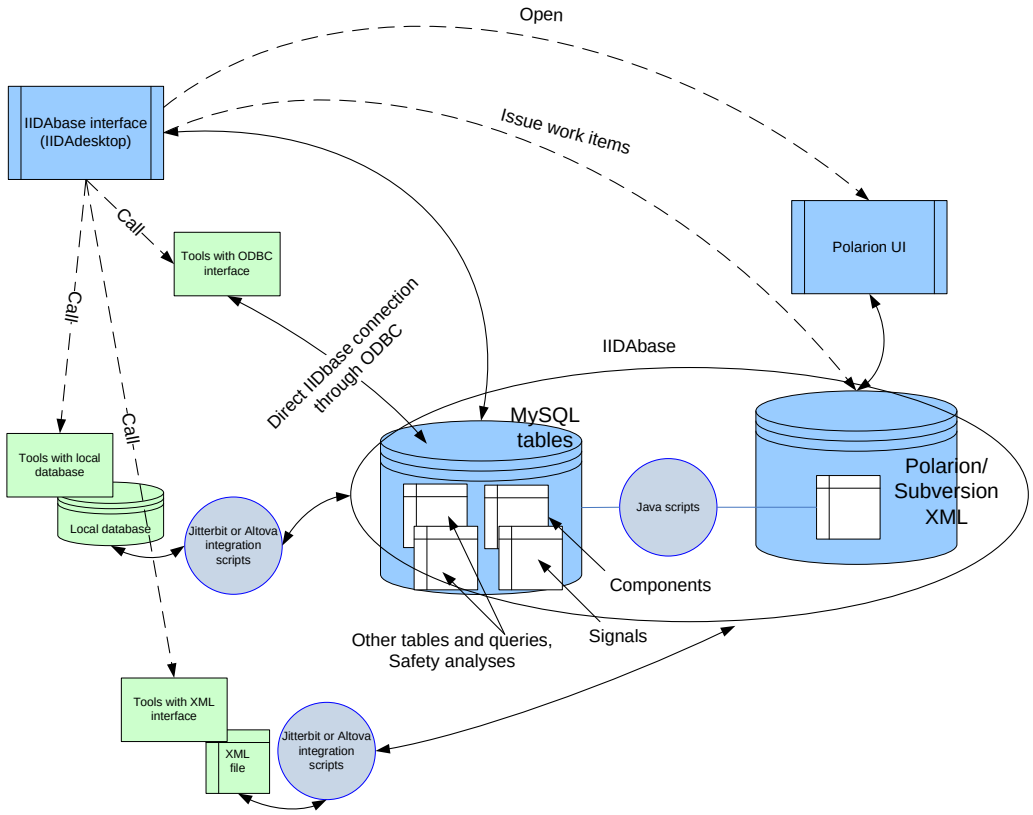


Figure 77. Tools integration model.

The integration model in Figure 77 assumes a dual IIDAbase repository model in which the MySQL/Access-based IIDAdesktop demonstrator works in conjunction with the Polarion ALM tool such that the master IIDAbase repository is under Polarion (actually under the Subversion version management program) and the necessary artefacts between IIDAdesktop and Polarion are synchronized by Java programs. As the main data repository resides under Polarion in Subversion (SVN), version management is handled by Polarion/SVN. This hybrid solution allows for the use of dedicated integration environments like Jitterbit or Altova MapForce to map data directly from the database tables or well-structured XML repositories of existing systems engineering tools to the IIDAbase repository through the MySQL database. The synchronization of data between MySQL and Polarion cannot be done with Jitterbit because the XML structure of Polarion is flat, i.e., it does not have a fixed XML schema[20]. Hence, Polarion's Java API is used and Java scripts are created to perform the synchronization of data.

The integration method depends on the type of tools, as follows:

A. Tools with an ODBC (Open Database Connectivity) interface. Such tools may connect directly to the IIDAbase database, but this may require changes to the tool to make full use of the IIDAbase model if the tool's configuration flexibility is poor. During the project, a special version of the Vertex ED electrical CAD tool was created by Vertex Systems Oy as a demonstration of a direct electrical CAD and IIDAbase integration (see Section 6.4).

B. Tools with a local database or well-structured XML repository. In such cases, an integration environment like Jitterbit or Altova MapForce can be used to synchronise the contents of the tool's local tables with the MySQL tables. This concept was also demonstrated with Vertex ED, but this time with a standard version of Vertex ED. The problem is that in some cases it is not possible to create new artefacts in the tool database via the database connection because of the internal logic of the tools (e.g., adding a record in a table may require updates to other tables). For one-way synchronization, from the tool to IIDAbase, this method may suffice, but if two-way integration with full manipulation of the artefact sets is needed, it is better to use the API (application programming interface) for integration (i.e., case C below), if the tool provides an API.

C. Tools with an API. This involves, e.g., Java programming. In the project, a hybrid IIDAbase solution that synchronises MS Access/MySQL and Polarion ALM implementations was demonstrated using the API-programming concept. The problem with this concept is the programming effort needed for the integration.

D. Other tools. E.g., a safety function verification tool called SISTEMA (by German DGUV/IFA) was integrated with IIDAbase using Java programming such that the SISTEMA project file, in which the artefacts are stored in a flat XML format created from the IIDAbase (implemented by MySQL) by a Java program, and vice versa the SISTEMA results are copied from the SISTEMA project file to IIDAbase by another Java program. Furthermore, the integration of the

[20] A fixed schema would be difficult to arrange because such ALM tools are generic and hence have to support all the possible artefact models of the companies buying the tool.

IEC 61131-3 PLC (programmable logic controller) programming environments supporting the PLCopen XML standard was performed using this method, and the results were demonstrated with the CoDeSys and MULTIPROG programming tools. Integration is limited to the I/O interface because the automatic generation of the function block networks requires well-established design patterns for safety-critical machine control software. The integration target in the PLCopen case was IIDAbase implemented as a Polarion Subversion repository. The problem with concept D is the programming effort needed for integration.

Waltersdorfer et al. [23] use a method similar to cases C and D; they use Smooks[21] as the scripting framework to create the integration transformations.

It may be that some tools cannot be integrated perfectly into IIDAbase, for example, in the case of SISTEMA. Although most of the information integrates well, the transfer of certain pieces of the information from SISTEMA to IIDAbase would require that part of the SISTEMA internal logic to be programmed into the integration scripts.

Type A tools include, e.g.:

- Microsoft Office tools incl. Visio
- VTT safety analysis tools
- Vertex ED electrical CAD tool (tailored version)
- (possibly Eclipse + EMF + CDO)

Type B tools include, e.g.:

- CaliberRM requirements management
- OSRMT requirements management
- Vertex ED electrical CAD tool
- Testlink test management tool

Type C tools or tools with an XML interface in general include, e.g.:

- IEC 61131-3 tools, e.g., CoDeSys 3.0 and MULTIPROG
- Possibly the new generation of CANopen configuration tools

Type D tools include, e.g.:

- SISTEMA safety evaluation tool
- Polarion.

[21] http://www.smooks.org [Referenced 12.01.2011]

6.1 Jitterbit

Jitterbit is an open source integration tool that allows integration of XML files, various database formats and other data sources such as FTP, web services and hierarchic file structures. It was used to mirror and synchronize MySQL tables from the Vertex ED electrical CAD tool to enable its interfacing to IIDAbase. With Jitterbit, it is possible to schedule these synchronizations or run them on demand. It also allows type conversions and combinations of table rows or cells. An example of a Jitterbit synchronization script is shown in Figure 78. In the figure, some fields are censored to protect the intellectual property of Vertex.

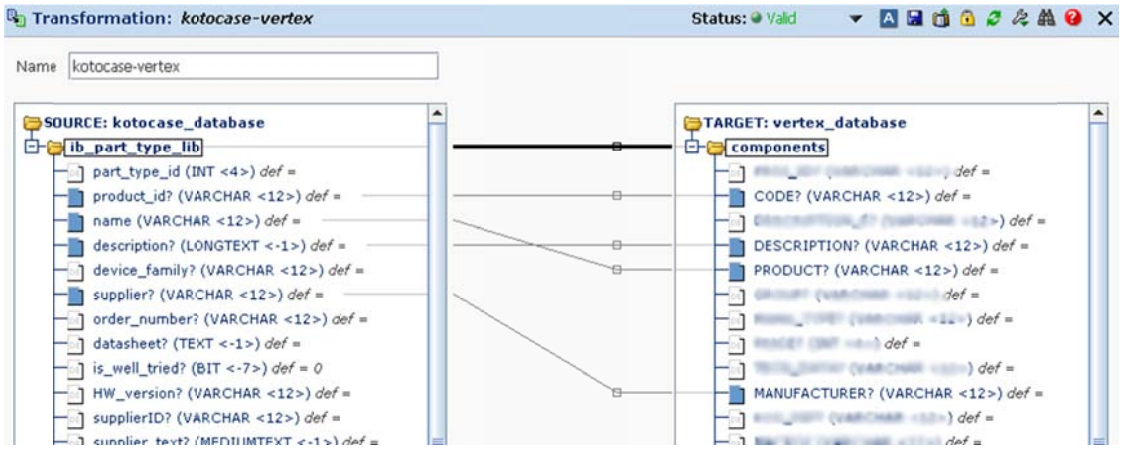


Figure 78. Jitterbit integration example.

The figure shows a graphical representation of the script created with the Jitterbit Integration Environment tool. In the example, a mapping between the IIDAbase database and the database of the Vertex ED tool is scripted to provide the information stored in IIDAbase to Vertex for the electrical engineer to use. As the databases are not fully compatible, only the relevant fields are synchronized. This script, or a 'transformation' in Jitterbit terms, can then be scheduled to run, for example, every night.

6.2 Polarion-MySQL integration

The data transfer between Polarion, the MySQL database and the Access tool was implemented using the Polarion Java API. Figure 79 shows the basic principle of the functionality. In the figure, the interactions between the components are shown as arrows with a descriptive text next to the arrow. The circles attached to the database shapes represent the method of accessing the database or repository.

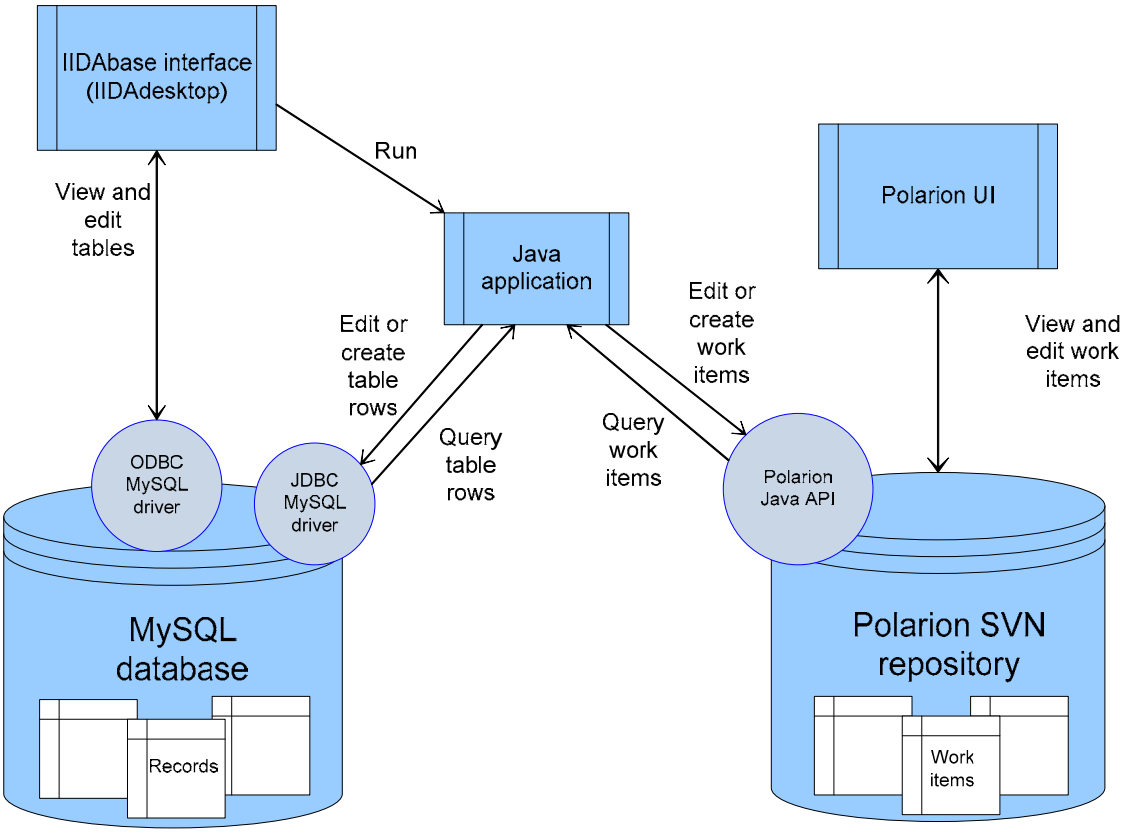


Figure 79. Integration between the MySQL database and SVN-based Polarion work item repository.

The Java application is called when a user presses a button in a form, for example, 'Fetch all' as shown in the Risk assessment form in Figure 46. The project in question and the mode are given as parameters to the Java application. The application then performs the necessary MySQL queries via JDBC (Java Database Connectivity) and uses the Polarion API to carry out the relevant operations to the Polarion work items in question. In the case of refreshing the risk assessments, the Polarion API is first queried for all *Risk assessment* work items related to the project in questions and the *Risk assessment* table in the MySQL database is then updated with the information from the work items. If a work item has no corresponding table row, a new one is created.

An example of the use of the Polarion Java API is shown in Program 4.

Program 4. Polarion API usage example code.

```
// Set up the address for Polarion
setUpPolarionAddress();
// Set up the Polarion web services
setUpPolarionWebServices();

// log in, get credentials from prop-class
sessionService.logIn(prop.getProperty("user"), prop.getProperty("passwd"));
project = projectService.getProject(projectID);

// Begin a transaction
sessionService.beginTransaction();

// Query the work items with hazard_code as their title
String lucenequery = ("title:\"" + hazard_code + "\"");
String[] uris = trackerService.queryWorkItemUris(lucenequery, "ID");

// ... do some checks and locate the work item in question
// (implementation omitted from this example, place of uri at variable i)

WorkItem wi = trackerService.getWorkItemByUri(uris[i]);

// set the properties of the work item that need to be updated
wi.setTitle(title);
wi.setDescription(desc);
wi.setSeverity(new EnumOptionId(severity.toLowerCase()));
wi.setProject(project);
wi.setType(type);

// set the custom field “hazard_code” of the work item
CustomField f = new CustomField();
f = trackerService.getCustomField(wiURI, "hazard_code");
f.setValue(hazard_code);
trackerService.setCustomField(f);

// ... set other custom fields (omitted)
// Save the updated work item
trackerService.updateWorkItem(wi);
// End the transaction and the session
sessionService.endTransaction(false);
sessionService.endSession();
```

In the example, a work item is fetched from the Polarion SVN repository, edited and then saved. The work items are handled through a web service called *TrackerService*, and the session is handled in *SessionService*. After it is acquired through the tracker service, a work item is an ordinary Java object with member functions for getting and setting its properties.

The JDBC MySQL connectivity is shown in Program 5.

Program 5. JDBC MySQL connectivity example code.

```
// This will load the MySQL driver, each DB has its own driver
Class.forName("com.mysql.jdbc.Driver");
// Setup the connection with the DB
conn    =    DriverManager.getConnection("jdbc:mysql://192.168.0.1",    "user",
"password");
// Create a statement
Statement statement = conn.createStatement();
// Create a query
String query = "SELECT * FROM kotocase.ib_pha_hazards i, kotocase.ib_hazards k
where i.pha_hazard_id = "
+ pha_hazard_id + " AND i.ib_hazard_id = k.hazard_id;";
// Execute the query
statement.execute(query);
// Get the results of the query
resultSet = statement.getResultSet();
while(resultSet.next())
{
// Store the fields in variables
hazard_type_or_group = resultSet.getString("hazard_type_or_group");
}
// Close the resultset, the statement and the connection
resultSet.close();
statement.close();
conn.close();
```

In the example, a MySQL connection is opened, a query is performed and the results of the query are read to variables. These mechanisms make the data interchange from the MySQL database to Polarion and back possible. The developed Java application has four modes of operation. These are:

- updating (or inserting) a single PHA row from MySQL to Polarion
- inserting (or updating) various PHA work items from Polarion to MySQL
- updating (or inserting) a single risk assessment row from MySQL to Polarion
- inserting (or updating) various risk assessment work items from Polarion to MySQL.

Figure 80 shows the functionality of the first mode as a sequence diagram.

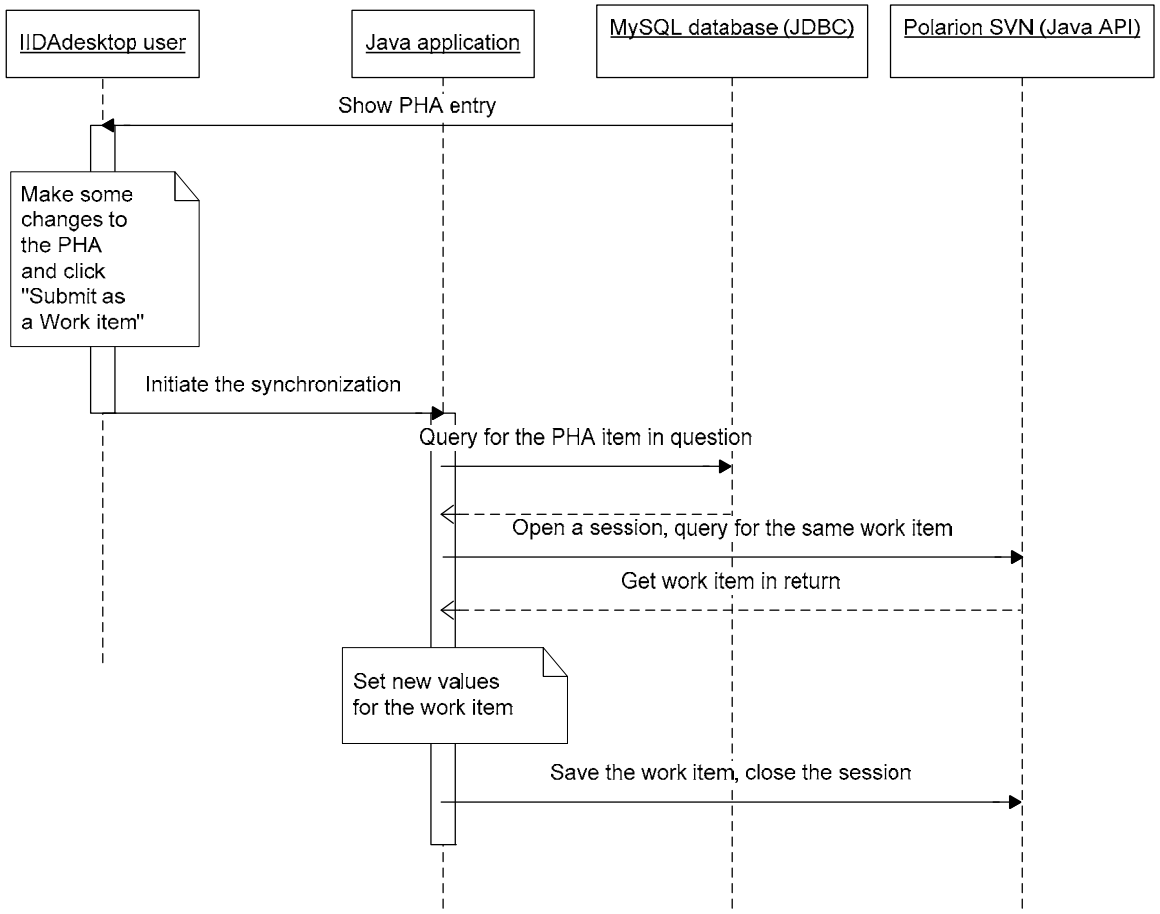


Figure 80. Java application functionality when updating a PHA.

The other modes function similarly, with queries from both the MySQL database and the Polarion SVN repository when needed, with some manipulation of information and checking of attributes for null values in between.

6.2.1 Discussion

The integration between Polarion and MySQL, while possible and to some extent quite easy to implement, has some problems. Using the Polarion Java API with work items of such complexity as those used in this project may easily cause the Java code used for integration to be quite inflexible. The custom field definitions and enumerations, such as the work items' customized statuses, have to be hard-coded into the program. This means that whenever a new project is created in Polarion, it has to have exactly the same kind of custom work items and enumerations; otherwise the integration program has to be customized again. It would be possible to program the integration software to be less hard-coded, but this method would also be prone to errors when the custom definitions of a Polarion project were edited.

The Java API does provide tools for doing almost everything to the work items and retrieving all their information, but replicating the hierarchy between the work items to the MySQL tables is quite difficult without radically changing the data model into something that replicates that of Polarion more closely, which would of course defeat the purpose of the IIDAbase model altogether.

Hence, it is suggested that the practical implementations of IIDAbase are either ALM or PLM tool implementations with an IIDAbase profile or database implementations such as the IIDAdesktop demonstrator. The former solution requires the creation of the IIDAbase profile, while the latter solution requires programming of version, management and traceability features into the IIDAdesktop tool. (The database model supports version handling either in the database itself or in a version management program like Subversion.) A hybrid solution like the one presented above is not the optimum solution.

6.3 CoDeSys and Multiprog integrations

The integration into IEC 61131-3 [9] environments was decided to be implemented using the PLCopen XML specification, which is an open standard for exchanging IEC 61131-3 structures in textual form. Currently, both CoDeSys and MULTIPROG support the import and export of PLCopen XML structures, and other IEC 61131-3 platforms will supposedly also do so eventually. PLCopen XML was chosen in spite of being a little less expressive than vendor-specific formats of these programming environments. The openness and standard form of PLCopen XML was considered an important factor when choosing the transfer method.

The integration is done with a specially crafted Java application. The basic principle of operation is presented in Figure 81. It was decided that the integration would be done against the Subversion (SVN) repository instead of MySQL. The operation was divided into two separate parts: forward and backward integration. Forward integration was defined as the transfer of data from IIDAbase to IEC 61131-3.

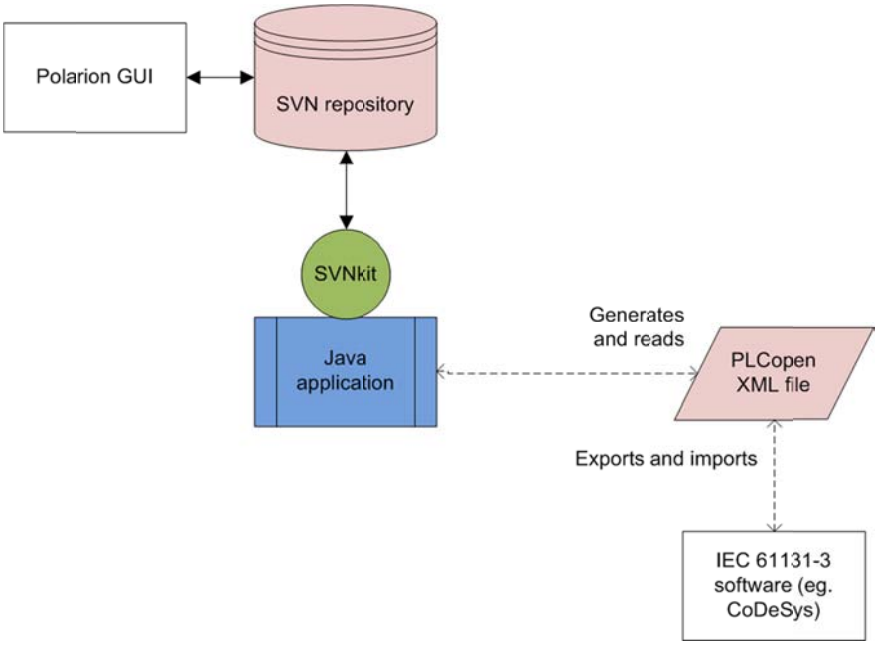


Figure 81. Basic principle of integrating IEC 61131-3 environments into IIDAbase.

The internals of the Java application are shown in Figure 82. In the forward integration mode, the application takes the address of an SVN repository as user input and if it finds a Polarion project that conforms to the IIDAbase *Structure* model (see Section 3.6) in the repository it then builds an inner meta model of the *Structure* model using the Polarion data. The application uses an SVNkit library for accessing the SVN server and SAX for parsing Polarion data files, which are XML documents. From the created data model, the application creates a DOM conforming to the PLCopen XML schema. This DOM is then written to an XML file that is the final PLCopen XML file. This file can then be imported by the IEC 61131-3 environments, which support PLCopen XML.

In backward integration, the internal IIDAbase model is built in the same way as in the forward mode. The desired PLCopen XML file (exported from the IEC 61131-3 environment) is then processed to fill an internal IEC 61131-3 model. This information is compared with the IIDAbase information while internally mapping the IEC 61131-3 structures in the internal model to corresponding ones in the IIDAbase model. If there are differences between these two information sets, the IIDAbase information is updated according to the one in the IEC 61131-3 model. Finally, the changed items of the IIDAbase model are written to the SVN server.

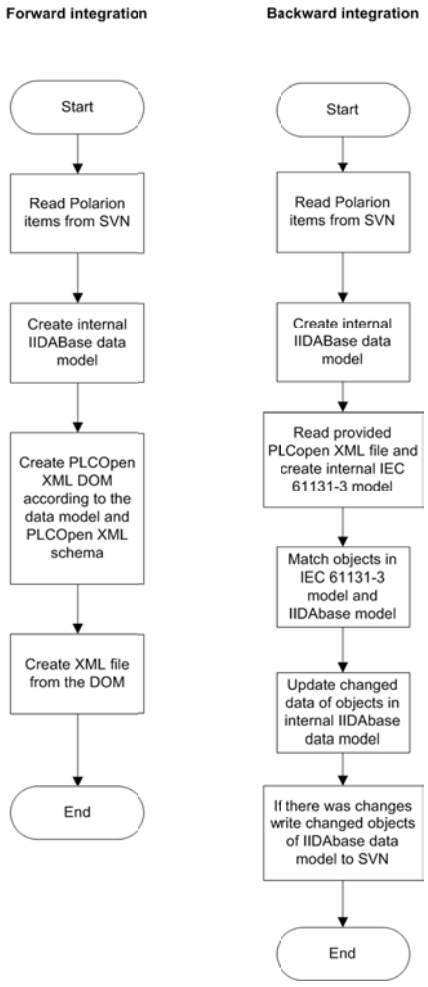


Figure 82. Internals of the IEC 61131-3 integration into IIDAbase.

IEC 61131-3 is a standard that defines programming languages for programmable logic controllers. It does not provide means to describe hardware-related structures. As the IIDAbase *Structure* model, on the other hand, is all about hardware structures, there is a semantic heterogeneity between these two data models. When observed from the hardware point of view, the IIDAbase *Structure* model is far more expressive than IEC 61131-3, which causes the integration between these two to be demanding. We are able to automatically generate the following IEC 61131-3 structures from IIDAbase as depicted in Figure 83:

- Configuration (from each *Part* in IIDAbase)
- Variable for Configuration (from each *Signal flow* linked to *Part*)
- Address for Variable (from *Electrical pin* linked to *Signal flow*).

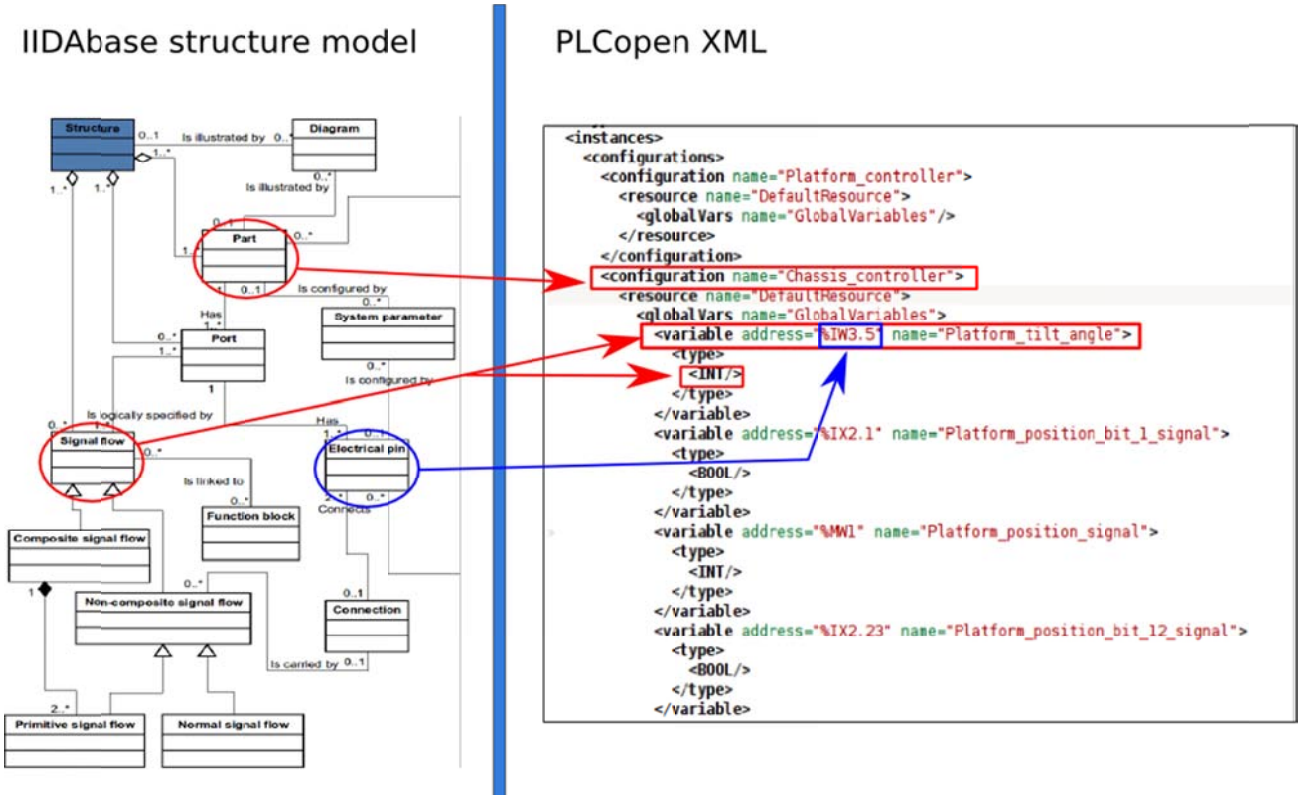


Figure 83. Relations between the IIDAbase structure model and the IEC 61131-3 structures in PLCopen XML.

In addition, a function block stub is created for every *Composite signal flow*. In this function block, two or more variables (derived from *Primitive signal flow*s, which together compose the *Composite signal flow* in question) are taken as an input, and one output variable is given. The logic for combining these *Primitive signal flows* into a *Composite signal flow* cannot be automatically determined and the user should fill in the implementation of the function block stub manually using the given variables in the IEC 61131-3 environment.

In Figure 84, the data are presented in CoDeSys after importing a generated PLCopen XML file.

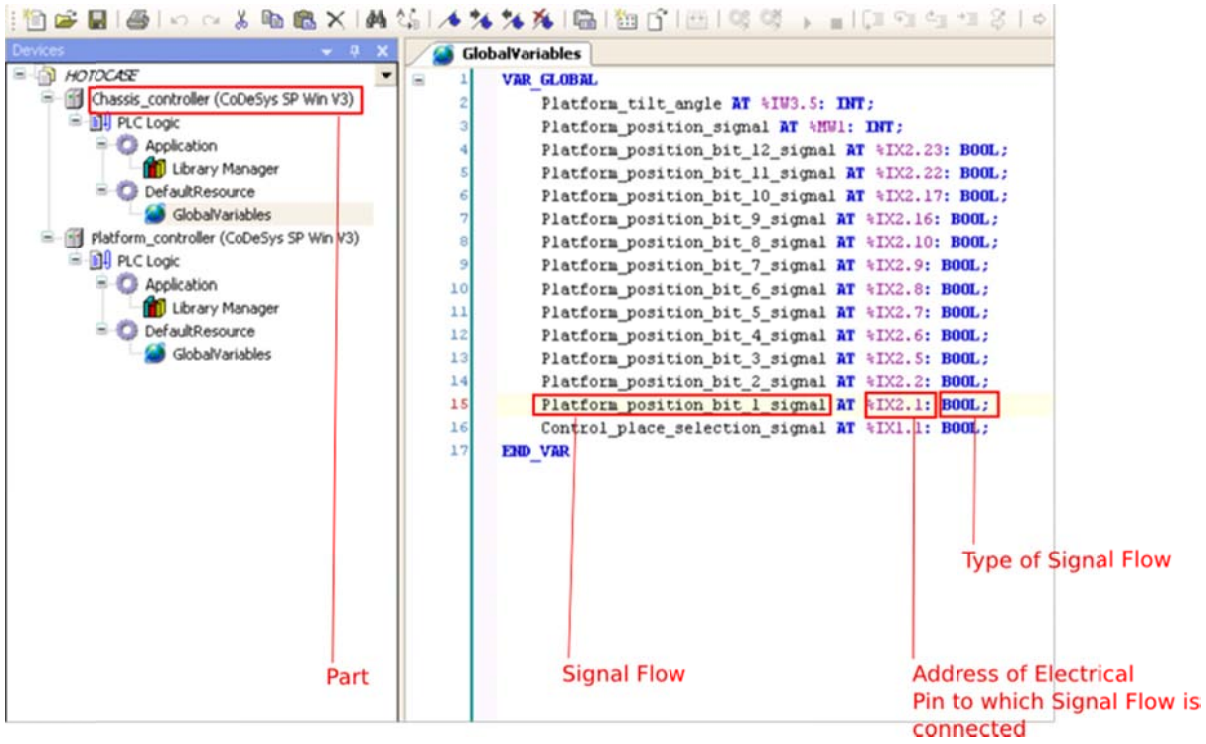


Figure 84. PLCopen XML data in CoDeSys and corresponding items in IIDAbase.

For the backward integration, only the changing of a variable type was implemented. Currently, CoDeSys and MULTIPROG do not support the Global ID feature of PLCopen XML, and hence the objects have to be identified by their name, thus preventing a change of name. For the same reason, we are not able to reliably change an address of a variable as we cannot identify the corresponding *Electrical pin* in the IIDAbase repository. We would be able to create a whole new structure model to IIDAbase from scratch using backward integration, i.e., having an empty structure model and populating it using the IEC 61131-3 data. This would leave the structure model inconsistent, however, and would require many of the blanks in the model to be filled in manually. Thus, it was not chosen for implementation at this point.

6.4 Vertex ED integration

Two integration demonstrations with Vertex ED were carried out. The first demonstration was done using the Jitterbit integration environment as presented in Section 6.1. The components library of Vertex ED was reflected in the IIDAbase part type library and vice versa. The demonstration proved the applicability of Jitterbit in cases in which a tool's data repository was implemented as a database. In general, however, it may be difficult with this type of integration to carry out two-way synchronization of the data, as pointed out in Section 0 Case B.

Fortunately, during the discussions with Vertex Oy, the persons from Vertex offered to make a demonstration version of Vertex ED such that the ED tool connected directly to IIDAbase without any

integration environment or scripts between the Vertex tool and IIDAbase. Hence, it was decided that a Vertex ED demonstration with direct use of the IIDAbase repository would be created. The goal of the demonstration was set as follows:

- to integrate Vertex eCAD into the IIDAbase *Structure* model
- to make it possible to pick *Part* artefacts from IIDAbase to an eCAD drawing. The parts contain meta information like part names, but they also relate to ports, and the ports relate to pins. Pins of ports of different parts can be connected together via a connection. A signal flow (the application-specific logical signal) can be designated to electrical pins and to a connection
- to display the meta information of a part, especially the name of the part in the drawing, while also providing the possibility of browsing other meta information, although not necessarily putting it in the drawing
- to pick the pin numbers and name from the IIDAbase and display them in the drawing
- to pick a signal flow from the IIDAbase, attach it to a wire in the drawing and display it in the drawing
- to use the symbol names of the parts from the IIDAbase.

A small control system was defined for use in the demonstration. The control system is illustrated in Figure 85.

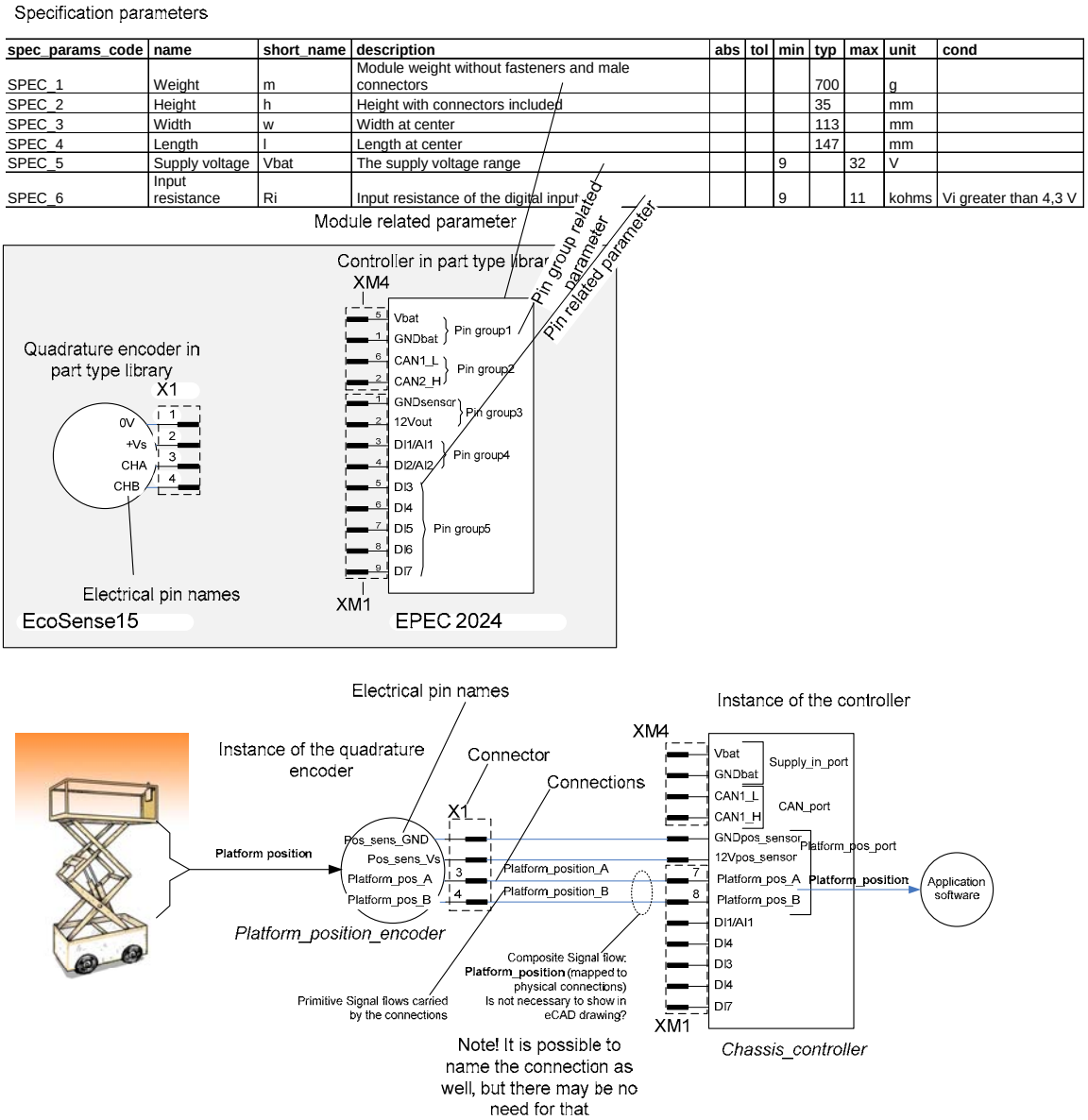


Figure 85. A control system case used to demonstrate the Vertex ED – IIDAbase integration.

The preconditions of the demonstration were such that the system engineer had created the control system in Figure 85 in the IIDAbase. Now, it was time to pass on the design to the electrical CAD designer. As the Vertex ED tool connects directly to IIDAbase, the electrical CAD designer can pick up the components from IIDAbase (see Figure 86) and start to design the physical wiring (see Figure 87).

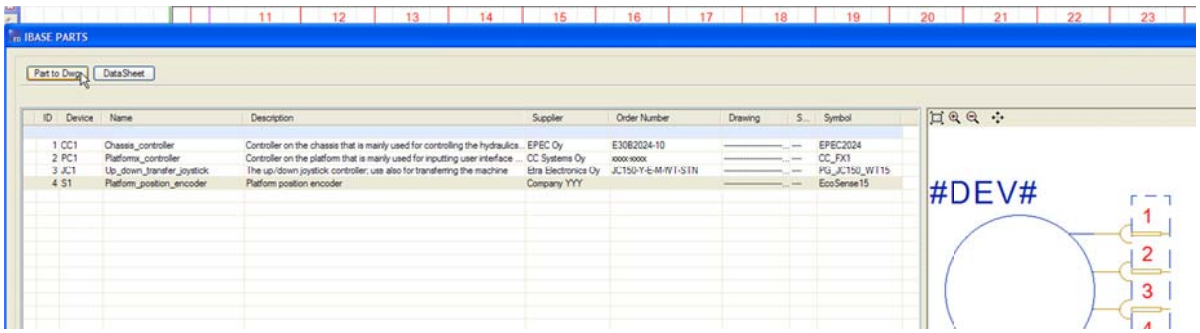


Figure 86. Selecting the components from IIDAbase (the user interface of Vertex ED combines the meta information for parts from the part type library and from the Part artefact [the instance of the part type])

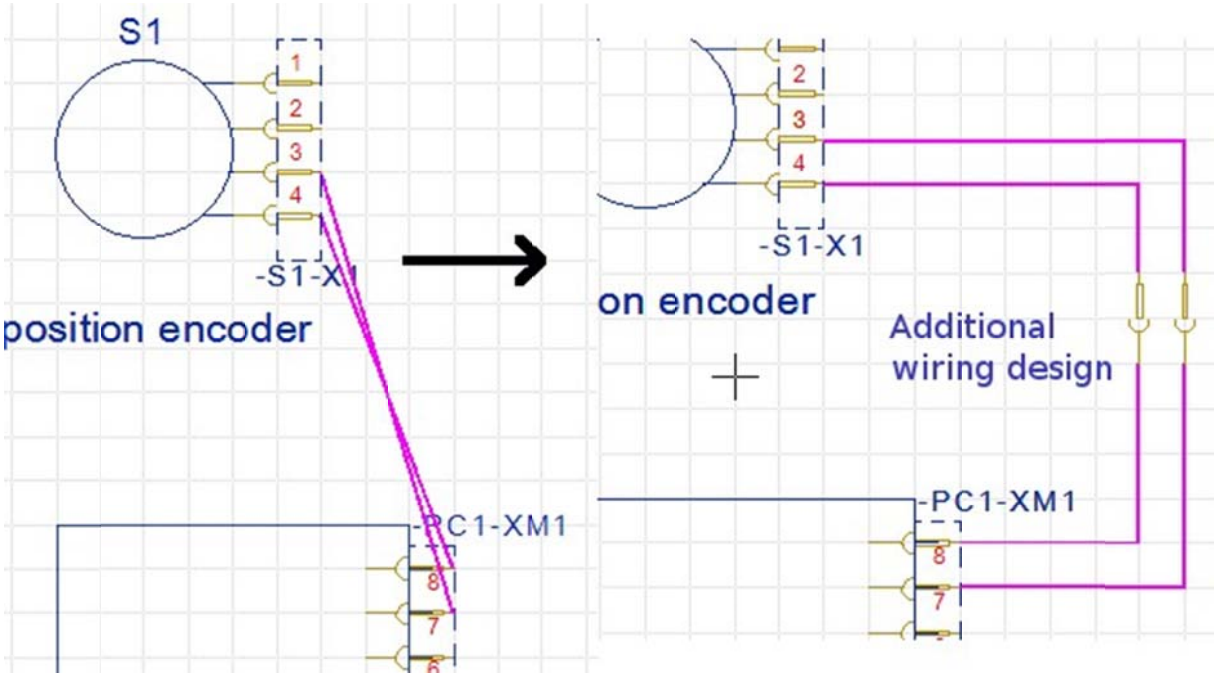


Figure 87. Starting the eCAD design.

As can be seen in Figure 87, the components picked up from IIDAbase are connected to each other with ‘rubber wires’ that show the connections made by the system engineer. When the eCAD designer adds joints etc. to the wiring (as also shown in Figure 87), he must press the ‘IIDAbase diagram checks’ selection to check that all the connections created by the system engineer are still valid after breaking the lines and adding the joints. Furthermore, the name of the part, *Platform position encoder* (partly shown in Figure 87), comes from IIDAbase. Hence, if the system engineer changes the name in the IIDAbase, the name of the component is also updated in the electrical CAD drawing after refreshing the figure. The same can be done for the signal flow names and for the pin names (not shown in the figure).

The Vertex ED demonstration proved that for a tool that already uses a database for storage, it is relatively easy to modify it to use IIDAbase directly.

The demonstration worked out well and highlighted the potential of the IIDAbase concept in electrical engineering.

6.5 SISTEMA integration

A two-way integration scheme was developed between SISTEMA (Safety Integrity Software Tool for the Evaluation of Machine Applications) tool by IFA (Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung) and IIDAbase. The integration makes it possible for the safety engineer to use the IIDAbase *Structure* data created by the system engineer, when calculating PL values of safety functions with SISTEMA. Furthermore, some information such as the PL, CCF and DC values calculated by the SISTEMA tool can be ported back into IIDAbase for review by the system engineer.

From an integration perspective, the IIDAbase MySQL database is first used as a source and after the work performed in SISTEMA, a target for the data. SISTEMA handles data in flat XML files called .ssm. An example of the SISTEMA XML file is shown in Figure 88.

```
- <table table_name="sfs">
  - <fields>
      <field field_name="isprotected" field_kind="integer" field_Size="0" />
      <field field_name="normversion" field_kind="string" field_Size="1024" />
      <field field_name="ssmversion" field_kind="string" field_Size="128" />
      <field field_name="name" field_kind="string" field_Size="1024" />
      <field field_name="oid" field_kind="string" field_Size="36" />
      <field field_name="document" field_kind="string" field_Size="2048" />
      <field field_name="documentation" field_kind="string" field_Size="8192" />
      <field field_name="plr" field_kind="string" field_Size="3" />
      <field field_name="plrdet" field_kind="string" field_Size="11" />
      <field field_name="plrdocumentation" field_kind="string" field_Size="8192" />
      <field field_name="plrstandard" field_kind="string" field_Size="8192" />
      <field field_name="plrstandardfile" field_kind="string" field_Size="2048" />
      <field field_name="reaction" field_kind="string" field_Size="0" />
      <field field_name="riskparamf" field_kind="integer" field_Size="0" />
      <field field_name="riskparamp" field_kind="integer" field_Size="0" />
      <field field_name="riskparams" field_kind="integer" field_Size="0" />
      <field field_name="safestate" field_kind="string" field_Size="8192" />
      <field field_name="sftype" field_kind="string" field_Size="1024" />
      <field field_name="triggerevent" field_kind="string" field_Size="8192" />
      <field field_name="plcal" field_kind="string" field_Size="3" />
      <field field_name="pfhcal" field_kind="string" field_Size="12" />
    </fields>
  - <rows>
      <row isprotected="0" normversion="ISO 13849-1:2006, ISO 13849-2:2003"
        ssmversion="1.1.1" name="Avoidance of unexpected movements"
        oid="5EB1BA30-6609-4E64-9BF3-8CDF99E38F5A" document=""
        documentation="" plr="plD" plrdet="detDirect" plrdocumentation=""
        plrstandard="" plrstandardfile="" reaction="Execution of the working
        movements of the earth-moving machine desired by the user"
        riskparamf="1" riskparamp="0" riskparams="0" safestate="Stop of the
        dangerous movements of the earth-moving machine" sftype="Prevention
        of unexpected start-up" triggerevent="Manual movement of the multi-
        purpose control (MPC) by the user" plcal="plD" pfhcal="3.9824927516797E-
        7" />
    </rows>
</table>
```

Figure 88. An .ssm-file XML example.

Figure 89 shows the relevant part of the IIDAbase *Structure* model.

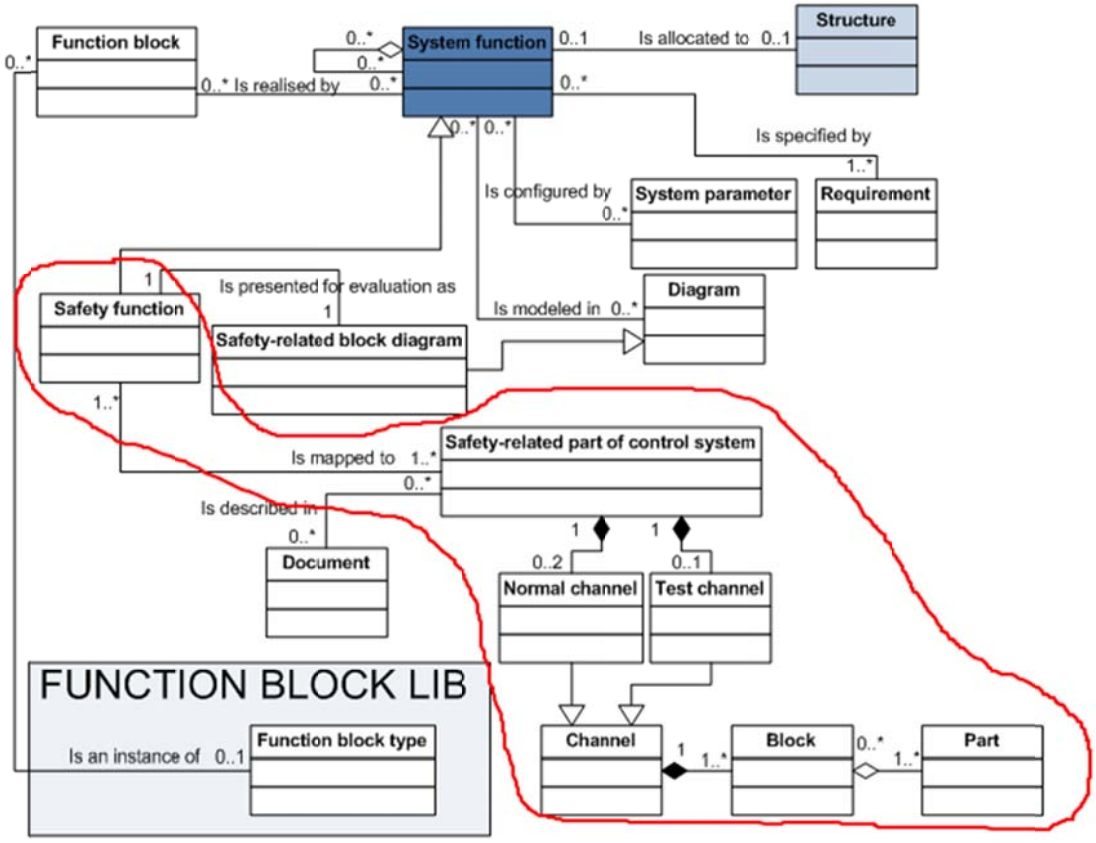


Figure 89. SISTEMA-related parts of the IIDAbase data model.

A SISTEMA .ssm-file consists of ‘projects’ (PR) that have safety functions (SF, named ‘components’ in the .ssm file), ‘subsystems’ (SB), ‘channels’ (CH), ‘blocks’ (BL) and ‘elements’ (EL). An example of the SISTEMA data hierarchy as it is shown in the SISTEMA user interface is presented in Figure 90.

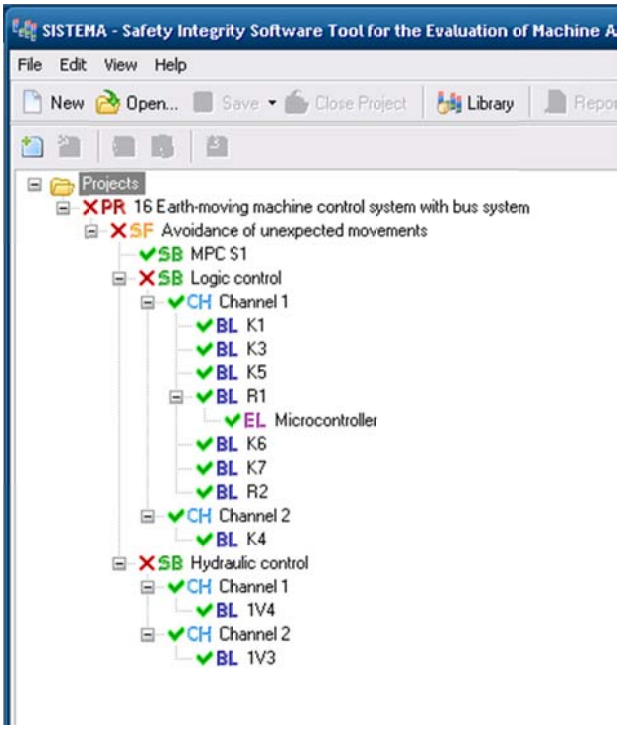


Figure 90. SISTEMA data model as presented in the SISTEMA user interface.

The technologies and techniques used in the integration include Java and its string manipulation, file access and OID generation facilities as well as JDBC for MySQL queries.

Two separate Java applications were written, *mysql2sistema* and *sistema2mysql*. They make the integration in different directions, as their names suggest.

The main input from MySQL to SISTEMA are all the artefacts of the SF-SB-CH-BL-EL hierarchy and the MTTFd values of individual blocks, parts (element in SISTEMA) or safety-related parts of a control system (subsystem in SISTEMA). After the safety engineer has finished with SISTEMA, PL, PFH, CCF and DC values are provided to the IIDAbase. The data flows (inputs and outputs) are also presented in Figure 91.

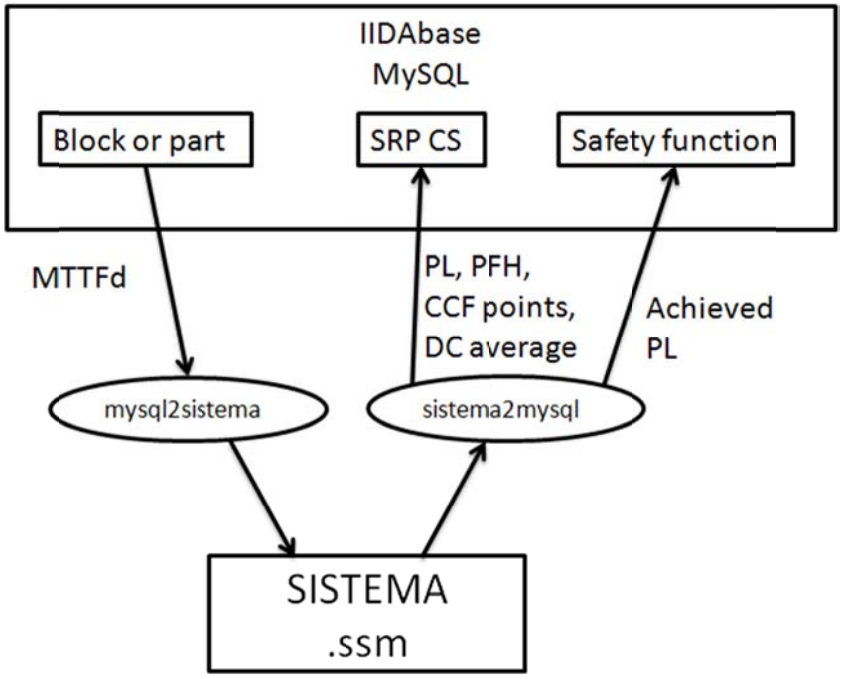


Figure 91. Data flows in IIDAbase-SISTEMA integration.

6.5.1 MySQL 2 SISTEMA

The *mysql2sistema* application integrates MySQL data into SISTEMA by reading the relevant data from MySQL via JDBC, saving it as Java objects and then writing a SISTEMA-compatible .ssm file with the contents read from the database. The application retrieves the *system_code* of the system under consideration as a parameter. The name and location of the .ssm file to be written are also read from the database. Finally, the SISTEMA application is started automatically, and it also opens the newly created .ssm file.

The data from MySQL are saved in Java objects that replicate the hierarchical structure of the data. The objects are instances of *SistemaProject*, *SistemaSafetyFunction*, *SistemaSubsystem*, *SistemaChannel*, *SistemaBlock* and *SistemaElement* classes, which include all the data related to a row in a corresponding MySQL row or SISTEMA object. The classes have getters and setters for all the data and an *ArrayList* data structure that holds the objects hierarchically below the class in question.

The .ssm file is first produced as a string by going through the object hierarchy and filling an empty string copied from an empty .ssm file. The string is then written to a file, which is created if necessary.

6.5.2 SISTEMA 2 MySQL

The *sistema2mysql* application also receives the *system_code* as a parameter and uses it to find the location and name of the .ssm file. It then reads the .ssm file contents and creates a set of Java objects portraying the data model present in the .ssm file. The Java objects that model the hierarchy of the elements are the same as in *mysql2sistema*, the only difference being that the unique OIDs of the artefacts are used as identifiers. After reading the .ssm file, the application updates the corresponding fields in the IIDAbase MySQL database with values provided by SISTEMA. The values that are added include the achieved PL value of the safety-related part of the control system (SRP/CS) carrying out the safety function, PL, PFH, CCF points, DC average, MTTFd values and Category for the safety-related sub-parts of the control system, and finally DC values for the blocks.

6.5.3 Notes

The integration between SISTEMA and MySQL, while successful in this case, is not without drawbacks. The SISTEMA application is not designed to work with .ssm files filled by an external program, and the .ssm files do not include all the information visible in the user interface. Hence, it is only partly possible to retrieve data from SISTEMA. Examples of data not available in the .ssm file include the category conditions for the artefacts and their values (condition met or not met). These cannot be extracted from the .ssm file without extensive logic and SISTEMA-like functionality in the integration software.

In the case of prefilled .ssm files, the SISTEMA may function unexpectedly. For example, if the PL value of a safety function is prefilled in the .ssm file but the value has not yet been calculated in the SISTEMA user interface (if some preconditions for calculation are not met), the value in the .ssm file may remain, even after saving, as it has been set manually (whatever the value). This may provide unfavourable results in integration if the values in .ssm files are set incorrectly and SISTEMA cannot yet calculate the actual results.

7. Summary

The systems engineering artefact model, the IIDAbase model, presented in this research note is ready for use by the machine control system developers. The lack of a sophisticated tool that implements and uses the full potential of the model hinders its use however. Commercially available Application Lifecycle Management tools, like Polarion ALM presented in this report, are good choices for implementing the model, provided that a dedicated user interface profile for the use of the model is created. This can be implemented without tailoring the tool itself. Another possibility is to create a new tool similar to the MS Access-MySQL demonstration tool, but it would require the implementation of features like version management and traceability, which are standard features of ALM and PLM tools. This still leaves a systems engineering tool with which the system engineer can draw the architecture diagrams such that the artefacts according to the IIDAbase model are created automatically while drawing. Existing SysML tools with database connection are good candidates for the basis of such a tool, but while the culture of using SysML is still in its infancy in industry, a better alternative would be to tailor such a tool from an existing electrical CAD tool, as these are better known by the machine manufacturing companies.

Nevertheless, while waiting for the tools, the machine manufacturers can use the IIDAbase model starting from the risk assessment model: the risk assessment needs to be put in order anyway due to the Machinery Directive.

References

[1] Anon. AP233 Public and Private Information Portal [Online]. Available at http://www.ap233.org [Referenced 23.03.2011].

[2] ISO/PAS 20542:2006. Industrial automation systems and integration – Product data representation and exchange – Reference model for systems engineering. International Organisation for Standardization.

[3] Price, D. ISO 10303-233 Systems Engineering and Design (AP233) [Online]. December 2006. Available at: http://www.scribd.com/doc/6658935/AP233-Brief-Introduction-Presentation [Referenced 23.03.2011].

[4] Anon. A guide to modelling with SysML [Online]. Publisher: ConnectingIndustry, 13.08.2007. Available at: http://www.connectingindustry.com/story.asp?storycode=181473 [Referenced 23.03.2009].

[5] ISO/DIS 10303-233. Industrial automation systems and integration – Product data representation and exchange – Part 233: Application protocol: Systems engineering. Geneva: International Organization for Standardization (ISO), 2009.

[6] Klein, H. MSR Working Group MEDOC Presentation of Results – Final report. MSR-MEDOC, 2002-07-01. 17 p. Available at: http://www.msr-wg.de/medoc/download/literature/msr-tr-medoc-en/msr-tr-medoc-en.pdf [Referenced 23.03.2011].

[7] Klein, H. MSR Engineering Documentation (MEDOC) – Structure Principles of the MSRSYS DTD – Scope: Systems and Hardware [Online]. MSR-MEDOC, 2002-02-07. 138 p. Available at: http://www.msr-wg.de/medoc/download/msrsys/v120a/msrsys-sp-en/msrsys-sp-en.pdf [Referenced 23.03.2011].

[8] Anon. Element und Attribut Dokumentation MSRSYS V1.2.0a. MSR-MEDOC, 2001-12-14.

[9] IEC 61131-3. Programmable controllers – Part 3: Programming languages. 2003-01-21 edn. Geneva: International Electrotechnical Commission (IEC), 2003.

[10] CiA DSP 311 2007. CANopen device description – XML schema definition. 1.0.2 ed. Germany: CAN in Automation e.V. 63 p.

[11] Weilkiens, T. Systems Engineering with SysML/UML – Modeling, Analysis, Design. MK/OMG Press, 2008. 320 p. ISBN 978-0-12-374274-2.

[12] ISO/IEC 15288 (IEEE Std 15288-2008). Systems and Software Engineering – System Life Cycle Processes. 2008-02-01 ed. International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), International Electrotechnical & Institute of Electrical and Electronics Engineers (IEEE), 2008. 81 p.

[13] Haskins, C., Forsberg, K. & Krueger, M. Systems Engineering Handbook – A guide for system life cycle processes and activities. V. 3.1 ed. International Council on Systems Engineering (INCOSE), 2007. INCOSE-TP-2003-002-03.1. 174 p. + app. 130 p.

[14] ISO 14121-1. Safety of machinery -- Risk assessment – Part 1: Principles. 2007-09-01 ed. Geneva: International Organisation for Standardization (ISO), 2007. 28 p.

[15] ISO 13849-2. Safety of machinery – Safety-related parts of control systems – Part 2: Validation. Geneva: International Organisation for Standardization (ISO), 2003. 50 p.

[16] IEC 61508-5. Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 5: Examples of methods for the determination of safety integrity levels. 1998-12-03 edn. Geneva: International Electrotechnical Commission (IEC), 1998.

[17] IEC 62061. Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems. Geneva: International Electrotechnical Commission (IEC), 2005. 205 p.

[18] ISO 13849-1. Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design. Geneva: International Organisation for Standardization (ISO), 2006. 85 p.

[19] IEC 61784-3. Industrial Communications – Fieldbus Profile – Part 3: Profiles for functional safety communications in industrial networks. Geneva: International Electrotechnical Commission (IEC), 2007. 47 p.

[20] Hedberg, H. & Wang, Y. Methods for Verification and Validation of Distributed Control Systems. PALBUS Work Package 10.10. Borås: SP Swedish National Testing and Research Institute, 2001. 66 p.

[21] Hietikko, M., Malm, T. & Alanen, J. Koneiden ohjausjärjestelmien toiminnallinen turvallisuus. Ohjeita ja työkaluja standardien mukaisen turvallisuusprosessin luomiseen [Functional Safety of Machine Control Systems. Instructions and Tools for the Creation of Standard Safety Process]. Espoo: VTT, 2009. 74 p. + app. 14 p. VTT Tiedotteita – Research Notes 2485. Available at: http://www.vtt.fi/inf/pdf/tiedotteet/2009/T2485.pdf [Referenced 23.03.2011] ISBN 978-951-38-7298-4.

[22] Alanen, J. Assessing safety of CAN-communications systems. The 6th International Conference on safety of Industrial Automation Systems (SIAS 2010). Tampere, 14–15.6.2010 SIAS 2010 Proceedings. Suomen Automaation Tuki Oy. (Finnish Automation Support Ltd). Helsinki 2010. 6 p.

[23] Waltersdorfer, F., Moser, T., Zoitl, A. & Biffl, S. Version management and conflict detection across heterogeneous engineering data models. Proceedings of 8th IEEE International Conference on Industrial Informatics (INDIN). Osaka, Japan, 13–16 July 2010. Pp. 928–935. ISBN 978-1-4244-7298-7.

Series title, number and
report code of publication

VTT Research Notes 2583
VTT-TIED-2583

Author(s)
Jarmo Alanen, Iiro Vidberg, Heikki Nikula, Nikolaos Papakonstantinou, Teppo Pirttioja & Seppo Sierla

Title

Engineering Data Model for Machine Automation Systems

Abstract

This research note presents a data model that defines the key artefacts of the systems engineering and safety processes of machine control systems. Existing data models, like the ISO 10303-233 and the German automotive specifications, especially MSRYS, are exploited when defining the model.

The model presented in this research note defines artefacts related to the overall system and its context, requirements and their validation, risk assessment, behaviour (system use cases and functional specifications), system structure and documentation. The emphasis is on defining the linkage between these artefacts, including across process boundaries. These links form the traceability chains from risk assessments to safety requirements, functional specifications, designs, implementation, verification and validation. Besides providing traceability of artefacts, such a model allows centralized, single source, data repository implementations that ensure a consistent view of artefacts in different design disciplines, like software development and electrical CAD.

Two implementations of the designed model are presented: Polarion ALM and MySQL with an MS Access front-end. Examples of tool integrations using the model are also demonstrated.

The benefits are also emphasized from the perspective of documentation. A centralized artefact repository can support the automated creation of documents based on demand. For example, the system requirements specification and the relevant parts of the technical file, which is required by the Machinery Directive, can be generated from the database.

ISBN
978-951-38-7711-8 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN
VTT Tiedotteita – Research Notes
1235-0605 (soft back ed.)
1455-0865 (URL: http://www.vtt.fi/publications/index.jsp)

Project number
31203

Date
May 2011

Language
English

Pages
131 p.

Name of project
Tietokantapohjainen koneenohjausjärjestelmän suunnittelu (TIKOSU)

Commissioned by
Tekes, FIMA ry

Keywords
Systems engineering, data model, risk assessment

Publisher
VTT Technical Research Centre of Finland
P.O. Box 1000, FI-02044 VTT, Finland
Phone internat. +358 20 722 4520
Fax +358 20 722 4374

VTT TIEDOTTEITA - RESEARCH NOTES

2565 Åsa Nystedt, Mari Sepponen, Seppo Teerimo, Johanna Nummelin, Mikko Virtanen & Pekka Lahti. EcoGrad. Ekotehokkaan kaupunkialueen toteuttaminen Pietarissa. 2010. 77 s. + liitt. 12 s.

2567 Tommi Kaartinen, Jutta Laine-Ylijoki, Auri Koivuhuhta, Tero Korhonen, Saija Luukkanen, Pekka Mörsky, Raisa Neitola, Henna Punkkinen & Margareta Wahlström. Pohjakuonan jalostus uusiomateriaaliksi. 2010. 98 s. + liitt. 8 s.

2568 Katariina Palomäki. Innovatiivisen verkostoyhteistyön edellytykset turvallisuusalalla. 2011. 113 s. + liitt. 6 s.

2569 Asko Talja. Ohjeita liikennetärinän arviointiin. 2011. 35 s. + liitt. 9 s.

2570 Tuomo Rinne, Peter Grönberg, Ville Heikura & Timo Loponen. Huoneistopalon sammutus vaihtoehtoisilla sammutusmenetelmillä. 2011. 80 s.

2571 SAFIR2010. The Finnish Research Programme on Nuclear Power Plant Safety 2007-2010. Final Report. Puska, Eija Karita; Suolanen, Vesa (eds.) 2011. 578 p.

2572 Kestävän rakentamisen prosessit. Häkkinen, Tarja (toim.) 2011. 100 s. + liitt. 3 s.

2573 Sirje Vares, Tarja Häkkinen & Jari Shemeikka. Kestävän rakentamisen tavoitteet ja niiden toteutuminen. Espoo Suurpellon päiväkodin arvio. 2011. 48 s. + liitt. 34 s.

2574 Marko Jurvansuu. Roadmap to a Ubiquitous World. Where the Difference Between Real and Virtual Is Blurred. 2011. 79 p.

2575 Towards Cognitive Radio Systems. Main Findings from the COGNAC project. Marja Matinmikko & Timo Bräysy (eds.). 2011. 80 p. + app. 23 p.

2576 Sebastian Teir, Antti Arasto, Eemeli Tsupari, Tiina Koljonen, Janne Kärki, Lauri Kujanpää, Antti Lehtilä, Matti Nieminen & Soile Aatos. Hiilidioksidin talteenoton ja varastoinnin (CCS:n) soveltaminen Suomen olosuhteissa. 76 s. + liitt. 3 s.

2577 Teuvo Paappanen, Tuulikki Lindh, Risto Impola, Timo Järvinen & Ismo Tiihonen, Timo Lötjönen & Samuli Rinne. Ruokohelven hankinta keskisuomalaisille voimalaitoksille. 2011. 148 s. + liitt. 5 s.

2578 Inka Lappalainen, Ilmari Lappeteläinen, Erja Wiili-Peltola & Minna Kansola. MULTIPRO. Vertaileva arviointi¬konsepti julkisen ja yksityisen hyvinvointipalvelun arviointiin. 2011. 64 s.

2579 Jari Kettunen, Ilkka Kaisto, Gerrit Weisenborn, Ed van den Kieboom, Riku Rikkola & Raimo Korhonen. Promoting Entrepreneurship in Organic and Large Area Electronics in Europe. Issues and Recommendations. 2011. 69 p. + app. 7 p.

2580 Оса Нюстедт, Мари Сеппонен, Микко Виртанен,Пекка Лахти, Йоханна Нуммелин, Сеппо Теэримо. ЭкоГрад. Концепция создания экологически эффективного района в Санкт-Петербурге. 2011. 89 с. + прил. 12 с.

2583 Jarmo Alanen, Iiro Vidberg, Heikki Nikula, Nikolaos Papakonstantinou, Teppo Pirttioja & Seppo Sierla. Engineering Data Model for Machine Automation 2011. 131 p.

ISBN 978-951-38-7711-8 (soft back ed.)
ISSN 1235-0605 (soft back ed.)
ISBN 978-951-38-6326-5 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0865 (URL: http://www.vtt.fi/publications/index.jsp)