nks
Nordisk kernesikkerhedsforskning
Norrænar kjarnöryggisrannsóknir
Pohjoismainen ydinturvallisuustutkimus
Nordisk kjernesikkerhetsforskning
Nordisk kärnsäkerhetsforskning
Nordic nuclear safety research

NKS-91
ISBN 87-7893-149-5

Traceability and Communication of Requirements in Digital I&C Systems Development Project Report 2003

Terje Sivertsen*, Rune Fredriksen*, Atoosa P-J Thunem*
Jan-Erik Holmberg**, Janne Valkonen** & Olli Ventä**

*IFE, Norway
**VTT, Finland

March 2004

Abstract

The overall objective of the TACO project is to improve the knowledge on principles and best practices related to the issues concretised in the pre-project. On basis of experiences in the Nordic countries, the project aims at identifying the best practices and most important criteria for ensuring effective communication in relation to requirements elicitation and analysis, understandability of requirements to all parties, and traceability of requirements through the different design phases. It is expected that the project will provide important input to the development of guidelines and establishment of recommended practices related to these activities.

In the year 2003, the TACO-project concentrated on four central issues:

- Representation of requirements origins
- Traceability techniques
- Configuration management and the traceability of requirements
- Identification and categorisation of system aspects and their models

The work was presented at the first TACO Industrial Seminar, which took place in Stockholm on the 12th of December 2003. The seminar was hosted by SKI.

Key words

Traceability, requirements, Digital I&C, systems development

NKS-91
ISBN 87-7893-149-5

Electronic report, March 2004

Published by
NKS Secretariat
NKS-775, Box 49
DK – 4000 Roskilde, Denmark

Phone +45 4677 4045
Fax +45 4677 4046
www.nks.org
e-mail nks@nks.org

Foreword

This document constitutes the 2003 report for the project "Traceability and Communication of Requirements in Digital I&C Systems Development" (NKS-R project number NKS_R_2002_16). The report discusses the main issues covered in the project's research activities in 2003, and presents details with regard to the project organisation, activities, and further plans.

The purpose of the report is to document the continued research work and related activities within the TACO project, including the First TACO Industrial Seminar (Stockholm, 12 December 2003). Particular emphasis has been put on relating knowledge on relevant software engineering issues to NPP needs and practice. The report provides an adequate basis and reference point for the work and activities to be carried out in the further continuation of the project, including the project's contribution to the Nordic Seminar on Nuclear Automation (Oskarshamn, 5-7 April 2004) and the Second TACO Industrial Seminar (Helsinki, December 2004). The plan for these activities is given in the appendix.

Halden, January 2004

Terje Sivertsen

Table of contents

Abbreviations

CEMSIS	Cost Effective Modernisation of Systems Important to Safety (Environment Project FIKS-CT-2000-00109)
CI	Configuration item
CMM	Capability Maturity Model
COTS	Commercial off-the-shelf
Digital I&C	Digital instrumentation and control
EUP	Enterprise Unified Process
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers, Inc.
IFE	Institute for energy technology
NKS	Nordic nuclear safety research
NPP	Nuclear power plant
NRWG	Nuclear Regulators Working Group
RUP	Rational Unified Process
SADT	Structured Analysis and Design Technique
SCM	Software configuration management
SCR	Safety Concept Report
SKI	Swedish Nuclear Power Inspectorate
SWEBOK	Software Engineering Body of Knowledge
TACO	Traceability and Communication of Requirements in Digital I&C Systems Development (NKS project number NKS_R_2002_16)
TVO	Teollisuuden Voyma Oy
UML	Unified Modeling Language
VTT	Technical Research Centre of Finland

Summary

The title of the reported project is "Traceability and Communication of Requirements in Digital I&C Systems Development", abbreviated TACO. The NKS project number is NKS_R_2002_16.

The overall objective of the TACO project is to improve the knowledge on principles and best practices related to the issues concretised in the preproject. On basis of experiences in the Nordic countries, the project aims at identifying the best practices and most important criteria for ensuring effective communication in relation to requirements elicitation and analysis, understandability of requirements to all parties, and traceability of requirements through the different design phases. It is expected that the project will provide important input to the development of guidelines and establishment of recommended practices related to these activities.

The overall aim of the preproject, which was carried out in the second half of 2002, was to identify the main issues related to traceability and communication of requirements in digital I&C systems development. By focusing on the identification of main issues, the preproject provided a basis for prioritising further work, while at the same time providing some initial recommendations related to these issues. The establishment of a Nordic expert network within the subject was another important result of the preproject.

The present report documents the continued research work and related activities carried out within the TACO project in 2003. These activities have constituted a natural continuation of the preproject, and have focused on the technical issues concretised in the preproject report. The work has concentrated on four central and related issues, viz.

- Representation of requirements origins
- Traceability techniques
- Configuration management and the traceability of requirements
- Identification and categorisation of system aspects and their models

The work was presented at the first TACO Industrial Seminar, which took place in Stockholm on the 12th of December 2003. The seminar was hosted by SKI.

1 Introduction

The title of the reported project is "Traceability and Communication of Requirements in Digital I&C Systems Development", abbreviated TACO. The NKS project number is NKS_R_2002_16.

The present report documents the continued research work and related activities carried out within the TACO project in 2003. These activities have constituted a natural continuation of the preproject, and have focused on the technical issues concretised in the preproject report. The work has concentrated on four central and related issues, viz.

- Representation of requirements origins
- Traceability techniques
- Configuration management and the traceability of requirements
- Identification and categorisation of system aspects and their models

The focus of the 2003 activities is reflected in the contents of the different chapters:

Chapter 2 focuses on systems aspects and their modelling, in particular with respect to the integration of systems engineering into (development) process engineering.

Chapter 3 discusses the origins, views, and hierarchical structuring of the collection of requirements, and relates this to the modelling of safety functions and related automation functions.

Chapter 4 provides a discussion on requirements changes with respect to three different lifecycle models: The Waterfall model, the Incremental model, and the Spiral model.

Chapter 5 focuses on *use cases* and their application in requirements elicitation, with an emphasis on their potentials as a common method and language for developers, end users, and domain experts.

Chapter 6 is concerned with requirements traceability, its relationship to software configuration management, and the management of changes. The chapter presents a summary of a TACO case study on the formal specification of fine-grained requirements traceability.

Chapter 7 contains the references in the report.

Appendix A gives an overview of the project organisation and activities.

Appendix B contains the minutes from the First TACO Industrial Seminar, arranged in the premises of SKI, 12th of December 2003.

2 System Aspects

During 2003, the TACO activities on system aspects have focused on requirement engineering and management in general, and integration of systems engineering into (development) process engineering in particular. Due to the fact that knowledge on system aspects and their modelling has a crucial role in suggestions on traceability and communication of requirements in digital I&C systems development, previous efforts on modelling system aspects have been refined and used.

The work started with the argumentation that, in order to find best practices associated with elicitation, analysis (specification), traceability and understandability objectives, guidelines and references for the elicitation of system aspects and for specification (modelling) of those aspects are needed. These guidelines and references should focus on how elicitation and engineering of requirements are done, how design and implementation details can be traced back to the requirements, how easy it is to understand the available documents, etc., The introductory work prior to suggesting a common model of system aspects included study of service-oriented (functional) and quality-oriented (non-functional) system aspects, and study and grouping of available modelling techniques.

2.1 System Aspects and their Associated Modelling Techniques

The terms “functional” and “non-functional” are often used to address two different groups of system aspects. Both terms, however, are used with different meanings in many fields. For example, in cognitive science, the term is used to specify a certain role of something, which in some sense is also adopted by some object-oriented techniques. Within computer science, the term “functional programming” refers to certain properties of a group of programming languages. In the work reported in the present chapter, the terms “service-oriented” and “quality-oriented” are used. “Service-oriented” aspects are those that are directly connected to the core purpose of the system, to its application, to its services offered internally in order for the system to operate, and offered externally in order for the system to stay in use. “Quality-oriented” aspects are those needed to measure the operation and usage quality of the system.

Historically, at least some of the service-oriented aspects have been regarded as natural ingredients of system models. This is not the case for quality-oriented aspects. Studies during recent years have acknowledged this shortcoming, as both groups of aspects should be dealt with and incorporated into the very first stages of the development process.

Generally, six service-oriented system aspects can be identified:

- System goals/requirements,
- System functions,
- System’s unintended roles (can appear as faults/failures),
- Component capabilities,
- Component behaviours,
- Inter-component communications

Capabilities can be regarded as the very basic property of an entity (object, component, system) in order for the rest of the service-oriented aspects to have any meaning. Behaviours can be defined as timed ordering of a group of capabilities, whereas communications can be conceived as spatial ordering of a group of capabilities. Many researchers and practitioners relate the behaviours and communication to the concept of functions. In order to explain the multi-purpose nature of today's computerised systems, however, the functions in this work are not regarded as the core of the system aspects, which is the case for single-purpose systems.

Some of the most known quality-oriented system aspects, sometimes also addressed as "dependability factors/indicators" are listed below.

- Decompositionability
- Comprehensibility
- Usability
- Reusability
- Profitability
- Efficiency
- Flexibility
- Portability
- Reliability
- Robustness
- Maintainability
- Safety
- Security

Minor work was also towards categorisation of different kinds of computerised systems, as well as different kinds of life cycle models/system development processes. This was, among others, in order to demonstrate the relevance of both groups of system aspects listed above.

Some studies were carried out towards identification and categorisation of different modelling techniques and methods. Table 1 shows the result.

Table 1. Function-centred, behaviour-centred and communication-centred specification techniques.

Function-Centred Specification Techniques	***Behaviour-Centred Specification Techniques***	***Communication-Centred Specification Techniques***
➢ Multilevel Flow Modelling ➢ Function Refinement Tree ➢ GoalTree-SuccessTree ➢ Hybrid MFM-GTST ➢ Structured Analysis and Design Technique ➢ Event Response Specification ➢ Declarative & Imperative Specification ➢ Use Case Diagrams ➢ Entity Relationship Diagrams ➢ Class Diagrams	➢ Process Graphs ➢ State Transition Diagrams ➢ Finite State Machines ➢ Extended Finite State Machines ➢ Mealy Machines ➢ Moore Machines ➢ State Charts ➢ Process Structure Diagrams	➢ Data Flow Diagrams ➢ Context Diagrams ➢ Activity Diagrams ➢ Collaboration Diagrams ➢ Statemate Activity Charts ➢ Object Communication Diagrams ➢ System Network Diagrams ➢ Message Sequence Charts ➢ Process Dependency Diagrams

A method was defined as a combination of the techniques for modelling one or several system aspects, and the techniques for using and integrating the models. Associated with the methods

are therefore some method-specific rules of interpretation of the models and some method-specific rules of knowledge communication among the models. How these two groups of rules are formed is crucial to traceability and communication of requirements throughout the system life cycle. For example, if system reliability models are developed by *assumptions* about the behaviour of the system (e.g., due to inherent limitation of the models), then it will be impossible to trace the attainment of a certain reliability degree, if this is one of the requirements of the system.

During these studies, the impression of most "methods" being in fact "techniques" (with no associated use and integration rules) was confirmed. Although most of the techniques identified can be applied in several phases and activities of the development process, it was also observed that there is no formalism or rule behind such applications. As an example, there exists no common rule on how to refine the contents of Message Sequence Charts through design, implementation and test activities of the development process.

2.2 A Common Perception of All Computerised Systems and a Common Model of Their System Aspects

The efforts towards the above are summarised in [22]. The described system perception and common model of system aspects, which at the same time oppose the established practice of stringent separation of system development activities, are believed to play a central role in the traceability and communication of requirements.

2.3 Systematic Integration between Systems Engineering and Process Engineering

The work in this stage is related to Chapter 5 and 6 of the pre-project report. Based on topics addressed in these chapters and questions related to them, a systematic and "formal" integration of systems engineering's techniques and associated models into all phases and activities of the system's life cycle model was proposed.

The topics receiving special attention were as follows:

1. Linking rationale and information sources to the requirements: How to model knowledge and information that usually are not modelled, and how to relate these to the final list of requirements.
2. Functional (service-oriented) system aspects: The discussion on system structure and behaviour and the supporting modelling techniques is hereby attempted to also cover other system aspects, their interrelationships and their supporting modelling techniques.
3. Integration of safety analysis into the software development life cycle: This can be extended to systems analysis with regard to other dependability factors (quality-oriented system aspects) such as security, reliability, and portability. Nevertheless, an important consideration is how to integrate this dependability analysis into all phases of the system life cycle.
4. The relationship between requirement, design and implementation specifications: In the pre-project report, a one-to-one connection between the design and requirement

models is implied. Other approaches, such as one-to-many from the requirements to the design models, might however be considered in order to improve the traceability and communication of the requirements, as well as to ease the change management in case of new design or implementation. Additionally, the validity of a stringent separation of requirement, design, implementation and test specifications needs to be further discussed.

5. Prototyping: How to make prototyping easier and cheaper.
6. Maintainability, requirement and configuration management: Approaches to reduce the level of uncertainties during the maintenance (e.g., whether to incorporate certain changes in the design or implementation, in order to maintain a certain functionality) might be investigated. Equally, approaches to automate configuration management stages need to be addressed.

Figure 1 illustrates the idea of integrating the system's model into the process' model. The phases of the development process decide the level of the focus on one or several system aspects, whereas the disciplines decide the level of refinement of the system model *as a whole*. This means, for example, that although the focus during the implementation and test is mainly on the component capabilities, the entire system model is considered during these activities/disciplines, as these might change some overall functions or requirements as well. The connection between the quality-oriented aspects or dependability factors to the component capabilities indicates that the concept of dependability can eventually be applied to the component capabilities, as they form the focal point in this interrelationship model.

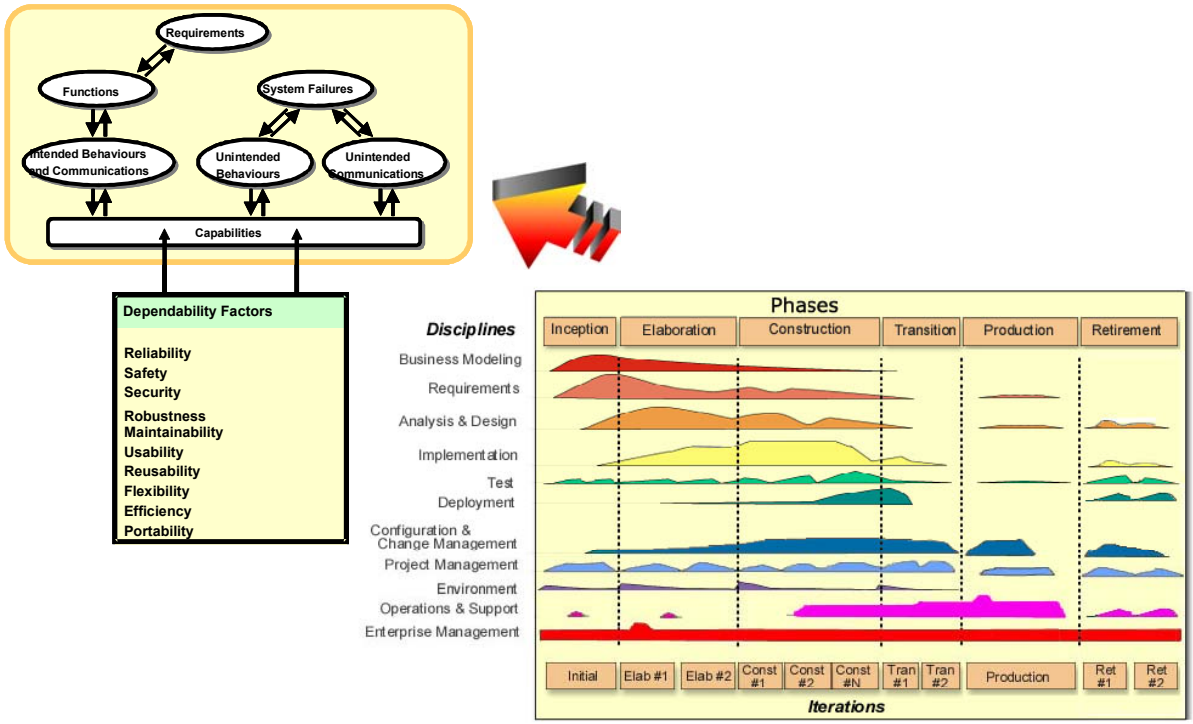

Figure 1. The integration of the system models into the development process.

It has been argued that such an explicit and systematic integration shown above will add to improved traceability and communication of requirements, also from the perspective of forcing the modeller to be more careful and critical about the actual use of various modelling techniques. This will in turn help the modeller to introduce terminologies on how to apply

these techniques in a more distinct manner. Another conclusion to draw from the above is that the arguments for strictly separating requirement, design, implementation and test models will no longer be valid, as their integration will cause no harm but be in harmony with the practical applications, as long as it is performed in an explicit and traceable manner. The modeller is forced to go through all the aspects for each stage, meaning that some implementation-related issues are considered already during the requirement specifications and some design-related aspects are altered during the implementation. As the framework is the same through all the stages, such an alteration will more easily be added to the models in the previous activities.

One of the means for requirements elicitation mentioned in the pre-project report is prototyping. Although prototyping is a very convincing tool, it is in many cases carried out with false or premature assumptions, often in order to reduce the cost bound with it. The ideas presented above will also contribute to a more clear setting around the prototyping and thus a more straightforward development of prototypes. The same arguments can be used for better maintenance, decommissioning and knowledge management, as an improvement of these activities depends on how good the traceability and communication of the requirements is.

Finally, there are also other related matters that might be included. One is the task of risk analysis with regard to, say, safety or security aspects. This normally involves the use of some traditional techniques for modelling faults, failures, deficiencies, etc. While these techniques are helpful in structuring the knowledge and information about the unintended events or system roles, and can therefore help the analysts to develop tools for automatic assessments, they cannot provide more information about the events than the analyst is already aware of. They do not offer ways to define and identify faults and failures, as they are merely techniques used to *model* already identified failures/faults. In order to *identify* these failures/faults, however, an alteration of perspective and fault perception is needed, as the concept of failure is highly related to intentional aspects of a system, and can therefore only be identified/defined/specified through a clarification of the relationships between those aspects. Such a clarification can, for example, reveal the roles of a certain communication pattern that in one system setting (configuration) can be regarded as completely harmless or even intended, while it becomes a severe source of failure in combination with certain behaviour of a software process in another system setting of *the same system*. In other words, during risk analysis, one needs perhaps to draw the attention from what the system should not do to what the system is capable of doing. It is believed that this is the approach to follow in order to reveal hidden risks as well as hidden potentials associated with the system. A better disclosure of these can in turn contribute to better introduction of changes and better change management.

3 Requirements Hierarchy

Ideally the requirements management is a process where requirements are elicited, analysed and specified from general ones towards implementation specific descriptions. During the life cycle of the plant (in case of new nuclear power plants) or the system (in case of I&C upgrading projects), various documents are extracted from the collection of requirements such as overall and detailed requirements specification, safety and reliability assessments, and plans for designated activities [11].

From the knowledge management point of view, the core of the requirement management process is the requirements database where the requirements are stored in a structured way.

There are several ways to build the structure. Figure 2 represents three origins or points of views to the requirements database: 1) user oriented, 2) objective oriented and 3) plant model oriented perspectives.

The user-oriented perspective is classical in the context of requirements management. The textbooks always emphasise the importance of the elicitation of all possible user requirements in the beginning of the requirements management process and the communication with the users in the later phases of the process. Figure 2 distinguishes five typical categories of users in the I&C projects of nuclear power plants: utility (i.e. the plant owner), the operating and the maintenance organisations of the plant, the project organisation responsible for the I&C project and authority (and other users independent of the utility).

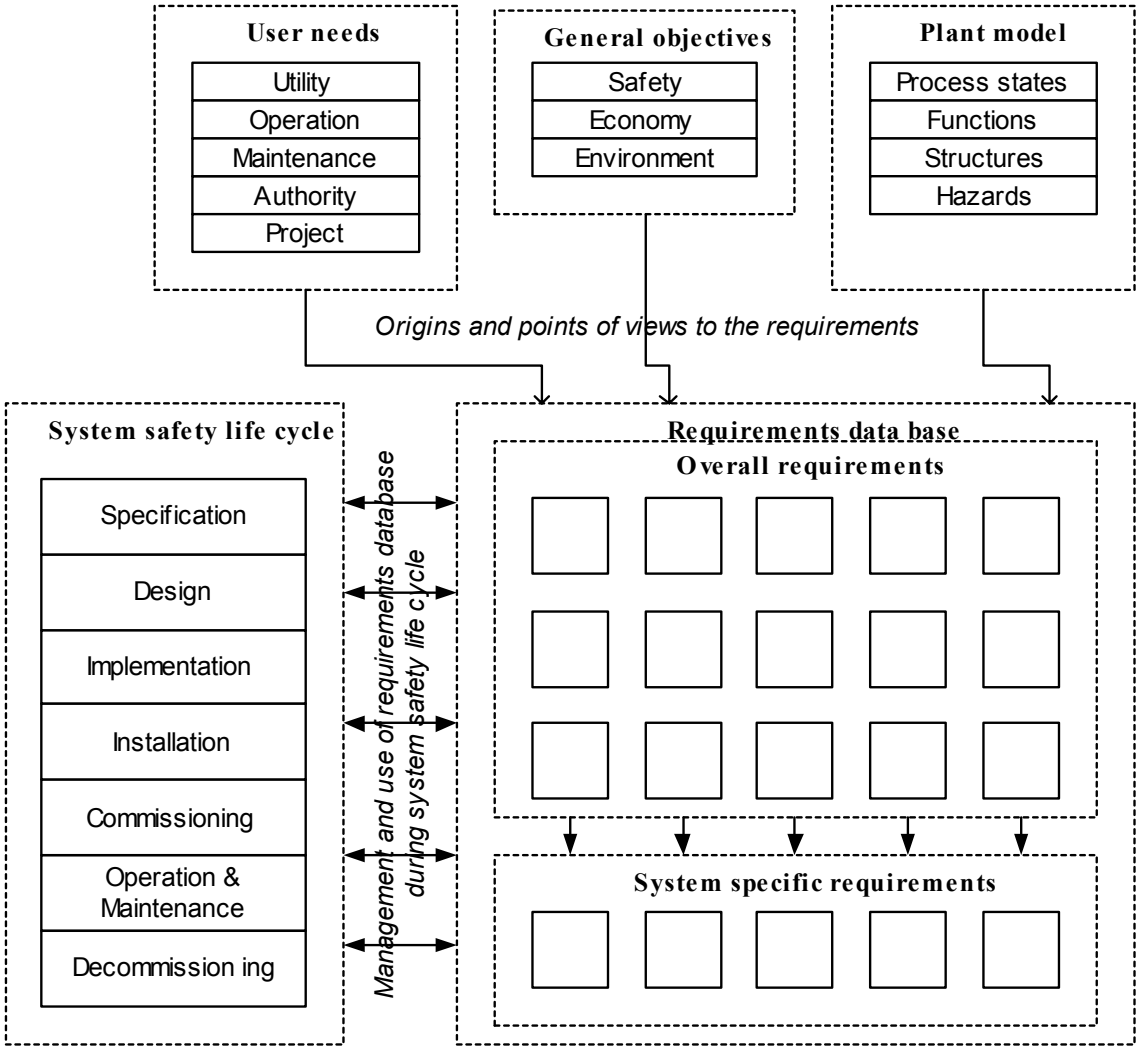

Figure 2. Different perspectives to the collection of requirements. The perspectives form the structure of the requirements database. During the system safety life cycle, the system specific requirements are derived from the overall requirements.

Requirements can be linked to general, user-independent objectives. To maintain safety (e.g. reactor safety, occupational safety), to achieve good economy and to avoid environmental damages are examples of high-level objectives.

In the analysis of requirements for the I&C functions, a model of the plant is needed. Concerning safety-related requirements, the safety analyses provide a model including concepts like

- process states, barriers, defence-in-depth principle, safety critical process parameters;
- safety functions, support functions, automation functions (ordinary process control, preventive, protective, back-up functions);
- plant structures, systems, and components;
- hazards: postulated initiating events, postulated system and component failures.

The plant model links the I&C functions with the process design of the plant.

In the I&C project, the system specific requirements are derived from the overall, implementation independent requirements. The border between the overall and the system specific requirements depends on the scope of the I&C project.

Figure 3 shows an example of a plant model. At the top there are the general classes of requirements from which requirements for the I&C functions and systems are derived. The overall requirements related to reactor safety can be described by relationships between safety functions, barriers and process parameters. Radioactive releases are avoided if the barriers remain intact, which means that the critical process parameters stay within critical limits. For this purpose, safety functions are needed.

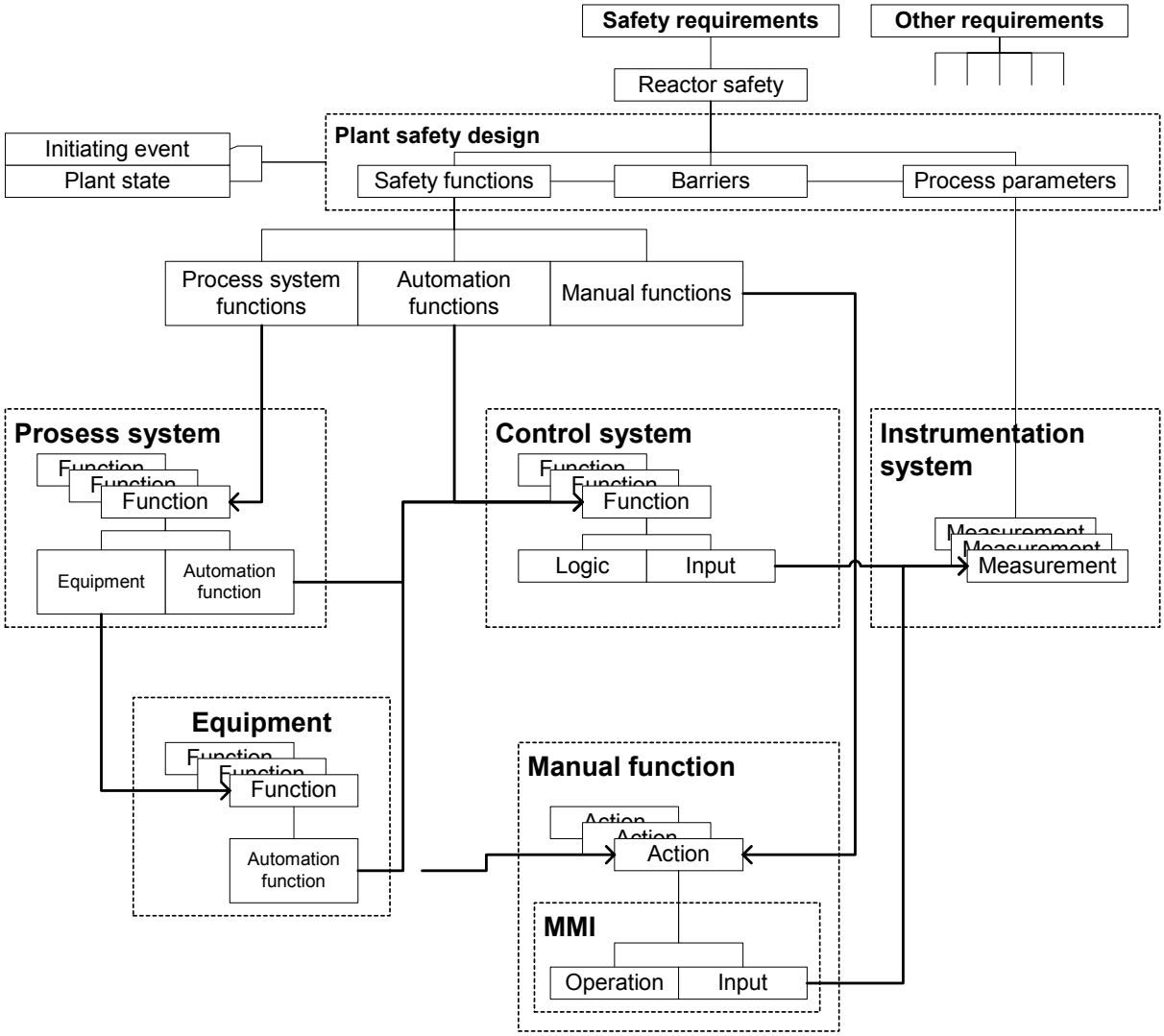

Figure 3. A model of safety functions and related automation functions. A thick arrow describes a reference to a place where the function or implementation is defined more thoroughly. In the requirements analysis process, requirements are defined for each object of the plant model. The structure of the model determines the hierarchy of requirements assigned for the objects of the plant and its I&C functions, systems and equipment.

Safety functions can be divided into process and automation functions. Process systems (i.e. fluid systems) are responsible for process functions and automation systems are responsible for automation functions. Regarding automation functions, control systems can be distinguished from instrumentation systems that have an important role to monitor and send inputs to the automation systems and to the operators performing manual functions.

Plant safety analyses (deterministic and probabilistic) are the basis both for constructing the plant model and for analysing the requirements for the safety-related I&C functions. The fundamental reactor safety objective is the prevention of radioactive releases from the reactor core. Both deterministic and probabilistic approaches analyse the plant response, i.e., behaviour of the safety functions, to postulated initiating events. Figure 4 shows an example of an initiating event, required safety functions to manage the disturbance, sub-functions of one of the safety functions, and one automation function comprising the process measurement part, automation system part and process system equipment part.

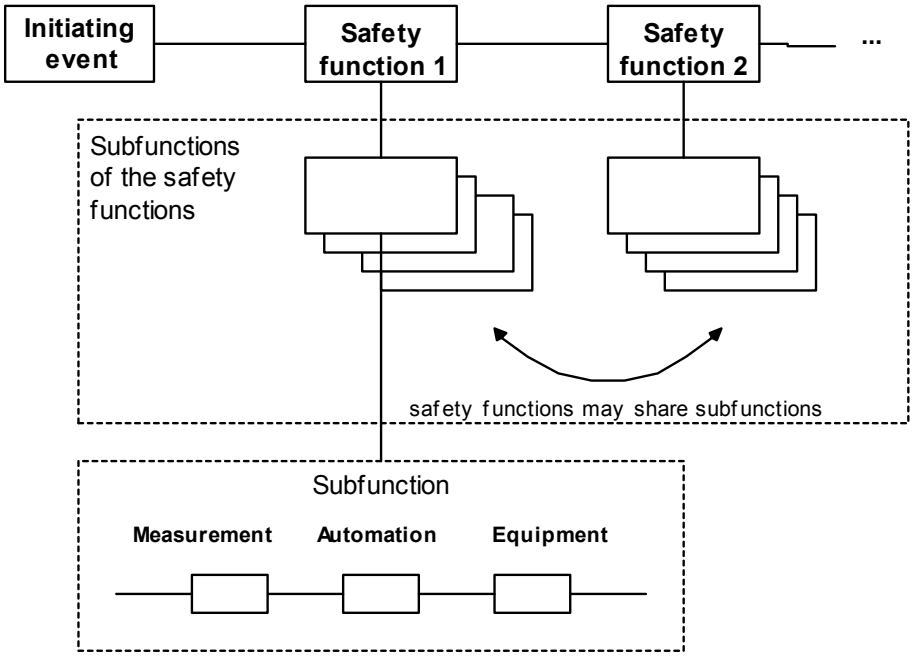

Figure 4. A model of an initiating event and associated safety functions, sub-functions and automation functions. The model aims at identifying process safety requirements for the I&C functions.

In the deterministic approach, the analysis is used to show that sufficient physical margins exist to meet the radiological objectives for all postulated initiating events. The deterministic assumptions include postulations of a single failure or even a single common cause failure in the safety function. The failure can appear e.g. in the measurement part, automation part or in the equipment. Thus one consequence of the deterministic requirements is to require sufficient redundancy and diversity in the safety functions.

In the probabilistic approach, the analysis is used to show that the risk of radioactive release is sufficiently low. In many countries, numerical criteria in the form of core damage frequency and frequency of large early release are stated in the regulatory guides. Failures (single and common cause failures) in the safety functions are assumed with probabilities based on statistics or engineering judgements. Using the probabilistic approach, reliability requirements can be defined for the functions and associated equipment. Then it is up to the utility, project or the designer to decide how to fulfil the reliability requirement.

Even when having a plant model, requirements management of safety related I&C systems is a hard task for several reasons. Firstly, a model is a simplification of the reality and does not represent all interactions of the I&C systems with the process. The model should capture the essential interactions. The postulation of initiating events and failures and the assessment of probabilities in a credible way is a demanding task. Both deterministic and probabilistic safety analyses include uncertainties that should be accounted in the requirements management. Uncertainties may lead to harder requirements on qualification of systems and equipment. Sometimes diversification of the function can be an alternative. Finally, the safety functions have other requirements to be fulfilled, e.g., economical requirements. For instance, a goal is to avoid spurious actuations of a function, which may be a contradictory requirement to the safety requirements. The designer needs to compromise between competing requirements.

4 Life-Cycle Models - Presentation in Taco Industrial Seminar

In the First Taco Industrial Seminar (see appendix B), requirements changes and life-cycle models were discussed. In practise, requirements are often changing along the project until the very end. According to authorities, the requirements specification should be frozen at some point, but it is hard to decide when. Some requirements may be unknown or unambiguously defined during the whole project, which makes determining the freezing point very difficult. On the other hand, implementation of the system is much easier if there is a frozen requirements specification available. After freezing, requirements changes may cause remarkable rework and additional cost.

In the seminar, some life-cycle models were presented as an introduction to the discussion. The traditional Waterfall model is presented in Figure 5 below. It is probably the best known and most used model and it is worth noticing that it was first time introduced already more than 30 years ago by W. W. Royce [16]. In general, it is not very well known that Royce included iterations in the model in addition to basic flow from the system requirements to operations (as can be seen from the figure). Because the process is quite straightforward, the Waterfall model is easy to manage. In a small project, Waterfall model is regarded cost effective if the requirements are well known [15].

The Waterfall model has also some weaknesses: it is inflexible to partition the project into distinct stages. It is also difficult to respond to changing requirements because it is so straightforward. Feedback is available very late and changes can become very expensive.

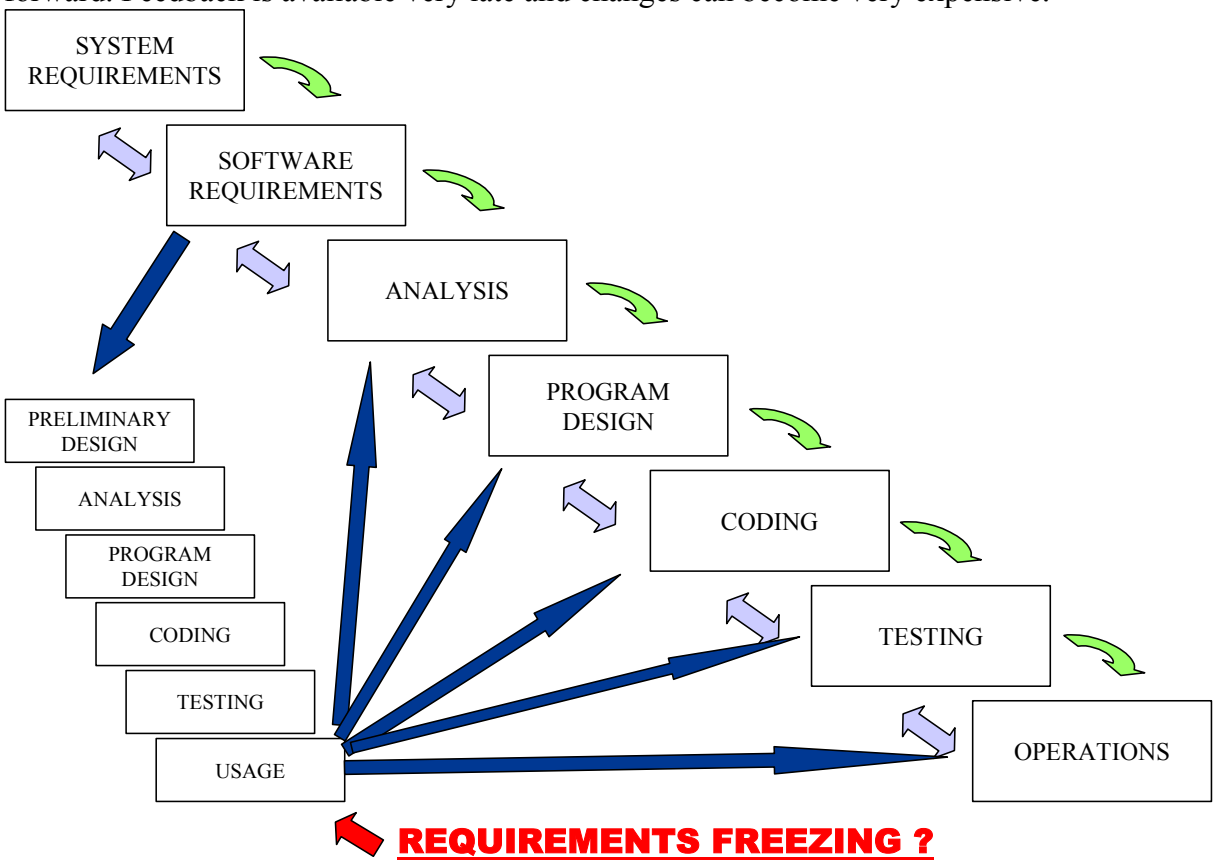

Figure 5. Waterfall model (modified from [16]).

Another model presented in the seminar was the Incremental model. Figure 6 below introduces the process steps. The model functions as follows:

- Divide the project into increments.
- Each increment adds functionality.
- Prioritise or classify requirements.
- Include high-priority requirements into early increments.
- Freeze requirements during each increment.
- Evolve the remaining requirements.

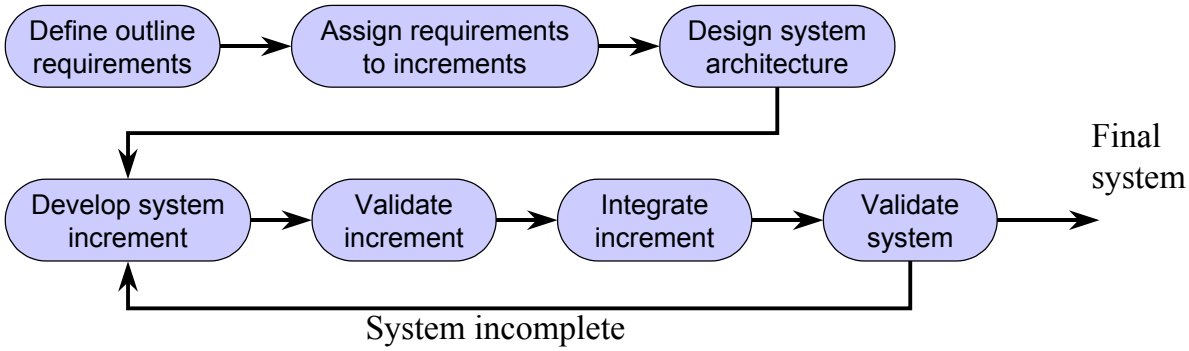

Figure 6 . Incremental model (modified from [15]).

Because of increments, system functionality is available earlier than in the Waterfall model. Early increments can be used as sort of prototypes, which helps in understanding the problems and possibilities. Increments also give feedback on system performance and it is easy to elicit more requirements to later increments. Because feedback comes along the project, the risk of overall project failure is low. It is also possible to set the highest priority to the most important parts of the system.

Incremental development is harder to plan and control than Waterfall development. It can also be more expensive than Waterfall development and it requires experienced people. There is always a risk that changes cause renegotiations with the customer [15].

The most sophisticated model, Spiral, is presented in Figure 7. When using the Spiral, the first thing to do is to determine objectives, different alternatives and constraints for the problem at hand. Then those alternatives and constraints are evaluated and risks are analysed. After that, the next level requirements (or the next level system or product) is designed, developed and verified. Before starting the next round in a spiral, the development plan is updated and future tasks are planned at a more detailed level.

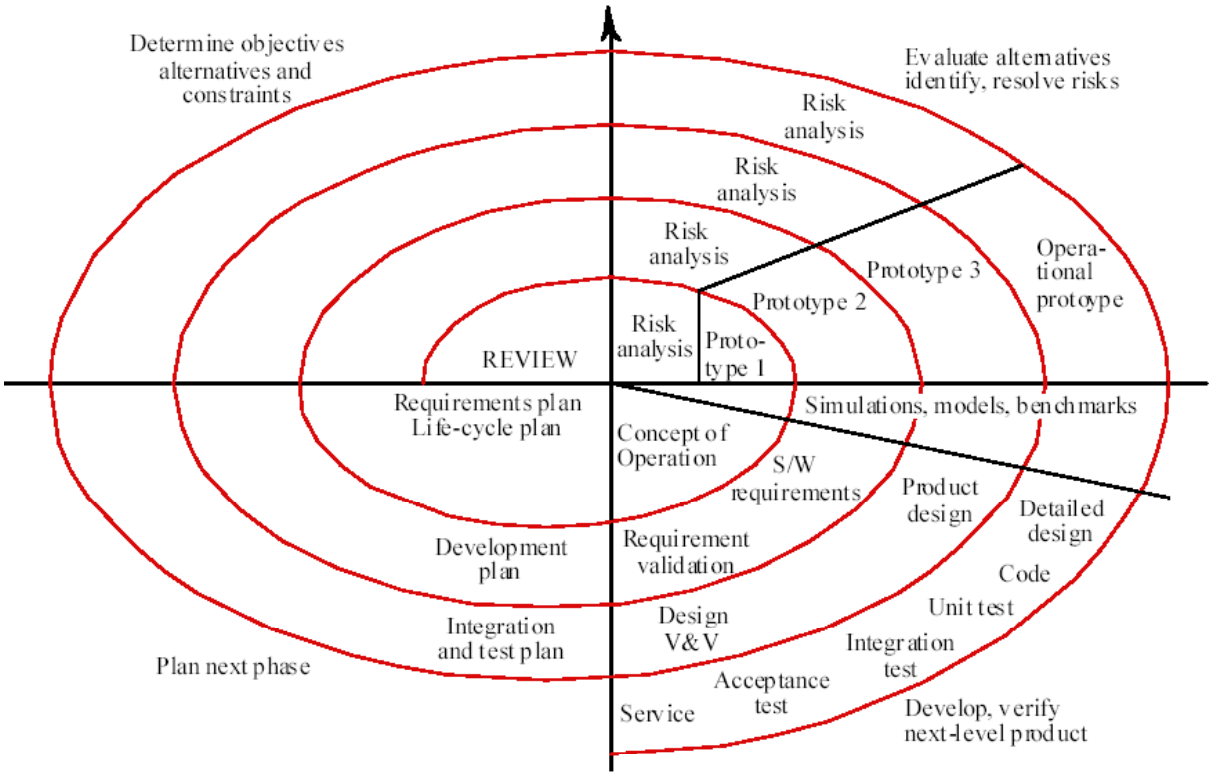

Figure 7 . Spiral model [5].

In a spiral model, design flexibility allows changes to be implemented at several stages of the project. Because systems are built up in small segments, it is quite easy to do cost calculations. The spiral model also involves the client to the project and he has a good control over the direction and implementation of the project. Also the client's knowledge of the project grows as the project grows. This makes it easier to give early feedback about the development process.

Because risk-assessment plays a remarkable role in the first segment of the spiral, utilising it demands considerable risk-assessment expertise. The Spiral model has not been employed as much as some other models (e.g. Waterfall), which makes it difficult to get information on how developers have experienced it. Performing the steps of the spiral requires a lot of elaboration and also some experience. Otherwise it may be inconsistent to use.

5 Use Cases in Requirements Elicitation

5.1 Requirements Elicitation Problems

Traditionally, requirements have been presented as a long list of items containing expressions “the system shall”. However, these lists may have some serious flaws such as appearance of duplicate and conflicting requirements and lack of structure (hierarchy). Requirements elicitation is often regarded as the first step in the requirements engineering process. The goals of requirements elicitation include finding out what the problem to be solved actually is and finding out what is expected from the system under development. This may sound easy, but in

practise, requirements elicitation can be a difficult task. [18] discuss what they consider to be the most important problems that hinder the identification and definition of the user's needs:

- *Poor communication*: Signs of lacking interest in the customer's needs, contradiction with the intended message, inadequate and indirect communication, failure to match the customer's communication preferences, etc.
- *Resistance*: Physical expression of opposition to the development project e.g. due to the envisaged changes implied by a particular technological solution.
- *Articulation/expertise problems*: All stakeholders don't understand how the terminology is used. There may also be intractable complexity, patronizing attitudes to other parties, etc.
- *Problem perspective differences*: Technology-driven development is not based on the customer's actual needs, there is insufficient knowledge of recent technology improvements, etc.

Also [12] present the following barriers to requirements elicitation:

- The "Yes, But" Syndrome: this refers to the reaction of the users when they, after months of waiting, get the ready system in front of them. When they start using it, they think it is a great system… but. That "but" comes from the fact that users have not been included in the development process from the beginning and therefore the system is not what they expected.

- The "Undiscovered Ruins" Syndrome: it is difficult to determine when you are done with requirements elicitation because the more that are found, the more you know remain.

- The "User and the Developer Syndrome": there is a communication gap between users and developers. These groups may be from different worlds, they speak different languages, use different terms of same things, and they have different backgrounds, motivations, and objectives.

To mitigate the above-mentioned barriers, there are a number of different techniques that can be applied to requirements elicitation, for example [19]:

- Interviews and questionnaires
- Scenarios
- Requirements workshops
- Brainstorming and idea reduction
- Storyboards
- Observation
- Use cases
- Role playing
- Prototypes

Which technique to deploy depends on various factors, such as application type, skills of the developers and the customer, the scale of the problem, and the technology used.

5.2 What Are Use Cases

Use cases have many benefits that have made their deployment popular: they are simple, informal, and rather easy to write. Also their graphical notation is quite trivial. However, labelling use cases an easy technique can be deceptive. Producing a set of use cases is not difficult but producing an appropriate well-formed use case model may be much harder [6]. Engineers typically have had different approaches to use case modelling. Applying use cases into practice often brings out problems that are not clearly presented in the literature. It is also easy to misuse use cases. Producing high quality use cases requires practice and skill and one should never assume that anyone could read and interpret use cases without proper training.

Use cases are used for capturing the behaviour of the system to be developed. They serve both as a common method and language for developers, end users, and domain experts to elicit system's functional requirements. Ideally, the goal is to find out and describe all the possible ways of how an external entity (human, another system) can use the system. Use cases can be illustrated in various ways, both textually and graphically (usually these go hand in hand).

A use case describes a sequence of actions, performed by a system, that yields a valuable result to a user. Use cases treat the system as a black box, just like ordinary end users see computers. Use cases should show what the system does for users, not how. They also present the interaction of actors and the system. An actor represents a set of roles that users play when they interact with these use cases. Typically, an actor represents a role that a human, a hardware device, or some other system plays in the system. A single actor may perform many use cases and a use case may have several actors performing it. Actors are not actually part of the system - they live outside the system [6].

5.3 Use Case Modelling

Use cases may be represented both textually and graphically. The graphical notation is called the use case diagram. Use case diagrams are important for visualising, specifying, and documenting the behaviour of an element and they are also part of the Unified Modelling Language (UML) [6]. The use case description is a textual presentation of the use case. UML does not specify any standard for the format of the description.

5.3.1 Use Case Diagrams

A simple use case diagram is illustrated in Figure 8. The actor is depicted with a stick figure and the use case is presented as an ellipse. The actor in the example plays the role of an 'Engineer' and the use case is called 'Create report'. The arrow from the actor to the use case is called an association, which indicates that both the actor and the use case can receive and send messages. The direction of the arrow indicates who is at the receiving end of the communication. A line without an arrow means that the association is bi-directional.

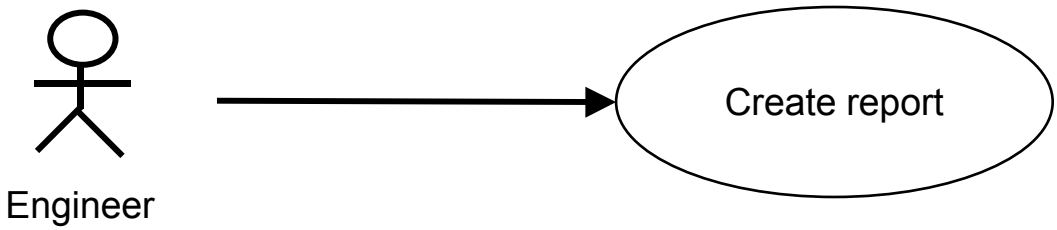

Figure 8 . A simple use case diagram.

In addition to associations between actors and use cases, relationships between use cases can be depicted. These are the *uses* and *extends* relationships among use cases. *Extends* relationship is used when a use case is similar to another use case but does a bit more. *Uses* relationship occurs when you have a chunk of behaviour that is similar across more than one use case and you don't want to keep copying the description of that behaviour [10].

The associations presented above are useful because they give possibilities to customise diagrams according to the project's needs. However, it must be kept in mind that the more these associations between use cases are used, the harder the diagrams become to read for users. That doesn't serve the original idea of use cases to be a simple communication method between designers and end users.

5.3.2 Use Case Descriptions

The textual description of a use case can be customised according to the purpose. However, there are some sections that are commonly suggested to be included. A use case description template could contain the following information:

- Name of the use case: descriptive name.
- Summary: a few sentences describing the use case.
- Participants: which actors will be involved?
- Prevailing conditions: what kinds of conditions are prevailing when starting?
- Description: informal description, also diagrams may be used.
- Exceptions: description of exceptions.
- Triggers: when or why the actors will enter this use case?
- Resulting conditions: what conditions are prevailing when the use case is ended?
- Other requirements: other non-functional requirements.
- Author: who wrote this use case?
- Modification history: who, what, when, why?

As the template should be quick and easy to use, only the most necessary information should be included to the description.

5.4 How to Utilise Use Cases

Use cases are an essential tool at requirements capturing and in planning and controlling an iterative project [10]. They can be utilised for any system interacting with the outside world. Use cases are useful in communication between developers and users. They can also be used as part of the contract between customers and developers regarding the functionality of the system [2].

Use cases are good in describing functional behaviour and functional requirements. Some other methods may suit better for describing quality attributes such as extensibility, availability, portability, reliability, and reusability. However, utilising use cases is a great way to document use cases for any system and they can be used also with non-object-oriented systems.

5.5 Further Work within Use Cases in TACO project

Until now, we have familiarised with use cases mostly at a general level. As we now know the basic principles about use cases, the next step is to try to utilise and adapt them at a more practical level. The aim is to find out if an object-oriented approach using UML diagrams, particularly use cases, could be used for this purpose. In UML, one object may be part of several diagrams, and one could therefore try to see the different hierarchies from the perspective of the object instead of combining the hierarchies. The other point is to research if safety analysis reports could utilise the concept of use cases, and to a certain extent be structured accordingly.

6 Requirements Traceability

During 2003, the TACO activities on requirements traceability have partly focused on identifying some basic concepts related to requirements traceability: definitions, tools and techniques – and how these can be used to facilitate traceability documentation and management.

The definitions identified include, but are not limited to:

- Requirements traceability - the ability to describe and follow the life of a requirement, in both a forwards and backwards direction.
- Traceability - the degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another.
- Consistency - the degree of uniformity, standardisation, and freedom from contradiction among the documents or parts of a system or component.
- Completeness - the degree to which all the parts of a software system or component are present and each of its parts is fully specified and developed.

Based on the basic definition of requirements traceability, as the ability to describe and follow the life of a requirement in both a forwards and backwards direction, four different types of traceability links can be identified, utilising both forward and backward direction tracing. For convenience, we will here repeat from description of the different types of tracebility links, given previously in the TACO preproject report.

The different links that can be created for requirement tracing are: Forward from the requirements, Backward to the requirements, Forward to the requirements and Backward from the requirements [9]. See figure below.

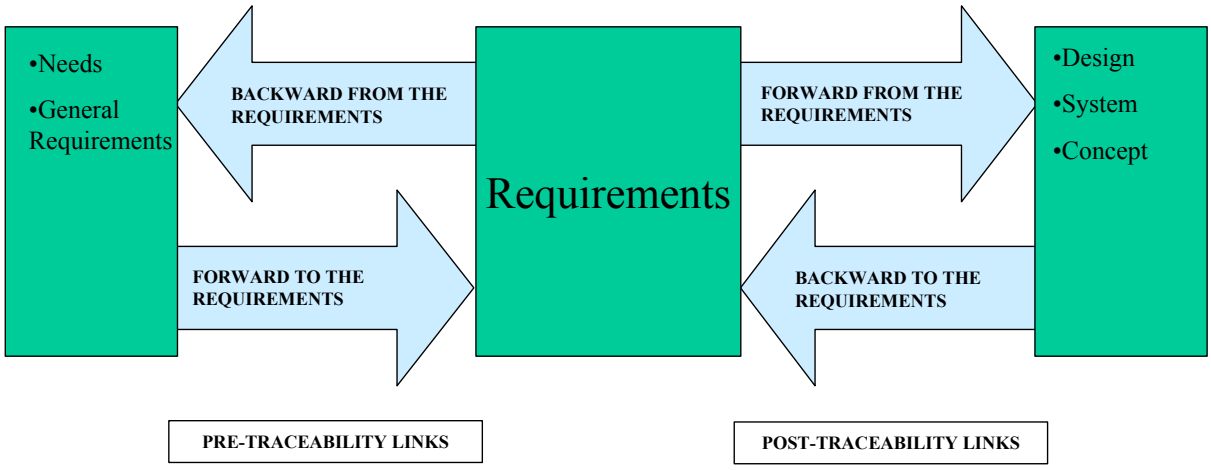

Figure 9. Traceability links.

The link "Forward from the requirements" can be described by assigning a requirement to one or more system components. In this way components are made "responsible" for the requirement. The links are good for evaluating impacts of requirement changes.

The link "Backward to the requirements" is done when the compliance of the system or a system component is mapped back to the requirements. This type of link shows that the requirement has been tested or verified for the system.

The link "Forward to the requirements" can be described by taking the stakeholders needs and technical assumptions and linking them to a requirement or requirements. If the needs or technical assumptions change, then the requirements can be analysed for impact.

The link "Backward from the requirements" gives the ability to validate the requirements of the stakeholders needs or technical assumptions.

By examining these links more closely, they can be broken down into two different types. The first two traceability links can be considered post-traceability links. They link requirements to design and implementation. They also document responsibility assignment, compliance verification, and the impact analysis of a requirement. The other two links are pre-traceability links. It means that they capture the essence or the meaning of how the requirements are created.

An interesting possibility for further research within this area is the use of TopicMaps, within the document centred approach. In general, TopicMaps enable multiple, concurrent views of sets of information objects. Classification of tools that can be used for supporting requirements traceability, and traceability metrics related to the products and processes involved at the software and systems level have also been identified as key areas.

Based on input from the First TACO Industrial Seminar in Stockholm 2003, areas such as identification of key properties and knowledge on how to use traceability tools, and how to address change and rework as a consequence of requirements change, will be evaluated as areas for further work.

6.1 Configuration Management and the Traceability of Requirements

Part of the TACO work on traceability has focused on the role of configuration management in ensuring traceability of requirements. Management of configurations and of traceability are closely related, in particular with respect to the management of changes. In general, configuration management and requirements traceability management should be part of the same approach, in the sense that the latter should be part of the first.

6.1.1 The Management of Requirements Changes

An important part of requirements management is the management of changes. Typically, the requirements for a given system undergo many changes before the development is completed. These changes may be due to changes in the prospected operation environment, but may also happen simply as a result of improved insight during the development. The task of managing changing requirements is closely related to requirements traceability. In fact, work on requirements traceability can to a certain extent been seen as a response to the need for keeping track of these changes. As is stated by [17], one benefit of traceability is the localization of the side effects of a modification and the identification of relationships that must be reconfirmed, thereby increasing the assurance that when changes are necessary they will be complete and consistent.

6.1.2 Software Configuration Management

There are at least two different ways to view the configuration of a digital I&C system. Following [7], the configuration of a system can be thought of as the function and/or physical characteristics of hardware, firmware, software or a combination thereof as set forth in technical documentation and achieved in a product. Alternatively, it can be thought of as a collection of specific versions of these items combined according to specific build procedures to accomplish a particular purpose. In the latter case, the purpose of configuration management is to identify the different configurations of a system in order to facilitate change management and the maintenance of integrity and traceability throughout the system life cycle. This is the perspective advocated in e.g. [4], and the one emphasised in the TACO project.

In general production and operations management, configuration management is a system by which a product's planned and changing components are accurately identified and for which control and accountability of change are maintained. In relation to software development, project documents as well as code can be put under configuration control. Following [1], also tools used to produce, examine, test and independently assess the system should be controlled under configuration management. This includes demonstrating that "the intended versions are those actually deployed, and that those versions have been subjected to known and acceptable levels of verification and validation".

The development of digital systems involves the identification of major system segments and the high-level hardware and software elements of each segment. Each element so identified is referred to as a configuration item (CI). The management of these items is an important part of the requirements management. According to [3], configuration management for software is defined as a technical and administrative discipline that performs the following functions:

- identifying and documenting the functional and physical characteristics of configuration items,
- controlling changes to them through the system life cycle,
- recording and reporting their implementation status and the status of change requests, and
- verifying their completeness, correctness, and compliance with requirements.

[23] states that, for safety system software, a minimal set of configuration management activities must accomplish the following:

- identification and control of all software designs and code;
- identification and control of all software design interfaces;
- control of all software design changes;
- control of software documentation (user, operating, and maintenance documentation);
- control of software vendors supplying safety system software;
- control and retrieval of qualification information associated with software designs and code;
- software configuration audits; and
- status accounting.

An overview of the knowledge area of software configuration management (SCM) is presented in [20].

6.1.3 Software Architecture

Software configuration management is an important element in the management of the software process, where the designs as well as the design process itself are integrated. In this context, [14] re-examines the role of software architecture and its relationship to requirements, design, and implementation, i.e. as the framework for satisfying requirements and as the technical and managerial basis for design and implementation. According to [14], "*architecture* is concerned with the selection of architectural elements, their interactions, and the constraints on those elements and their interactions necessary to provide a framework in which to satisfy the requirements and serve as a basis for the design". On basis of this relationship, two important categories of analysis are with respect to consistency and dependency:

- *Consistency analysis*: Consistency within the architecture and architectural styles, of the architecture with the requirements, and of the design with the architecture.

- *Dependency analysis*: Dependencies between architecture and design, and between architecture and requirements, in particular with respect to implications of changes.

The role of architecture in the design processes is important to a proper understanding of design patterns and reuse. In particular, architectures provide patterns that guide the composition of components into systems [21]. By properly distinguishing requirements, architecture, design, and implementation, the architect, designer, or implementer may be able to select the appropriate architectural element, design, element, or implemented code to satisfy the specified constraints at the appropriate level [14]. By way of example, the design of a component may be reusable even if its implementation is not. The solution is to check the constraints of

the design and the implementation against the requirements of a particular architectural element and choose the proper level accordingly.

As is stated by [8], software configuration management may benefit from software architectures in two ways:

- Architecture-oriented SCM will improve the software process through a high-level product description. Such high-level descriptions are emphasized in software architecture.

- The design of an appropriate architecture of a SCM system requires a clear understanding of the relations between the SCM system and other software components.

The disciplines of software architecture and software configuration management overlap because they are both concerned with the structure of a software system being described in terms of components and relationships. Software configuration management is concerned with the management of all kinds of software objects created throughout the software life cycle. These are arranged into software configurations by means of various kinds of relationships such as hierarchies and dependencies. The software objects are however usually treated at a low semantical level, i.e. as "black boxes". Clearly, this is not sufficient for managing the evolution of a software system, which may involve semantic changes at the architectural level. As is maintained by [24], the evolution of software architectures has to be dealt with at two levels: at the architectural level and at the SCM level. These two levels complement each other and cannot be unified easily.

As is argued by [24], the software process should be requirements-centered (or user-centered). In particular, software configuration management should support traceability from the initial or negotiated requirements to the finally delivered code. In contrast, architecture-centred software processes tend to overemphasize the importance of software architecture and neglect requirements engineering.

Recently, object-oriented modelling languages, in particular UML (the Unified Modeling Language), have been used both for expressing software architectures and for supporting requirements engineering.

6.2 A Case Study on Fine-Grained Traceability of Requirements Changes

In relation to the TACO research activities on requirements traceability, a case study has been initiated that demonstrates how fine-grained traceability can be specified, animated, validated, and automatically implemented by using the algebraic specification language HALDEN ASL and the associated tool HALDEN Prover, developed at IFE. In the present report, only an overall description of this work is given. The case study is described in detail in the TACO document TACO-013, which is available on demand.

The formal specification of fine-grained traceability can be used directly in the implementation of a simple, generic configuration management system. The objects of interest can be of different kinds, but are in the TACO project considered to be requirements in the development of a digital I&C system. The objects are called *paragraphs*, represented as pairs of para-

graphs labels and version labels. The case study is adequate also for other purposes and for experimentation with other methods, and can easily be extended or modified to cover other aspects of traceability and configuration management. By way of example, a similar approach can be adopted in handling complex sets of requirements given in technical standards.

The general idea to fine-grained traceability is based on [13]. Accordingly, the following set of possible changes to a requirement are considered:

- *add*: create a new paragraph with no prior history;
- *delete*: delete a paragraph;
- *split*: create a number of new paragraphs by splitting an existing paragraph;
- *combine*: create a new paragraph by combining existing paragraphs;
- *replace*: replace existing paragraphs by a new paragraph;
- *derive*: derive a new paragraph from existing paragraphs;
- *modify*: modify a paragraph without changing it's meaning.

The different types of changes can be illustrated in terms of the evolution of paragraphs given in Figure 10.

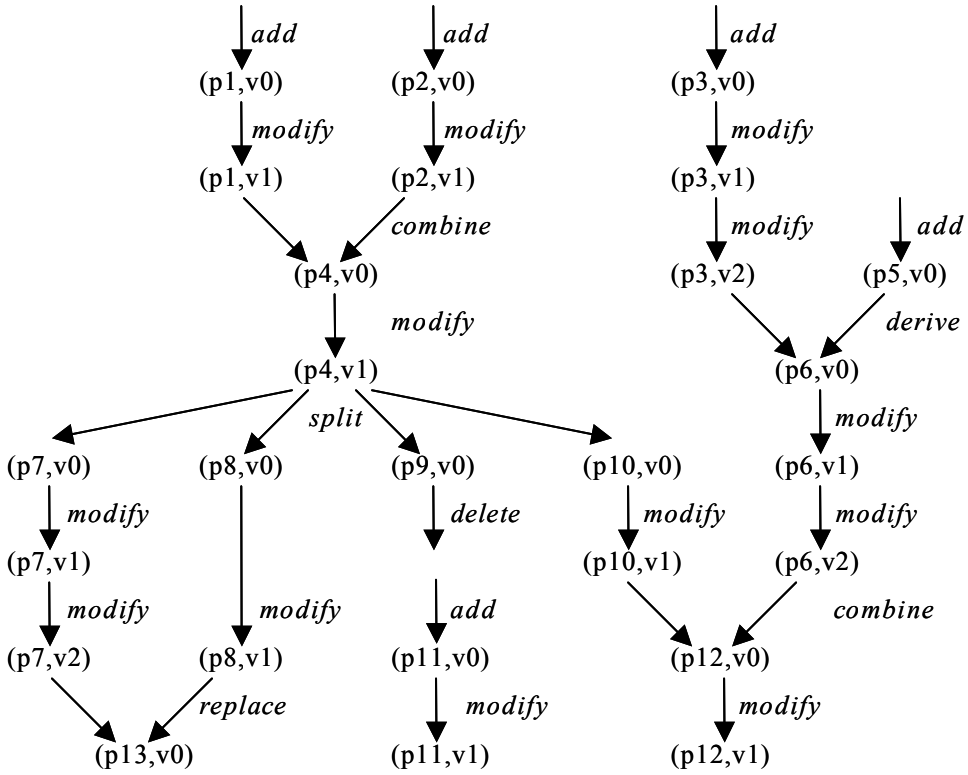

Figure 10. The example change history.

The development of the requirements starts with the introduction of the paragraphs p1, p2, and p3. At later stages, another two new paragraphs are introduced, viz. p5 and p11. All the other paragraphs are developed on basis of these five paragraphs. Paragraphs p1 and p2 are first modified and then combined into a new paragraph p4. After a modification, this paragraph is split into four separate paragraphs p7 to p10. The latter of these paragraphs are modified and then combined with p6, originally derived from paragraphs p3 and p5, giving para-

graph p12. Note that, at any point in the development of the paragraphs, at most one version of a paragraph is applicable (in the sense that it is the valid version of the paragraph).

The requirements change management is specified in two layers, reflected in a hierarchy of two specifications. In the "lower" specification, the different change types are specified inductively as generators, thus providing a data structure for the change history. At the top of this specification, the different change types are specified as operators, checking the validity of the given change and producing a representation at the lower level together with the set of applicable paragraphs. By applying these operators, only valid change histories will be represented.

By the end of 2003, the case study covers the formal specification of the following aspects of fine-grained traceability:

- *The paragraphs*: The basic objects, or *configuration items*, specified as pairs consisting of a paragraph label and a version label. In the set of applicable paragraphs for a given configuration, each paragraph label is associated to at most one version label.

- *The change types*: The different kinds of changes constituting the possible elements of a change history, or *configuration*. By specifying the change types inductively, each change history is represented as a single term representing the changes involved, ordered "chronologically". An important advantage with this approach is that the same term represents both the configuration itself and the history development history that led to this configuration. Accordingly, it is possible to distinguish between different development histories leading to the same set of applicable paragraphs.

- *The initial and deleted paragraphs*: The operations of finding all paragraphs that are initial to a given change history, and all paragraphs that were deleted during a given change history. The initial paragraphs play the important role as initiators for the development of paragraphs. These are typically replaced during the course of the development, and are therefore not necessarily included in the final set of applicable paragraphs.

- *The applicable paragraphs*: The paragraphs that actually apply in a given configuration. Since a paragraph may be deleted, replaced, etc., not all paragraphs introduced necessarily survive. Modifications also imply that a new version of a paragraph may be introduced to replace the existing version. Finding the set of applicable paragraphs is in general a non-trivial problem, especially when the number of paragraphs is large or many changes are undertaken.

- *The history of a paragraph*: The changes related to a given paragraph for a given configuration. A paragraph has its own history in the sense that it may go through creation, modification, combination, etc., corresponding to the different change types.

- *Backwards traceability*: The minimum subset of the given change history that leads to a given set of paragraphs, i.e. the minimum fragment of the change history that has influenced the development of the given paragraphs.

- *Forwards traceability*: The development of paragraphs starting with one or more of the paragraphs in a given set of paragraphs of interest.

- *Validity of changes*: The criteria for the validity of the different change types, i.e. whether or not proposed changes should be considered legal and meaningful, specified in terms of predicates that define the conditions that determine whether or not a given change is valid for a given configuration.

- *Legal change histories*: The criteria for a change history to be considered legal, specified inductively on the recursive structure of the terms representing the change histories.

Further work on the case example will focus on

- automated proofs of invariant properties of the requirements change management;
- automated implementation of the specification into a prototype tool;
- extension of the case example to cover other aspects of the requirements management;
- Generalization of the results to other types of management, such as document management, knowledge management, etc.

7 References

[1] AECB, DSIN/IPSN, NII, USNRC, Four party regulatory consensus report on the safety case for computer-based systems in nuclear power plants. Health & Safety Executive, Nov. 1997.

[2] B. Anda and D. Sjoberg, 2002, Towards an inspection technique for use case models, in: *Proc. 14th International Conference on Software engineering and Knowledge Engineering*, ACM International Conference Proceeding Series, ACM Press, 2002, pp. 127-134.

[3] ANSI/IEEE, *IEEE Standard Glossary of Software Engineering Terminology*, ANSI/IEEE Standard 610.12-1990, IEEE, Piscataway, New Jersey, 1990.

[4] E. H. Bershoff, Elements of software configuration management, *IEEE Trans. Software Engineering*, **SE-10**, 1 (January 1984), pp. 79-87.

[5] B. W. Boehm, A spiral model of software development and enhancement, *IEEE Computer*, **21**, 5 (May 1988), pp. 61-72.

[6] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

[7] F. Buckley, *Configuration Management: Hardware, Software, and Firmware*, IEEE Press, 1993.

[8] R. Conradi and B. Westfechtel, Version models for software configuration management, *ACM Computing Surveys*, **30**, 2 (June 1998) pp. 232-282.

[9] A. M. Davis, The analysis and specification of systems and software requirements, in: *Systems and Software Requirements Engineering,* IEEE Computer Society Press, 1990, 119-144.

[10] M. Fowler, and K. Scott, *UML Distilled, Applying the Standard Object Modeling Language,* Addison-Wesley, 1998.

[11] IEC 61513/FDIS. 2000, Nuclear power plants — Instrumentation and control for systems important to safety — General requirements for systems. International Electrotechnical Commission.

[12] D. Leffingwell and D. Widrig, *Managing Software Requirements*, Addison-Wesley, 2000.

[13] P. Lindsay and O. Traynor. Supporting fine-grained traceability in software development environments. Technical Report No. 98-10, Software Verification Research Centre, School of Information Technology, The University of Queensland (July 1998).

[14] D. E. Perry and A. L. Wolf, Foundations for the study of software architecture, *ACM SIGSOFT Software Eng. Notes*, **17**, 4 (Oct. 1992) 40-52.

[15] K. Rautiainen, Process models, Lecture in a course *Software processes* in Helsinki University of Technology, 2003.

[16] W. W. Royce, Managing the development of large software systems, in: *Proc. Wescon*, 1970, pp. 1-9.

[17] A. Saeed, R. de Lemos, and T. Anderson, On the safety analysis of requirements specifications for safety-critical software, *ISA Transactions*, **34** (1995), pp. 283-295.

[18] H. Saiedian and R. Dale, Requirements engineering: making the connection between the software developer and customer, *Information and Software Technology*, **42** (2000), pp. 419-428.

[19] P. Sawyer and G. Kotonya, Software requirements, in: *SWEBOK Trial version 1.0*, http://www.swebok.org.

[20] J. A. Scott and D. Nisse, Software configuration management, in: *SWEBOK Trial version 1.0*, http://www.swebok.org.

[21] M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, and G. Zelesnik, Abstractions for software architecture and tools to support them, *IEEE Trans. Software Eng.*, **21**, 4 (April 1995) pp. 314-335.

[22] A. P.-J. Thunem, Modelling of knowledge intensive computerised systems based on Capability-Oriented Agent Theory (COAT), in: *Proc. International IEEE Conference on Integration of Knowledge Intensive Multi-Agent Systems*, *IEEE-KIMAS'03*, Cambridge (MA), USA, September 2003, pp. 58-63.

[23] U.S. Nuclear Regulatory Commission, Regulatory Guide 1.169 - Configuration management plans for digital computer software used in safety systems of nuclear power plants (Sep. 1997), http://www.nrc.gov.

[24] B. Westfechtel and R. Conradi, Software architecture and software configuration management, in *Proc. SCM 2001/2003*, LNCS **2649**, Springer-Verlag, 2003, pp. 24-39.

8 Appendix A: Project Organisation and Activities

8.1 Project Organisation

The project is coordinated by Terje Sivertsen (IFE), and comprises the following organisations and persons:

Organisation	Address	Project participants
IFE	Institute for energy technology P.O. Box 173 NO-1751 Halden Norway	Terje Sivertsen +47 69 212403 (terje.sivertsen@hrp.no) Rune Fredriksen +47 69 212430 (rune.fredriksen@hrp.no) Atoosa P-J Thunem +47 69 212322 (atoosa.p-j.thunem@hrp.no)
VTT	VTT Industrial Systems P.O. Box 1301 FIN-02044 VTT Finland	Olli Ventä +358 9 456 6556 (olli.venta@vtt.fi) Janne Valkonen +358 9 456 6469 (janne.valkonen@vtt.fi) Jan-Erik Holmberg +358 9 456 6450 (jan-erik.holmberg@vtt.fi) Teemu Tommila +358 9 456 6457 (teemu.tommila@vtt.fi)
Ringhals AB (Barsebäck Kraft)	Barsebäck Kraft P.O. Box 524 SE-246 25 Löddeköpinge Sweden	Jan-Ove Andersson +46 46 724148 (jan-ove.andersson@ringhals.se)

The project coordinator is responsible for organising the work within the project and for directing it towards its objectives. This includes

- project planning and tracking;
- establishment and maintenance of the project archive;

- establishment of good communication and cooperation within the project;
- reporting to NKS;
- coordination of activities, in particular the production of the project deliverables;
- follow up of meetings and decisions;
- securing of proper quality control, including review and approval of documents included in the project archive;
- reporting of deviations and implementation of agreed corrections.

All the individual participants represent important parts of the technical competence within the project, and are responsible for contributing to the activities in such a way that the project reaches its objectives.

The project organisation is intended to constitute a Nordic expert network on requirements elicitation, specification, and assessment for digital I&C. The network will provide a forum for exchanging experiences and research results on the questions to be addressed by the project, and will provide a basis for evaluating the relative merits of the different practices, the relative importance of identified criteria, etc. A related concern is to facilitate knowledge transfer from other areas applying equipment that are used in NPPs.

The emphasis on best practices and identified success criteria means that the project will need to deal with real cases involving the development of a digital I&C system. By organising the project on basis of a Nordic expert network, it is expected that the project will contribute to the synthesis of knowledge and experiences, enhancement of competence on requirements elicitation, specification, and assessment, improved awareness of alternative practices, a basis for assessing current practices, and an incentive to search for best practice.

8.2 Project Activities and Further Plans

The project is carried out through a combination of project meetings, industrial seminars, and coordinated report writing. Each meeting focuses on a limited set of issues, where the participating organisations are asked to prepare a presentation on their experiences and viewpoints. Particular emphasis is given on concrete experiences from safety-critical applications. On this basis, the meetings attempt to provide a synthesis, evaluate the merits of the different practices, etc. The project activities in 2003 have included two regular project meetings and one industrial seminar. The minutes from the industrial seminar are given in appendix B.

The TACO project, which was initiated in 2002, delivered a preproject report in January 2003. The purpose of the preproject report was first of all to provide a technical basis and plan for further work. Particular emphasis was put on relating knowledge on relevant software engineering issues to NPP needs and practice. In this sense the report, with its identification of challenges and issues, provides an adequate basis and reference point for more detailed discussions and evaluations.

The activities in 2003 have constituted a natural continuation of the preproject, and have focused on the technical issues concretised in the preproject report. The work has concentrated on four central and related issues, viz.

- Representation of requirements origins
- Traceability techniques

- Configuration management and the traceability of requirements
- Identification and categorisation of system aspects and their models

The work was presented at the first TACO Industrial Seminar, which took place in Stockholm on the 12th of December 2003. The seminar was hosted by SKI.

On basis of this work and responses received at the Industrial Seminar, the TACO activity in 2004 will start by preparing a contribution to the planned Nordic Seminar on Automation, to be arranged in Oskarshamn, 5-7 April 2004. The further activity will aim at providing a unified exposition on the issues studied, and thereby facilitate a common approach to requirements handling, from their origins and through the different development phases. The approach will be presented at the second TACO Industrial Seminar, to take place in Helsinki in December 2004.

The responses received at the second Industrial Seminar will be utilized in the writing of the final report of the TACO project, to take place in the first half of 2005. The project is scheduled for completion on the 30th of June 2005.

The discussions from the meetings and the progress of the project are carefully reported by means of detailed minutes. Several of the main issues identified in the preproject are pursued in further research, and documented in terms of separate notes presented and discussed at the meetings. These notes are developed in parallel to the project activities, presented at the industrial seminars, and included in the overall documentation of the project.

The overall documentation schedule is as follows:

- January 2003: Preproject report (*completed*)
- December 12th, 2003: Presentations and materials to the first TACO Industrial Seminar (*completed*).
- January 5th, 2004: Documentation of the work for 2003 collected and sent in a suitable form to NKS (*the present report - completed*).
- April 5th, 2004: Presentations and materials to the Nordic Seminar on Automation.
- December 2004: Presentations and materials to the second TACO Industrial Seminar.
- January 5th, 2005: Documentation of the work for 2004 collected and sent in a suitable form to NKS.
- June 30th, 2005: Final TACO project report.

9 Appendix B: Minutes from the First TACO Industrial Seminar

Where*:* *SKI, Klarabergsviadukten 90, Stockholm*
When*:* *Friday 12th December 2003, 10:00 - 15:00*

Participants

Øivind Berg	Institute for energy technology	Norway
Rune Fredriksen	Institute for energy technology	Norway
Terje Sivertsen	Institute for energy technology	Norway

Atoosa P-J Thunem	Institute for energy technology	Norway
Bo Liwång	SKI	Sweden
Jan-Ove Andersson	Ringhals AB	Sweden
Hans Edvinsson	Ringhals AB	Sweden
Bjarne Grönqvist	Forsmarks Kraftgrupp AB	Sweden
Peter Reibe	Forsmarks Kraftgrupp AB	Sweden
Karl-Erik Eriksson	Oscarshamn Kraftgrupp AB	Sweden
Olli Ventä	VTT Industrial Systems	Finland
Björn Wahlström	VTT Industrial Systems	Finland
Lars-Erik Häll	TVO	Finland
Kai Salminen	Fortum	Finland
Martti Välisuo	Fortum	Finland

The intention with the First TACO Industrial Seminar was to present and discuss the work within the TACO project with a wider audience representing actors in the nuclear sector in the Nordic countries. In this way, the seminar would contribute to the dissemination of the research results to the intended end-users, and also to providing input to further work within the project, reflecting priorities set by end-users. Background and questions for discussion were presented in the agenda (TACO-015).

9.1.1 Welcome

After a word of welcome - and some practical information - by Bo Liwång, Terje Sivertsen extended a welcome to all the participants on behalf of the TACO project. For the theme of the seminar, Sivertsen referred to the full title of the project, "Traceability and communication of requirements in digital I&C systems development". A special thank was given to Bo Liwång for his positive and helpful response to the idea of arranging the first industrial seminar in the premises of SKI. A thank was then given to all the participants, whose participation was taken as a sign that the topic dealt with in the TACO project was an important one to the business of the different organisations. After having presented the agenda, the word was given to the first speaker.

9.1.2 Regulatory Aspects on Software Based Safety Systems

In his presentation, Bo Liwång focused on the need for development of a regular review strategy, including the development of a guidance document. Liwång referred in particular to the work being done by the Nuclear Regulators Working Group (NRWG) in EU. NRWG is planning a new report in 2005 where the issue of "safety demonstration" expectably will be given a stronger emphasis. Of the important, but often-neglected evidences in a safety demonstration, Liwång mentioned in particular competence and qualifications.

Liwång maintained that turnkey solutions based on a requirements specification does not work well in general for modernization projects. With reference to the approach followed in the CEMSIS project, Liwång stressed the importance of having a picture in forehand to see which activities need to be performed during a project. When it came to major classes of faults in the development of digital I&C systems, Liwång mentioned (1) incompleteness or faults introduced in the requirements specification, and (2) faults introduced in the maintenance phase. In the latter case, there is a need for adequate configuration management, requirements change management, etc. Liwång finally focused on traceability, structures, and strategy as three important facets of documentation.

Questions/discussion

On a question on whether SKI distinguish between COTS and pre-developed software, Liwång answered in the negative on the grounds that in both cases the main problem is lacking knowledge on how the software has been developed.

A relatively detailed discussion centred on the needs vs. possibilities of a more deterministic licensing process, with more objective criteria against which to do the judgement in a safety case. Liwång maintained that the judgement must be based on expert judgment, the adequacy of which could however be difficult to explain to the public. According to Liwång, it is not realistic to expect sufficient measures on the product and the process that would allow a purely deterministic approach. As an example on the need for some kind of expert judgement, Liwång mentioned reasoning centring on how well an organisation satisfies the criteria in the Capability Maturity Model (CMM).

9.1.3 TACO: Introduction

In his introduction to the TACO project, Terje Sivertsen gave an overview on the background and objectives of the project, some of the main issues, and the project activities 2003-5. Among the main issues, Sivertsen mentioned several aspects of requirements, requirements engineering, and management, with particular emphasis on the aspects of traceability and communication. Sivertsen maintained that these aspects were central to many software engineering activities and life cycle phases. Traceability is important in particular for demonstrating correctness of the implementation, but also for tracing back to the original requirements and information sources when needed. Communication plays an important role for example in demonstrating that the requirements correctly reflect the safety analysis of the plant and other relevant information. With regard to TACO activities, Sivertsen emphasised the role of the two industrial seminars as a way to present the TACO work and extend the network to a broader representation of Nordic nuclear power.

Questions/discussion

On a question on the role of the functional specification in the TACO work, Sivertsen replied that the focus in the TACO project is on the management of requirements throughout their lifecycle instead of isolating them to specific life cycle phases. The idea is to relate requirements to the different phases, not only to the requirements specification phase.

9.1.4 TACO: Requirements Traceability

Rune Fredriksen introduced some basic definitions related to requirements traceability, and presented various possibilities for how tools and techniques can be used to facilitate traceability documentation and management. Starting from the basic definition of requirements, as the ability to describe and follow the life of a requirement in both a forwards and backwards direction, Fredriksen described the four different types of traceability links. These links can be classified into pre-traceability and post-traceability links, which can be further refined by distinguishing between forward and backward direction tracing. From his overview of various traceability techniques, Fredriksen mentioned in particular the use of TopicMaps, within the document centred approach, as an interesting possibility. In general, TopicMaps enable multiple, concurrent views of sets of information objects. Fredriksen also discussed several ways in

which traceability techniques differ, four classes of tools that can be used for supporting requirements traceability, and finally some traceability metrics related to the products and processes involved at the software and systems level.

Questions/discussion

In a comment to the topic presented, it was underlined that tools for traceability must be applied from the beginning of a project. A traceability tool becomes very important in the quality assurance and needs to be treated as a configuration item in itself.

In another comment, the use of traceability tools was mentioned as useful also for safety demonstration purposes. To this comment it was added that also V&V need to be traced backwards and forwards, similar to the requirements themselves.

Another aspect that was discussed addressed the need for adequate knowledge on how to use the traceability tools, a viewpoint that was supported by findings in the CEMSIS project. The need to keep things simple was mentioned as one of the success criteria.

An important, practical problem that was addressed in the discussions concerned the treatment of changes and modifications of the requirements. The central issue in this respect is how much one needs to change and rework as a consequence of requirements change.

It was agreed that configuration management and requirements traceability management should be part of the same approach, in the sense that the latter should be part of the first.

9.1.5 TACO: Requirements Communication, Understanding, and Analysis

Olli Ventä started his presentation by giving an overview of the different software engineering activities involved in a development project, and continued with SWEBOK's breakdown of the software requirements area. On this basis, communication and understanding of requirements were refined into a number of aspects, with particular attention to the potentials of a common language to facilitate understanding across professions. UML was mentioned as a language that both the domain experts and software engineers can agree upon. Ventä promoted the professional use of diagram techniques, and presented examples of SADT, data flow diagrams, class diagrams, control flow diagrams, timing diagrams, state transition diagrams, activity diagrams, and breakdown of systems into subsystems. As an item to discuss, Ventä questioned the practical relevance of making a requirements specification that is really design and implementation free.

Questions/discussion

From a licensing viewpoint, the idea of relaxing the obligation that a requirements specification is design and implementation free was opposed as conflicting with the need for flexibility. It was maintained that such a relaxation was in conflict with the expressed viewpoint of the NRWG, which uses the formulation that the requirements should be independent. According to this view, the concerns about design and implementation can be included when designing the development process, but the requirements to the system should be independent.

To this viewpoint, Ventä pointed out that his point must be understood in the context of defence-in-depth, where the solution one is identifying is in practice not implementation independent. This is what happens in other industries. To this reply a comment was added that one always has the design in mind, and that it therefore may be a philosophical question whether the requirements can be fully independent from design and implementation.

From an industrial perspective, it was emphasised that independence is important in order to facilitate deliverables of design and implementation from different companies, and that more focus need to be put on the early process description phases.

9.1.6 TACO: Requirements Engineering and Management - *Integration between systems engineering and process engineering*

In her presentation, Atoosa P-J Thunem focused on issues related to the integration between systems engineering and process engineering, and on how this has been treated in TACO. In order to facilitate such an integration, Thunem proposed that the capabilities of a digital system should be seen in relation to its requirements, functions, system failures, intended behaviours and communications, unintended behaviours, and unintended communications. On this basis, she proposed that the non-functional system aspects should be seen in relation to a set of key functional system aspects, viz. goals, functions, behaviours, communications, capabilities, and faults/failures. The specification of the different aspects can be done by using a combination of function-centred, behaviour-centred, and communication-centred specification techniques, of which Thunem listed several candidates. One of the issues presented for further discussion was whether it is possible to find the freezing points for the requirements, and whether it is really important to find such points.

Questions/discussion

Two of the questions concerned the use of prototyping and its relationship to the system life cycle model. To these questions, Thunem replied that prototyping can use the same model in different development phases, and confirmed that this approach is closely related to the spiral model. In relation to the latter point, it was asked whether the requirements still need to be handled as frozen. To this question, it was commented that the freezing points are important in practice, especially with respect to contractual matters.

It was further commented that lack of requirements should be considered undesired functions. Lack of requirements tends to be easily ignored. To this comment, Thunem stated affirmatively that over-focus on intended functions is a common tendency in industry in general.

With regard to the presented list of specification techniques, it was commented that standards on how to use them are lacking, which makes it difficult to judge the results when they have been used. Thunem replied that this was indeed her point, i.e. there does not exist a universal model that provides the necessary guidance in focusing on the different specification aspects. A related comment addressed the fact that there are good standards for how to use functional I&C diagrams. To this, a comment was added that one may know how to use the different diagrams but still one has to know what to describe.

9.1.7 TACO: Requirements Engineering and Management - *Requirements and life-cycle models - theory vs. practise*

In the absence of Janne Valkonen (due to passing illness), Olli Ventä gave a presentation on requirements management with particular emphasis on the relevance of different life-cycle models. Taking the waterfall model as his starting point, Ventä focused on the problem of having to retrace previous development phases when changes to requirements become necessary in later phases. With reference to the EUP (Enterprise Unified Process, an extension and refinement of RUP, the Rational Unified Process), Ventä illustrated how the focus can be moved from time (as in the waterfall model) to phases, where several phases can be run in parallel. Ventä maintained that such a model can be put into a spiral approach, where the system is developed through different versions and with new layers of functionality. Ventä concluded his presentation by inviting the audience to come forward with their experiences with these issues and how they deal with them.

Questions/discussion

One of the concerns that was raised was how to make a good argument for the safety case when using a development approach based on prototyping and the spiral model. It was argued that the main differences between the waterfall model and the spiral model when it came to change control related to the amount of changes. Briefly, when one has many changes, one is in practice employing the spiral model. Conversely, the waterfall model generally assumes a small amount of changes. Accordingly, the idea is that the development should involve few changes so that it can be performed within the waterfall model, with backtracking on changes. To this viewpoint, it was commented that the waterfall model can in principle be employed within the spiral model.

It was further commented that it is also in the interest of the utility to have a licensing process that is as simple as possible. Part of the strategy has therefore been to reuse as much as possible. Accordingly, one should try to be optimal with respect to what can be designed at a given level in the development, and keep in mind that the concern for licensing is of interest also for the utility.

From a licensing viewpoint, it was stated that even if the waterfall model may be the most adequate for class A systems, one could consider greater flexibility when it comes to class B and C systems. On a question whether this was not a relaxation with respect to earlier licensing practice, it was replied that one will try to allow different development models. In order to facilitate this, there is a need to look into how the safety case can be done for other models. In this respect, IEC 61508 was considered too rigid when it came to acceptable models.

9.1.8 Requirements Engineering and Traceability within the Oskarshamn 1 MOD-project

In his presentation, Karl-Erik Eriksson presented some of the experiences from the Oskarshamn 1 MOD-project with respect to requirements, the design and review process, and configuration management. The project involved the development of systems in all the classes A, B, and C, and hence with certain differences in how they were handled. An important basis for discussing and handling the requirements in the project was established through the subproject SCR (Safety Concept Report).

The different requirements could be traced throughout the development by means of different tables. These tables also included requirements given in regulatory guides, and their relationship to related technical standards. Eriksson illustrated how the use of these tables facilitated traceability of requirements through to the factory acceptance tests. In order to manage requirements changes in later development stages, a design and review process was established that allowed controlled loops in the development involving revisions of the original requirements. This was important in order to maintain the ability to trace the requirements from the beginning. Eriksson explained how these modifications were implemented and documented at different steps during design and commissioning. Finally, a brief overview was given on the configuration management approach employed, where the consistent use of the first four letters of the alphabet, combined with numbers from zero and upwards, facilitated an easy identification of the different versions for the four phases integration, verification, validation, and commissioning.

Questions/discussion

On a question on how the project kept track of all the dependencies between the requirements, it was replied that this was indeed a difficult and effort-consuming task with a clear potential for improvement. It was emphasized that keeping track of these dependencies was a central issue in the development, but that it needs to be improved and simplified, preferably trough the support of some kind of database. A particular need is to see how other requirements might be affected by a requirements change. While changes in the main documents were relatively easy to handle, the task was more difficult when changes was made to subordinate documents.

9.1.9 Concluding Remarks

In his concluding remarks, Terje Sivertsen emphasized the usefulness of the seminar discussions for further work within the TACO project, and drew the attention to two important events in 2004, viz. the automation seminar (Oskarshamn, April), and the Second TACO Industrial Seminar (Helsinki, December).

Title	Traceability and Communication of Requirements in Digital I&C Systems Development. Project Report 2003.
Author(s)	Terje Sivertsen*, Rune Fredriksen*, Atoosa P-J Thunem*, Jan-Erik Holmberg**, Janne Valkonen** & Olli Ventä**
Affiliation(s)	*IFE, Norway **VTT, Finland
ISBN	87-7893-149-5 *(Elektronisk rapport)*
Date	March 2004
Project	NKS_R_2002_16
No. of pages	38
No. of tables	1
No. of illustrations	10
No. of references	24

Abstract

The overall objective of the TACO project is to improve the knowledge on principles and best practices related to the issues concretised in the pre-project. On basis of experiences in the Nordic countries, the project aims at identifying the best practices and most important criteria for ensuring effective communication in relation to requirements elicitation and analysis, understandability of requirements to all parties, and traceability of requirements through the different design phases. It is expected that the project will provide important input to the development of guidelines and establishment of recommended practices related to these activities.

In the year 2003, the TACO-project concentrated on four central issues:

- Representation of requirements origins
- Traceability techniques
- Configuration management and the traceability of requirements
- Identification and categorisation of system aspects and their models

The work was presented at the first TACO Industrial Seminar, which took place in Stockholm on the 12th of December 2003. The seminar was hosted by SKI.

Key words Traceability, requirements, Digital I&C, systems development