SKI Report 2006:27

Research

Safety Justification of Software Systems

Software Based Safety Systems
Regulatory Inspection Handbook

Swedish Nuclear Power Inspectorate

Dahll, Gustav, OECD Halden Project
Liwång, Bo, Swedish Nuclear Power Inspectorate
Wainwright, Norman, Wainwright Safety Advice

2006

ISSN 1104-1374
ISRN SKI-R-06/27-SE

SKI Perspective

SKI Report Serie “Safety Justification of Software Systems”

The introduction of software based systems in nuclear power facilities also needs introduction of new strategies for development and safety justification activities. As reports with different specific topics but related to the same overall issue, how to make a safety justification of the system, all issued reports related to the topic will have the same starting title “Safety Justification of Software systems”.

Background

The introduction of new software based technology in the safety systems in nuclear power plants makes it necessary to develop new strategies for regulatory review and assessment of these new systems that is more focused on reviewing the processes at the different phases in design phases during the system life cycle.

In the Swedish regulation there is a general requirement that the licensee shall perform different kinds of reviews. From a regulatory point of view it is more cost effective to assess that the design activities at the suppliers and the review activities at the licensee are performed with good quality.

But the change from more technical reviews over to the development process oriented approach also cause problems. When reviewing development and quality aspects the issues are more “soft” and it is more to review the structure and validity of arguments and evidences that the requirements are met, taking into account the development process over the whole lifecycle from concept phase until installation and operation. Even if we know what factors that is of interest we need some guidance on how to interpret and judge the information.

Purpose (of the project)

The purpose of this project was to collect information from a selection of the most important sources as guidelines, IAEA and EC reports etc, and map the information into a selected lifecycle model (from IEC 61508)

Results

For each of the different phases in the lifecycle model important issues are identified. Based on this a number of questions are defined together with a discussion on expected answers.

This research report shall be seen as a step to develop a structure in “measure” the quality level in a development process, and not as detailed regulatory requirements.

Effects on SKI work

The result will be used as background material in the development of a regulatory safety review strategy and in the regulatory review of safety systems based on software technology at the Swedish utilities

Project information

This report 2006:27 is replacing the planned publication of SKI Report 2004:18.

SKI reference: 14.8-000267
Responsible at SKI has been Bo Liwång

SKI Report 2006:27

Research

Safety Justification of Software Systems

Software Based Safety Systems
Regulatory Inspection Handbook

Swedish Nuclear Power Inspectorate

Dahll, Gustav, OECD Halden Project
Liwång, Bo, Swedish Nuclear Power Inspectorate
Wainwright, Norman, Wainwright Safety Advice

2006

Preface

The introduction of new software based technology in the safety systems in nuclear power plants also makes it necessary to develop new strategies for regulatory review and assessment of these new systems that is more focused on reviewing the processes at the different phases in design phases during the system life cycle.

It is a general requirement that the licensee shall perform different kinds of reviews. From a regulatory point of view it is more cost effective to assess that the design activities at the suppliers and the review activities within the development project are performed with good quality.

But the change from more technical reviews over to the development process oriented approach also cause problems. When reviewing development and quality aspects there are no "hard facts" that can be judged against some specified criteria, the issues are more "soft" and are more to build up structure of arguments and evidences that the requirements are met. The regulatory review strategy must therefore change to follow the development process over the whole lifecycle from concept phase until installation and operation. Even if we know what factors that is of interest we need some guidance on how to interpret and judge the information.

For that purpose SKI started research activities in this area at the end of the 1990's.

In the first phase, in co-operation with Gustav Dahll at the Halden project, a lifecycle model was selected. For the different phases a qualitative influence net was constructed of the type that is used in Bayesian Believe Network together with a discussion on different issues involved.

In the second phase of the research work, in co-operation with Norman Wainwright, a former NII inspector, information from a selection of the most important sources as guidelines, IAEA and EC reports etc, was mapped into the influence net structure (the total list on used sources are in the report). The result is presented in the form of questions (Q) and a discussion on expected answers (A).

This research report shall be seen as a step to develop a structure in "measure" the quality level in a development process, and not as detailed regulatory requirements.

Finally I would like to thank the British Health and Safety Executive and the Controller of Her Majesty's Stationery Office for their kind permission to use their internal material.

Bo Liwång
Swedish Nuclear Power Inspectorate

Contents

1 Introduction.

The Swedish Nuclear Power Inspectorate Regulatory Code, SKIFS 1998:1, entitled "The Swedish Nuclear Power Inspectorate's Regulations Concerning Safety in Certain Nuclear Facilities", August 11, 1998[1] requires that nuclear power reactors (amongst other nuclear facilities for which permission is required from the Government for nuclear activities to be undertaken) should prevent nuclear accidents "through a basic facility-specific design which shall incorporate multiple barriers as well as a facility-specific defence-in-depth system". This defence-in-depth is to be achieved in part though multiple devices to protect the integrity of barriers which physically contain any radioactive substances and, if the integrity of these barriers is breached, then these devices should mitigate the consequences that ensue. The safety systems (see Chapter 2, "Terminology" for the definition of some of the terms used in this guide) of nuclear power plants are just such devices since they act in response to a fault to prevent or mitigate a radiological consequence. The objective of these guidelines is to assist SKI in their evaluation and approval of the Safety Report and Safety Reviews (see SKIFS 1998:1 Chapter 4) for a nuclear power plant which includes a software-based safety system. These safety systems should not only be fit-for-purpose but they must be demonstrably so. The Safety Report and the Safety Reviews should contain an adequate safety demonstration for any software-based safety systems associated with a nuclear power plant.

It should be noted that where reference is made in the guide to a "safety system" this should be interpreted as meaning a "software-based safety system" unless it is specifically stated not to be so. Also, as will be seen from the terminology in Chapter 2, a *safety system* consists of a *protection system*, *safety actuation systems* and *safety system support features*. All these individual entities may contain computer systems which incorporate software. The guidance provided applies to all this software, and an adequate safety demonstration should be provided for each and every instance, including software incorporated into *smart sensors*, *intelligent actuators and computer-based electrical switch gear etc*. It is, however, acknowledged that the majority of the software will be incorporated into the *protection system,* and this is where the main assessment effort will, in practice, be focussed (often the term *safety system* could be replaced by the term *protection system*).

These guidelines should be used during the whole development process, from the first plans by a power company to install a software-based safety system until the approval of the use of the final system. The focal point of these guidelines is the safe use of the system, and the intended safety shall be preserved during all steps of the system development.

These guidelines are based on the following standards and guidelines, plus other references mentioned in the text:

1 IEC 61508 'Functional safety of electrical/electronic/ programmable electronic safety-related systems'.

2 IEC 60880 'Software for computers in the safety systems of nuclear power stations' 1986.

3 IAEA Safety Standards Series, Safety Guide NS-G-1.1, "Software for computer-based systems important to safety in nuclear power plants", September 2000.

4 IAEA 'Software Important to Safety in Nuclear Power Plants', Technical Report Series No. 367, 1994.

5 Four party regulatory consensus report on the safety case for computer based systems in NPPs.

6 EWICS TC7 guidelines.

7 SKI-report 94:9.

[1] *The views expressed in this document are based upon the English version of these Regulations. It should be noted that in all cases concerning interpretation the Swedish version takes precedence.*

8 "Common position of European nuclear regulators for the licensing of safety critical software for nuclear reactors", European Commission's Advisory Experts Group, The Nuclear Regulators' Working Group Task Force on Safety Critical Software, EUR 19265 EN, version 11, May 2000;

9 IEC 61513, "Nuclear power plant - Instrumentation and control for systems important to safety - General requirements for systems", March 2001;

10 IAEA Safety Standards Series, NS-R-1, "Safety of Nuclear Power Plants. Design Requirements", September 2000 which replaces IAEA Safety Standard, 50-C-D (Rev 1) 1988,;

11 IAEA Safety Guide, 50-SG-D3, 1980, "Protection systems and related features in Nuclear Power Plants".

12 UK Nuclear Installations Inspectorate, Assessment Guide AG3, "Protection Systems - software aspects of digital computer-based protection systems", Issue 1, February 1991, Draft for trial use. (Currently under revision)

13 "Safety assessment principles for nuclear plants", UK Health and Safety Executive, 1992, ISBN 0 11882043 5.

14 "The use of computers in safety-critical applications", UK Health and Safety Commission, 1998, ISBN 0 7176 1620 7.

However, it should be particularly noted that extensive use, often in the form of verbatim extracts with some modification to fit the needs of this document, has been made of the European Commission's document (reference 8) and the IAEA software guidelines (reference 3). This approach has been considered necessary in order to ensure Sweden complies with this international guidance. Extracts from references 12 and 13 have also been used, again sometimes with modification. These documents are subject to UK crown copyright. Crown copyright material is reproduced with the permission of the Controller of Her Majesty's Stationery Office.

2 Terminology

The following terms and definitions are used in this report.

- *Safety system*: a system which acts in response to a fault to prevent or mitigate a radiological consequence.
- *Protection system*: the instrumentation within a safety system which measures (or monitors) plant parameters (or states), and the equipment which generates safety actuation signals when these parameters (or states) move beyond pre-set limits.
- *Safety actuation system*: the equipment within a safety system which physically accomplishes the required safety action(s) in response to actuation signals from the protection system.
- *Safety system support features*: the equipment within a safety system which provides services such as energy, cooling and lubrication required by the protection system and the safety actuation systems.
- *Software-based safety system*: a safety system which incorporates a computer system or systems containing software. The software may be contained in any part of the safety system, i.e. the protection system, the safety actuation system or the safety system support features. This guide relates to all these items of software.
- *Plant* The plant, or the part of the plant, which interacts with the safety system, and which can cause a hazard.)
- *(Power) Company* The company operating the power plant, and which will develop a safety related system which needs approval from the licensing authorities (SKI).
- *Milestone* A stage in the development where the company contact SKI for discussions about whether the development follows the right track, and safety is preserved, so that the development can continue.
- *(Milestone) goal* An expression about what needs to be achieved as a goal at each milestone.
- *Influence net* A graphic depiction of aspects that influences the decision at the milestone goal, and connects these again to observable nodes
- *Observable node* Node which identify sources of information which can be used to make conclusions about the influencing aspects.

3 Structure of the guideline.

The structure of the guidelines is such that it contains a set of milestones. At each milestone there should be a check that the intended safety goal is preserved. Besides ensuring that the requirements of SKIFS 1998:1 are met, SKI will follow the system development through all milestones and check that the safety measures are fulfilled before the development continues.

One intention with the guideline is that it should be easy to understand and follow during the development process. We will therefore try to make graphical models which show which observations, and corresponding documentation, influence the decision at each milestone.

The guidelines will be built up around a lifecycle model. There exists various, although similar, lifecycle models. A natural choice would be to use one model from one of the guidelines mentioned above, perhaps modified with input from the others. The lifecycle model used as basis here is the one in IEC-61508. The phases in this lifecycle model form a basis for the selection of milestones in the assessment guideline. However, the life-cycle model for the development of the software-based safety system itself should be based on that shown in figure 6 on page 62.

As stated above, at each milestone there should be a check that the intended safety goal is preserved. These checks should be made on the basis of some criteria. These criteria could be rule based, i.e. certain strict rules should be fulfilled before a milestone is approved, and the development can continue. This may, however, be very rigid, and such rules may also be an obstacle to utilisation of technological development. A better alternative is that these checks should be made on the basis of a combination of evidences from different information sources, available about the system at the milestone.

The guidelines contain one chapter for each milestone. Each chapter contains a short introduction, an influence net (see definition above) and a questionnaire. The influence nets are intended to give a better understanding of the relation between observations, subjective judgements, and the milestone goals (see definition above).

To facilitate a practical use in the regulatory review, a corresponding questionnaire, which reflects the structures of the corresponding graphs, is made. The questions relate to the observable nodes identified in the influence nets. To simplify the questionnaires, the questions relating to a higher level node in the hierarchy of the influence diagram are grouped together. Guidance is provided with each question to assist the assessor in evaluating the licensee's responses.

4 Lifecycle milestones.

A proposal for a set of milestones and corresponding milestone goals is given in table 1. Notice that this list contains few references to software. The intention is that the milestones should be connected to the system development as a whole, and that particular software aspects should be treated as details in this development. However, these guidelines will make particular emphasis on software aspects.

A short summary of these milestones is given below, whereas more detailed explanations are made in the following sections.

Table 1 List of milestones

Milestone	**Milestone goal**
1. Concept	Preparedness
2. Overall requirements	Completeness
3. Hazard and risk Analysis	Safety of concept
4. Requirement specifications	Dependable specifications
5. Project planning	Quality of plans
6. Overall System Design	Dependable design methodology
7. System realisation • Detailed design • Programming • System integration	Dependable system
8. System validation	Validated system
9. Maintenance and modification procedures	Safety preserved during operation.

The *Concept*s milestone occurs when the power company seriously considers installing a software-based safety system, or replacing an already existing system with a software-based one. At this stage there is no detailed specification of the safety system. The milestone goal *Preparedness* expresses the confidence at this stage that the power company have properly considered the safety aspects of introducing a software-based safety system, and also that that they are experienced enough to be able to handle the safety aspect throughout the system development.

At the *Overall requirements* milestone there should be a set of functional requirements which describe what the system is intended to <u>do.</u> The milestone goal *completeness* is a measure of how complete, unambiguous and comprehensible these requirements are. This milestone is met in Sweden through the regulations incorporated into SKI 1998:1. Before a new NPP can be constructed, a preliminary Safety Report must be prepared which will provide a set of functional requirements. The content of the Safety Report is stipulated in Appendix 2 of SKIFS 1998:1. Information is required on siting of the NPP, the design basis, facilities & functions, radioactive substances, radiation protection, operation of the facility, analysis of operational conditions (safety analysis), references and drawings. It, basically, covers the high-level requirements, and how the different systems are designed to meet the requirements. This has to be subjected to two Safety Reviews (SKIFS 1998:1 Chapter 4 paragraph 3) before construction can commence. One review ensures that the applicable safety-related aspects of a specific issue have been taken into account and that appropriate safety-related requirements with respect to the design, function, organisation and activities of the NPP are met. - the use of a software-based safety system on an NPP is just such a specific issue, and should be part of the Safety Review. A second Safety Review is then performed by an independent group within the parts of the NPP's organisation which are responsible for the specific issues as well as within a safety review function appointed for this purpose. This group must be independent from the parts of the organisation which are responsible for the specific issues covered in the first review. Both these reviews must be documented. The Safety Report and the two Safety Reviews constitute a preliminary safety demonstration for the NPP. The first Safety Review should be regarded as formulating the basic safety demonstration for the software-based safety system, and the second, independent, Safety Review is an *independent assessment* of the safety demonstration as required by Chapter 1.9, "Independent Assessment" of reference 8. The independent assessors should satisfy themselves that the claims made in the Safety Review are correct and that the documentary evidence supports those claims. Where appropriate, they should employ diverse means, such as different software tools, to confirm the findings. This independent Safety Review should detail any differences of opinion between the two review groups, and should detail their resolution.

It is recommended that SKI assessors sample the actual project documentation, and consult the project personnel, to satisfy themselves that the reporting in the *Safety Reviews* is accurate and sufficiently comprehensive.

For a new NPP, the Preliminary Safety Report and its reviews should describe the basic commitment to the use of a software-based safety system, and should provide a programme of activities that demonstrate how the safety system will be demonstrated to be fit-for-purpose. The guidance in this report should then be used by SKI to determine whether or not these plans will deliver an adequate safety demonstration for the safety system. Construction should not be allowed to commence until a satisfactory programme has been agreed. One particular aspect of this phase is to agree a set of regulatory inspection points, some of which will be linked to the production of the safety documentation, i.e. the appropriate parts of the Safety Report and the Safety Reviews which describe the safety system and address the associated issues together with their independent reviews.
.
Prior to operation, a Final Safety Report is required to be produced, and reviewed, in the same way as for the Preliminary Safety Report. This set of documentation will contain the safety demonstration for the installed software-based safety system and should show why the system is fit-for-purpose. This final documentation plus the regulatory inspections that have been performed throughout the project must satisfy the questions posed in these guidelines.

Where an existing analogue safety system is to be replaced by a software-based safety system, or where a modification is to be made to an installed software-based safety system then SKIFS 1998:1, paragraph 6, "Modifications" requires that SKI be notified before the modification may be introduced. The Inspectorate may then decide that additional or other requirements or conditions shall apply with respect to the modification. This paragraph also states that "engineering or organisational modifications to a facility which can affect the conditions specified in the safety report as well as essential modifications to the report shall be reviewed in accordance with 3 para". It follows, therefore, that in both the above cases involving modifications, approval should be sought early so that the strategy for producing an adequate safety demonstration can be agreed with SKI.

In the case of a digital replacement for an analogue safety system, there will obviously be a significant change to the Safety Report due to the change in technology alone. Additionally, there may be changes to the system's functionality. Safety Reviews covering this particular safety issue will also need to be performed so that they, together with the Safety Report, will form a documented safety demonstration. The adequacy of this safety demonstration should then be established by subjecting it to a searching review through the questions in this guide. Where an existing software-based safety system is being modified, it has to be assumed that the licensee has the necessary experience in software to make such changes. Nonetheless, the modification process should be subjected to the questioning indicated in Chapter 13 of this guide.

It is suggested that the most appropriate approach to modifications is to treat any replacement of an analogue system, or any major change to a digital system, as a "new facility" that requires approval by SKI before "construction" can commence. This would then mean that a preliminary proposal for the modification of the Safety Report and preliminary Reviews would need to be produced. These would then be assessed by SKI against these guidelines, using chapters 5, 6, 7, 9 & 13. Following approval and installation, a final Safety Report and final Reviews would be produced and again assessed against all the guidance in this document.

Returning to the milestones, the two first milestones (*Concept* and *Overall requirements*) may often, in practice, occur in parallel and not in sequence. This means that the assessment does not start before the functional specification is produced, so that both *Overall requirements* and the *Concept* are assessed simultaneously.

Based on the documentation from the two previous milestones, a *Hazard and risk Analysis* should be performed. The objective is to determine the hazards and hazardous events the system may have on the plant for all reasonably foreseeable circumstances including fault conditions and misuse. The milestone goal *Safety of concept* is a measure of the confidence that the specified system can be developed into a system which fulfils safety requirements, provided that necessary safety requirements are added and implemented into the system.

The next step is to make complete *Requirement specifications,* including both the functional- and the safety requirements. The milestone goal *dependable specification* is a measure of the degree to which the results from the safety analysis are incorporated, and conform with the general safety and reliability requirements.

The assessment objective *of the Project planning* milestone is to show that the company can demonstrate that it has development plans to fulfil the requirements identified at the previous milestones. However, the project planning is difficult to place as a fixed step in the development process, and may be seen as a parallel activity to the actual development, and may therefore be assessed at various stages in this process.

At the *Overall system design* milestone the company shall document how the system shall be made to fulfil the requirements. Aspects to be evaluated in the assessment are design methodology, system architecture, the use of commercially available equipment (hardware and software), available design supporting tools, etc. The milestone goal *Dependable design* is a measure of the confidence that the design fulfils the dependability requirements stated in previous milestones.

The *System realisation* milestone is when the detailed design, programming and integration of the system are completed. Traditionally, it is often at this stage that the safety assessment, and contact with regulating body, have started. The objective is to assess whether the requirements are followed by the system, and to propose testing and other verification activities which are required to fulfil the safety requirements.

At the *System validation* milestone all the tests and other verification activities proposed have been completed, and the results evaluated. The milestone goal *validated system* is an expression of the degree to which the dependability targets required for the system is actually reached.

However, to ensure that this is a safe system during the operation life time, a *Maintenance and modification procedure* is also required. This should also be approved before start up of the system.

An influence diagram is then constructed at each milestone which identifies the relevant evidences and shows how they influence the milestone goal.

5 Concept

The first milestone is chosen to be the first node in the IEC 61508 guideline, viz. Concept. The objective at this milestone is, according to IEC 61508, that there is developed a level of understanding of the EUC and its environment (physical, legislative etc) sufficient to enable the other safety lifecycle activities to be satisfactorily carried out. (IEC 61508 uses the term EUC (Equipment Under Control), which means the plant, or the part of the plant which interact with the safety system, and which can cause a hazard.)

In practical terms, this milestone occurs when the power company seriously consider to install a (computer based) safety related (safety critical) system, or replace an already existing system with a computer based one. Notice that at this stage there are no detailed specification of the target system, only a more general description of what it is intended to do. The intention is that SKI and the power company shall interact at as early a stage as possible, to get on the right track from the beginning (this should be covered by the SKIFS 1998:1 preliminary Safety Report and Safety Reviews or by the modification process for the replacement of an exisiting system - as discussed earlier in Chapter 4 of these guidelines).

The influence net for this milestone is shown on Figure 1. The milestone goal (which is here, as in all figures, marked with a shadow) for this milestone, Preparedness, expresses the confidence at this stage that the power company have properly considered the safety aspects of introducing a computer based safety system, and also that that they are experienced enough to be able to handle the safety aspect throughout the system development.

Topic 1: Safety Culture

IEC 61508 identifies two relevant aspects of safety culture (, viz. safety engineering knowledge appropriate to the technology, and knowledge of the legal and safety regulatory framework. This topic is illustrated in Figure 1a.

<u>Questions about safety engineering knowledge</u>:

Comments:

These questions can be answered on the basis of documentation about previous experience on handling safety critical systems. The answers to these questions give an indication of the degree to which the Company has an experience with risk and hazard analysis. Both the amount of previous experience and the relevance of this experience is of importance.

A general knowledge about possible hazards at the plant and environment should also be present, in order to be prepared for a more detailed hazard analysis.

Q5.1 How much previous experience does the Company have in safety engineering?

A5.1.1 SKFS 1998:1 requires in Chapter 2. Para 3 that the licensee (the Power Company) shall have guidelines on how safety shall be maintained, and it (the Power Company) shall ensure that the appropriate personnel are acquainted with these guidelines. These regulations ensure by statute that the licensee has adequate experience in safety engineering. This aspect of safety culture should be verified by others in SKI who have the relevant expertise in overall plant safety.

Q5.2 Is their experience based on similarity in the function of the planned system?

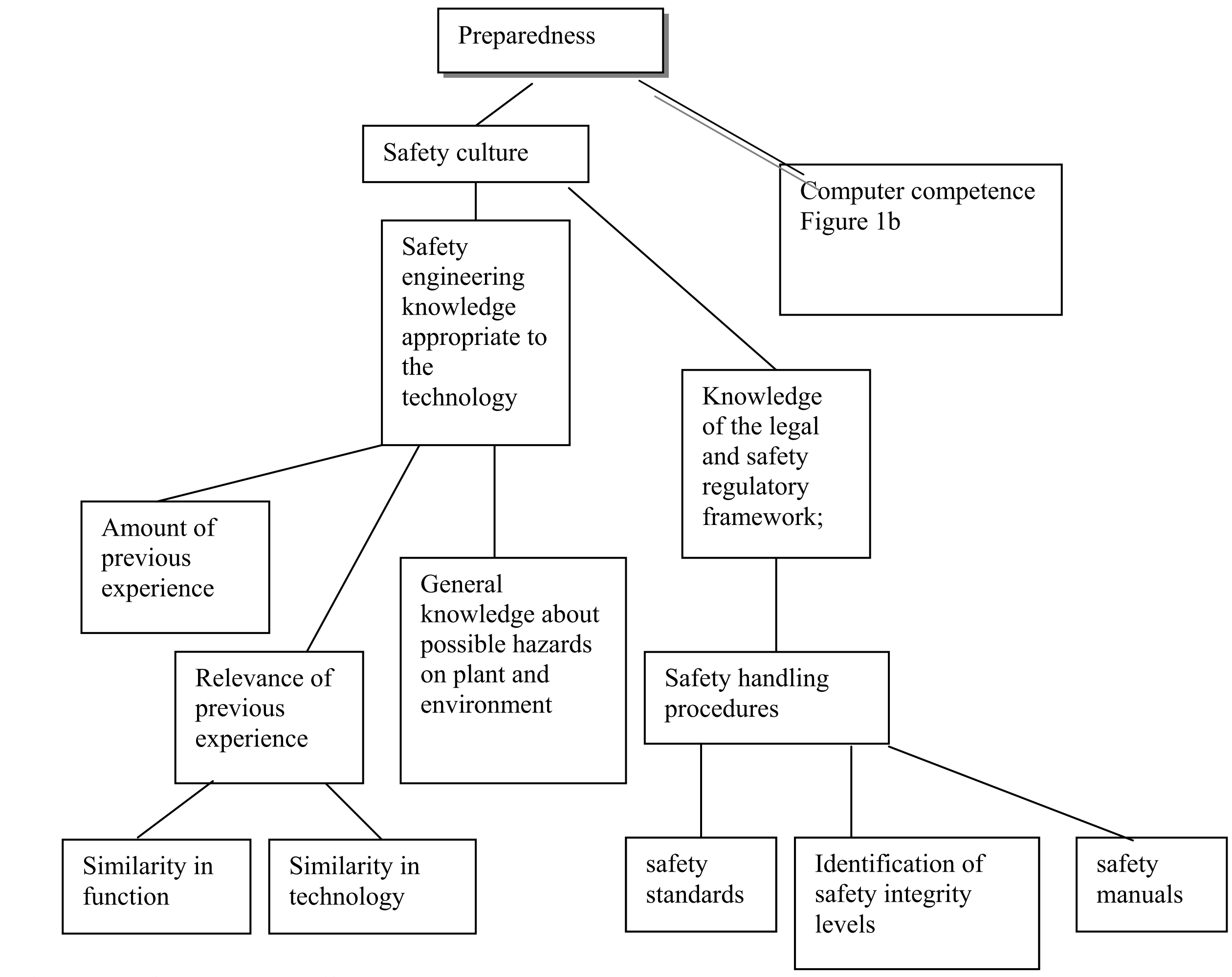
A5.2.1 Whilst the licensee must have adequate experience in safety engineering, it may not have the required in-depth knowledge of software-based, safety systems (the planned system) to perform all the tasks itself. These services will, most probably, be procured from outside the licensee's organisation. To ensure an adequate safety culture, it is essential that the licensee is capable of fulfilling the role of intelligent customer, i.e. the licensee should have at least a small number of staff with professional qualifications which meet the requirements of the equivalent of the "Safety, competency and commitment: Competency guidelines for safety-related system practitioners", published by the IEE, 1999, ISBN 0 85296 787 X, a document authored by UK's Institution of Electrical Engineers, the British Computer Society and the Health and Safety Executive. Typically, the staff should be familiar with, and, in particular instances competent in, the application of the relevant standards, procedures and methods associated with the design, development, installation and operation of the planned system such that they are able to understand and discuss these matters on equal terms with the supplier's specialists. These staff will act as the intelligent customers and should ensure that the software-based safety system is fit for purpose.

Q5.3 Is their experience based on similarity in the technology of the planned system?

A5.3.1 As with the previous question the licensee needs to have the capability to act as an intelligent customer.

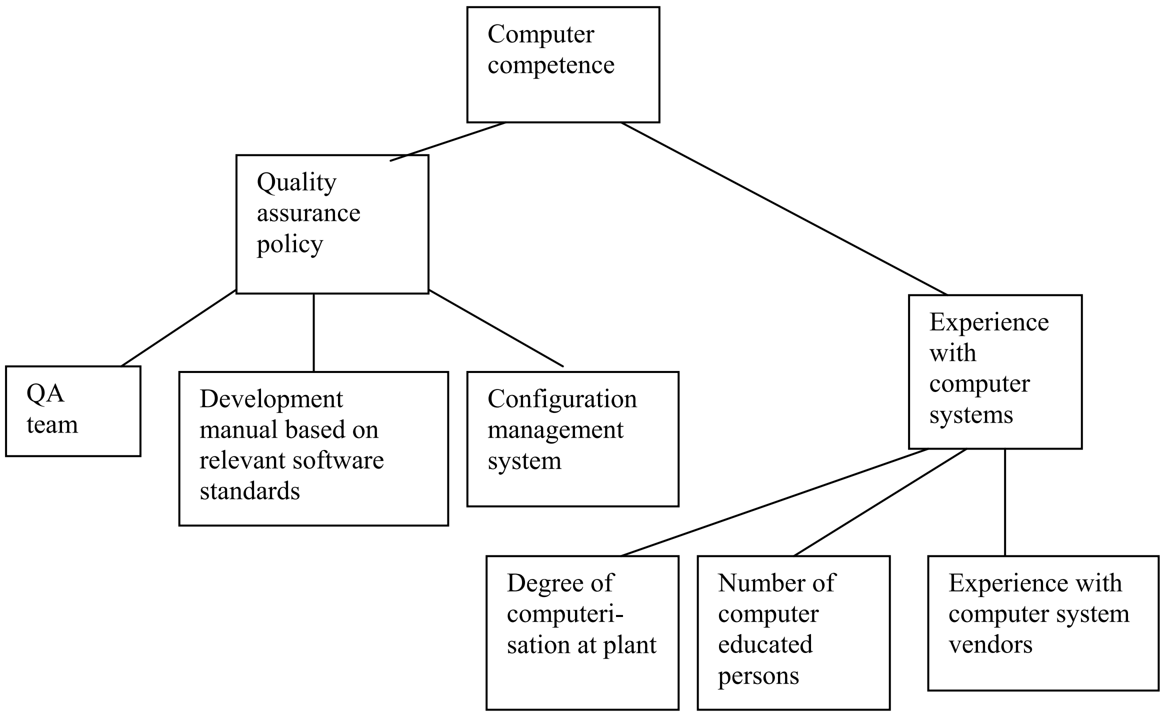
Q5.4 Does the Company have adequate knowledge about possible hazards on plant and environment?

A5.4.1 SKFS 1998:1 requires in Chapter 4. Para 1 that the licensee (the Power Company) shall perform a Safety Analysis "based on a systematic inventory of such events, event sequences and conditions which can lead to a radiological accident". This Safety Analysis shall be reported in the Safety Report. The appropriate SKI specialists should check this aspect.



Preparedness
Safety culture
Computer competence Figure 1b
Safety engineering knowledge appropriate to the technology
Knowledge of the legal and safety regulatory framework;
Amount of previous experience
General knowledge about possible hazards on plant and environment
Relevance of previous experience
Safety handling procedures
Similarity in function
Similarity in technology
safety standards
Identification of safety integrity levels
safety manuals

Figure 1a Influence net for milestone *Concept*



Computer competence
Quality assurance policy
Experience with computer systems
QA team
Development manual based on relevant software standards
Configuration management system
Degree of computeri-sation at plant
Number of computer educated persons
Experience with computer system vendors

Figure 1 b Influence net for milestone *Concept (cont.)*

<u>Questions about legal and safety regulatory framework</u>:

Comments:

The questions relate to the Company's awareness of legal and regulatory aspects. This can be demonstrated through documentation of previous experience, or through the set of procedures on how to deal with safety critical systems.

The Company should have a safety standard with a written safety manuals to follow. This should preferably also contain a classification scheme with safety integrity levels for safety related systems. IEC-61508 has a scheme with 4 levels. Other classification schemes are the American standard 1E, 2E etc. and the A,B etc. classification of IEC 1226.

Q5.5 Has the Company demonstrated good knowledge about safety standards?

A5.5.1 SKIFS 1998:1 Chapter 2.Para 3 requires that a licensee shall have documented guidelines on how safety will be maintained. These guidelines should meet the requirements of the Act (1984:3) on Nuclear Activities, namely that nuclear accidents shall be prevented through a basic facility-specific design (see SKIFS 1998:1, Chapter 1,para 1). A licensee will be in breach of its licence if it cannot demonstrate that it has staff who have a good knowledge of safety standards. Chapter 2.Para 3.Sub-section 4 requires that staff competence be documented - it is recommended that staff competence is checked by the SKI assessor.

Q5.6 Has the Company written safety manuals to follow?

A5.6.1 Again this is covered by SKIFS 1998:1 Chapter 2.Para 3.Sub-section 1, i.e. that the licensee shall have "established documented guidelines". Licensed sites must meet this requirement.

Q5.7 Has the Company a classification scheme with safety integrity levels which the planned system shall be placed in?

A5.7.1 SKIFS 1998:1 Chapter 3.Para 4 requires that licensees shall design, manufacture, install, controll and test building components, devices, components and systems in accordance with "requirements which are adapted to their importance for safety". This regulation means that a licensee must have a means of classifying systems into which the planned system can be placed. Appendix 2, "Design Basis" requires a description of this classification system. The fact that this document is dealing with safety systems means that the IAEA classification system for safety systems is assumed, and the question is to some extent irrelevant in this context. However, the power company should recognise that it has to classify such systems accordingly. With regard to the safety integrity level, this aspect will more naturally flow from the probabilistic safety analysis (see "Comments on Certain Paragraphs, On Chapter 4.1para", pp 32 & 33) since the safety system's integrity level will be determined by the risk reduction required to meet the safety goals of the level 1 or level 2 PSAs.

Topic 2: Computer competence

Another aspect of relevance at this milestone is the company's competence in the application of computer based system for plant control. (See Figure 1b) Even if the company is not developing the system itself, it should still have the competence to deal with relevant questions and make correct decisions, i.e. to act as an “intelligent customer”. This topic is illustrated in Figure 1 b .

Questions about Quality Assurance policy:

Comments:
The questions relate to the Company's quality assurance policy, in particular on software quality assurance. An evaluation of this policy must be based on whether the Company has a quality assurance team, how this is organised, and whether it has a software development manual based on generally accepted software standards. It should also be documented that this policy is really implemented at all levels. A particular aspect is whether the Company follows a configuration management system. These questions apply equally to a safety system supplier.

Q5.8 Does the Company follow a well-defined Quality Assurance policy with an associated development manual?

A5.8.1 SKIFS 1998:1 Chapter 2.Para 3.Point 2 requires that a quality system be in place for activities that are important to safety. This is further clarified in "Comments on Certain Paragraphs, On Chapter 2.3 para", Point 2 which indicates that the IAEA's codes and guides for the quality assurance of the safety of a nuclear power plant can provide guidance for the design of the quality assurance system. Chapter 2.Para 3.Point 2 requires that activities important for safety are developed with the support of a quality system. Chapter 2.Para 4 requires that a quality manual or similar document shall be produced. The response to this question is therefore that the regulations require that a suitable Quality Assurance system be in place on a licensed site, and that it should be judged against the relevant IAEA codes and guides. The Safety Review should demonstrate how compliance with these IAEA codes and guides has been achieved.

Q5.9 Is this policy based on relevant commonly accepted software standards?

A5.9.1 Whilst SKIFS 1998:1 indicates that the quality system should follow IAEA codes, inspection of these IAEA codes and guides will show that, at present, they do not cover software quality assurance. However, some guidance is given in the Technical Report series No. 282, "Manual on quality assurance for computer software related to the safety of nuclear power plant", Vienna 1988. In addition, the IAEA software Safety Guide (reference 3, section 4.11) and the document giving the European nuclear regulators' common position (reference 8, section 1.6) provide some basic guidance.

A5.9.2 In judging whether the QA policy is based on the relevant commonly accepted software Quality Assurance standards, the Safety Review should contain a compliance statement from the licensee against either:

i) IEEE Std 730.1-1989, "Standard for Software Quality Assurance Plans";

ii) ISO 9001

and

ISO 9000-3 "Guidlines for the application of ISO 9001 to the development, supply and maintenance of software".

A5.9.3 For safety systems the quality system should cover all aspects of the system life cycle. It is important that the content of the QA Plans be agreed with SKI at the beginning of the project.

Q5.10 Has the Company a qualified Quality Assurance team?

A5.10.1 SKIFS 1998:1 Chapter 2.Para 4 requires that an independent Quality Assurance function shall periodically investigate the application, suitability and effectiveness of the quality system. The regulations, therefore, require that a licensee has a Quality Assurance team. The "Comments on Certain Paragraphs" amplify a requirement for this team by stipulating that it should have direct organisational links to the highest manager of the facility. It also stipulates that the quality audits should be conducted by "people who have a good knowledge of the activity which is being audited". Hence the regulations require that the QA team be suitably qualified. With regard to the software aspects, the Safety Review should demonstrate that the licensee's QA team contains a member who has a software QA competence at least to the level of a TickIT Auditor. A TickIT Auditor will be on the International Register of Certificated Auditors (IRCA). The licensee's QA team should either be accredited to the Swedish Board for Accreditation and Conformity Assessment (SWEDAC) or a suitable organisation with this accreditation should have reviewed both the licensee's and the software-based safety system's supplier's QA arrangements.

Q5.11 Does the Company follow a configuration management system?

A5.11.1 The QA requirements associated with an adequate quality system (as required by SKIFS 1998:1) mandates that an adequate configuration managent system (CMS) be in place throughout the software-based safety system's lifecycle. An adequate CMS will ensure for example that that faults are not introduced due to incorrect versions of software items being used and should form an integral part of an adequate software change process. Only the appropriate persons should have access to a software item until it has been approved and placed under configuration control. Version control and the issuing of approved software items should be formalised. This approach applies both to the supplier's and licensee's premises. Written procedures should be available which for instance comply with one of the following standards:

i) IEEE 1042-1987, "Guide to Software Configuration Management";

ii) IEEE Std 828-1990, "Software Configuration Management Plans".

A5.11.2 The Safety Review should provide evidence that the configuration manageement system is adequate. The use of the procedures should be demonstrated by means of a suitable audit which should be reported in the Safety Review.

<u>Questions about Experience with computer systems</u>:

Comments:
The questions relate to the Company's experience with the use of computers for control and supervision of the plant. Of relevance here is the degree of computerisation at the plant as well as the number of computer educated persons at the Company. Since the system most probably will be purchased as a standard system, experience with computer system vendors is also of importance.

Q5.12 What is the degree of computerisation at the plant?

A5.12.1 If there are a number of large software-based I&C systems already on the plant, particularly controlling and monitoring the reactor and turbine, then the degree of computerisation can be considered as extensive. Where the computerisation is limited to non-plant systems, the competence of the licensee to act as an intelligent customer should be reviewed carefully (see Q5.2 above).

Q5.13 What is the level of computer education among the persons who will work on this system?

A5.13.1 The level of computer education required by the persons who will work on the system depends upon their role in the project. The qualifications, experience and qualities need to be appropriate to the person's duties. Where the licensee is purchasing a software-based safety system rather than manufacturing it himself, the licensee must again be able to act as an intelligent customer (it is unusual for a licensee to have all the necessary qualifications, experience etc to produce a software-based, safety system himself: hence, where this is the chosen course of action, the qualifications, experience and attitudes of the personnel involved should be scrutinised most carefully). To fulfil that role, the staff dealing with the technical aspects of the system's supply will require to be educated to the equivalent of a university degree in a technical discipline such as electrical engineering and have sufficient experience such that they would be eligible to join their relevant professional body. They will also need experience in the nuclear power industry and the associated safety engineering. They should also be aware of their responsibilities in regard to the safey of the facility and demonstrate an appropriate attitude to these responsibilities. The Safety Review should discuss the competence of the persons who will work on the system.

A5.13.2 The vendor's personnel (or where the licensee is producing the safety system himself) will need similar qualifications, experience and attitude. However, they will require a greater in-depth knowledge and experience of the particular computer technology that they are applying. For example, they should be aware of, and how to comply with, the relevant software standards such as IEC 60880 and IEC 60880-part 2 plus the other standards listed in the reference section of this guide.. The system designers and programmers should be fully familiar with the computer hardware and programming languages being used; they should be able to demonstrate previous experience in the use of the technology. The Safety Review should discuss the competence of the vendor's personnel who will work on the system

Q5.14 How much of previous experience does the Company have with potential vendors?

A5.14.1 This relates to the role of intelligent customer. Previous experience with potential vendors is not necessary provided the licensee can demonstrate that he is an intelligent customer. However, the licensee's QA department will need to audit the potential vendors to estabilsh that they meet the QA requirements (see Q5.8 and Q5.9) required of a supplier of software-based safety systems.. A summary report of the audits should be provided in the Safety Review.

6 Overall Requirements

This milestone corresponds to the second node in the IEC-61508 lifecycle: 'Overall Scope Definition'. The objective at the milestone is to evaluate the company's effort to make system specific preparations for the hazard and risk analysis and the detailed requirement specification.

As mentioned in Chapter 4, in the case of a new nuclear power plant, the regulations in SKIFS 1998:1 require that a Safety Report is produced prior to the commencement of construction, and later, operation. This Safety Report contains the 'Overall Scope Definition' as can be seen from the description given in Chapter 4 of these guidelines under *Overall requirements*. The Safety Report required prior to construction (the Preliminary Safety Report) will contain the information required at this milestone, and which has to be evaluated and approved by SKI prior to the commencement of construction. This Preliminary Safety Report should also be supported by *Preliminary Safety Reviews*.

For an existing NPP, SKIFS 1998:1 requires that any engineered or organisational modifications, which could affect the conditions specified in the Safety Report, be subjected to the same review process as when it was originally produced. This means that a software-based safety system, which is replacing an existing analogue system, should be covered by a revised Safety Report and be subjected to a full review process. The *Overall requirements* will, therefore, be covered by the revised Safety Report. Since SKI must be informed of any such proposals, then responses to the questions in this chapter can be obtained prior to any further development taking place.

The issues listed below should be fully resolved in the preliminary *Safety Review*.

The Influence net for *Overall requirements* is in figure 2

<u>Questions on Functional Requirements</u>

Comments:

The set of functional requirements describes what the system is intended to <u>do.</u> This can be expressed by different means, as specification of different degree of formality, textual description, references to existing systems, etc. Particular emphasis should be put on the specification of the boundary and interfaces with the surrounding system. The functional specifications should be sufficiently detailed to form a basis for approval of further development of the system. It should also be suited to be a basis for the necessary hazard and risk analysis to be done. (See reference 8, Chapter 2.1, "Computer-based System Requirements" for more background.)

Q6.1 Are the requirements sufficiently complete in details to approve a further development of the system?

A6.1.1 This is part of the preliminary *Safety Review* process. The safety system's functional requirements should be derived from the *safety analysis* required by SKIFS 1998:1, Chapter 4 and should be implementation independent. The safety analysis should cover all *postulated initiating events* and fault sequences within the design basis of the NPP (all operating modes should be covered), together with transient analyses as appropriate, to determine the effect on the plant of these fault sequences. There should be documented evidence (referenced in the *Safety Review*) that this is has been reviewed by the appropriate specialists to establish that it is complete and correct. The safety system's functional requirements will then depend upon which of the fault sequences it is required to control. For the chosen fault sequences, there should be a precise definition of the plant parameters required to detect the event (process variable and operator signals), and of the output variable that must be controlled (outputs to actuators and indicators), i.e. there should be a precise definition of the system boundaries. The physical ranges of both the input and output parameters should be given. The physical and logical equations linking the input parameters to the outputs should be stated in mathematical form. The required operator actions should also be identified - here a full task analysis may be required. The safety system activation settings and any diagnostic requirements (for identifying failures in input parameters and output actuators) should also be defined. With regard to the functional requirements, an important principle required to reduce common cause failure is that safety systems and control systems should be physically separated, and not share the same equipment or services. Evidence should be provided that due consideration has been given to the use of functional diversity in the detection of a fault sequence. A summary report (with appropriate referencing) should be provided in the preliminary *Safety Review*.

A6.1.2 The non-functional requirements, such as reliability (determined from the PSA) - single failure criterion, separation, segregation, redundancy, diversity need to be specified. Accuracy, response time, required input sampling rates, security, availability, usability and modifiability should all be specified. In addition, there should be a definition of the restrictions and constraints placed on the system by compatibility requirements with other existing systems in the plant, and by the physical and natural environmental constraints. All non-functional requirements should be described in the preliminary Safety Report and reviewed in the preliminary *Safety Review*.

A6.1.3 The impact of the failure of the safety system functions on the fault sequences should be evaluated and, where necessary, additional functional requirements included. This analysis, and the additions required, should be clearly identifiable in the preliminary *Safety Review*.

A6.1.4 All functional and non-functional requirements should be traceable (via a suitable document) to the safety analysis feature from which it was derived. The Preliminary *Safety Review* should provide a summary of this document.

A6.1.5 As part of the *Safety Review* process of SKIFS 1998:1, the functional and non-functional requirements should be validated by an independent group consisting of the appropriate specialists to demonstrate the correctness, completeness and consistency of the safety system requirements. This review should, where appropriate, include animation or simulation of the requirements specification. The documented Preliminary *Safety Review* should provide evidence of adequate coverage of all functional and non-functional requirements with any testing or review being full traceable to the appropriate requirement.

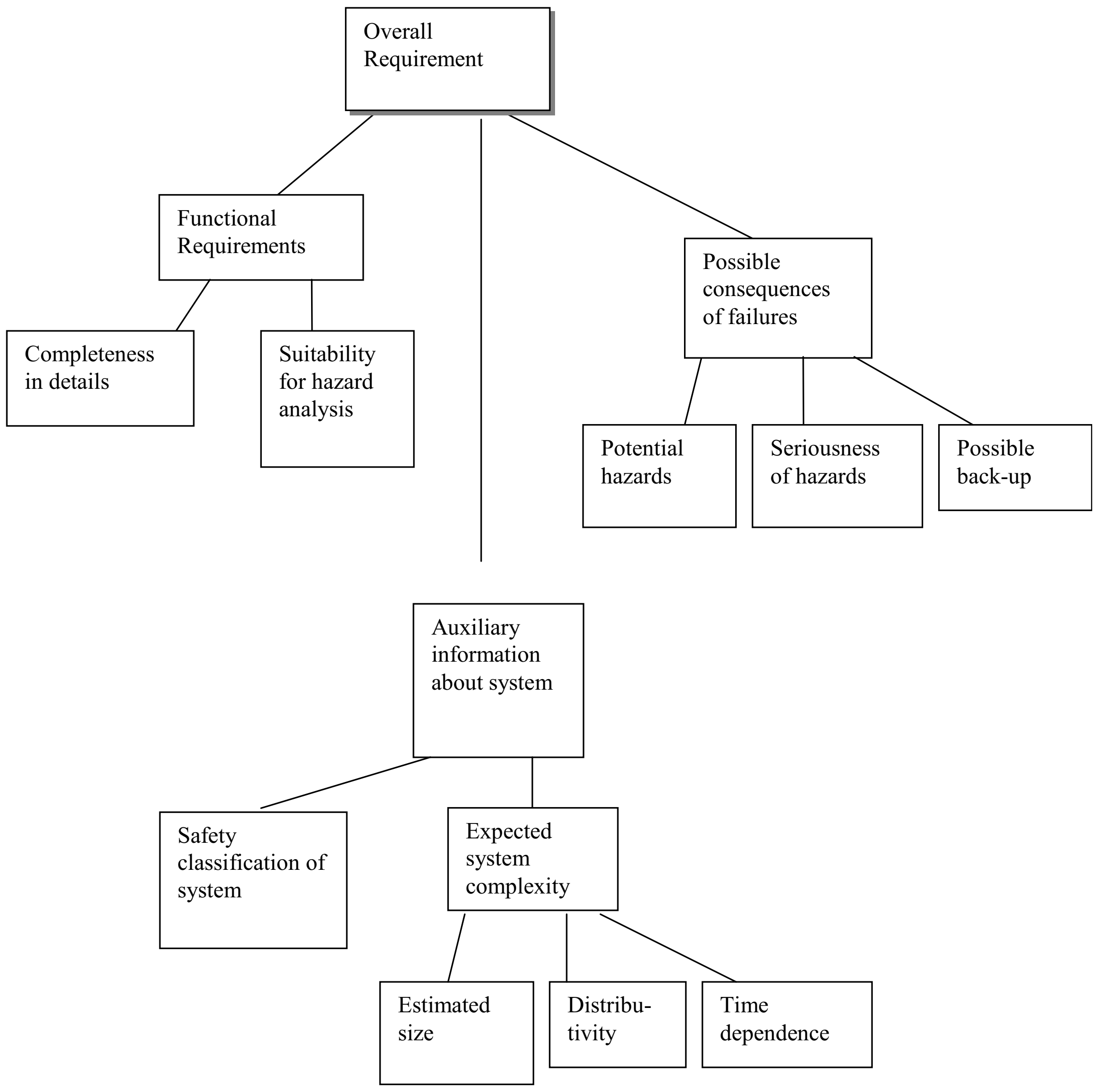


Figure 2 Influence net for milestone *Overall requirements.*

Q6.2 Are the requirements suitable to specify the scope of the necessary hazard and risk analysis to be done?

A6.2.1 This activity forms part of the requirements specification of the safety system (see Q6.1 above). The adequacy of the specification to enable a hazard and risk analysis is a measure of the adequacy of the safety systems specification. Provided the responses to Q6.1 are acceptable, the requirements will be suitable for performing the hazard and risk analysis.

Q6.3 Is the boundary and interfaces with surrounding system sufficiently well specified?

A6.3.1 If the requirements of Q6.1 above are met then the boundary and interfaces will have been specified sufficiently well.

Questions on Expected system complexity

Comments:
The expected complexity of the system influences the decision at this stage. A complex system is more difficult to evaluate, and more error prone than a simple system. Parameters which can be used to estimate the complexity of the system are the estimated size of the system, the degree of distributivity, i.e. whether it can be executed on a single computer, or whether it requires several programmable modules distributed over the plant, and the degree of time dependence expected in the system, etc.

The avoidance of unnecessary complexity is important issue here, and should be a central feature of the design. However, systems that are considered to be complex cannot be rejected provided they have an adequate safety demonstration. Of note though, is the fact that the evidence and reasoning in support of an adequate safety demonstration will need to be greater the more complex the system. For example, a software-based system with a high complexity metric (as say measured by the McCabe Cyclomatic technique - see McCabe, T J: "A Complexity Measure", IEEE Trans on Soft.Eng., Vol 2 No. 4, 1976 pp 308-320) will contain many paths; and hence, will require much more testing effort than one with a low complexity metric. Additionally, the assessment of such a system will need to be more stringent and will be time consuming.

Q6.4 How large would you estimate the magnitude of the planned system will be?

A6.4.1 The most universally known method of estimating the potential size of a system is using Function Points - a system first defined by Allan J Albrecht in the late '70s (see for example, "Function Points in the Estimation and Evaluation of the Software Process", IEEE Trans. Software Eng. Vol. 16, 1990). The size is based on the functions provided by the system without regard to the method of implementation. The method relates more to traditional data processing systems but could be adapted to cover safety systems.

A6.4.2 This should be part of the *Preliminary Safety Review.* Possibly the best method to use to estimate the size of the system is to request the licensee to approach the potential suppliers and ask them to provide an estimate. Because the suppliers have the experience of making such estimates, their values will be the most accurate.

A6.4.3 It should, however, be remembered that size is not a good measure of complexity. A small system may be small because it repeatedly uses code in a complex control flow. Conversely, straight-line code is simple but where it delivers a large amount of functionality it will be large when measured in terms of lines of code.

Q6.5 Will the system most probably be based on a single computer or a set of computers distributed in a network?

A6.5.1 This will depend to a large degree on the number of inputs, outputs and the algorithms that link them plus the time responses required and the operator interactions. There will be advantages to using a distributed system since it enforces modularity and good software structure. In addition, it may be the only way that the time-response requirements can be met. This aspect should be covered by the preliminary *Safety Review*.

Q6.6 Will the system be time critical?

A6.6.1 By its very nature, a safety system must be time critical.

Questions on safety aspects

Comments:

The safety aspect at this milestone is related to the planned system itself, in distinction from the safety aspects at the previous milestone which relates to the safety culture at the company in general. One such aspect is the safety classification, i.e. whether the planned system is assigned to any safety class or safety integrity level.

Another is the scope of a hazard analysis required. To get some idea about this, one needs an overview of the systems to be analysed. This comprises the physical equipment to be included in the hazard and risk analysis, as well as the subsystems which are associated with the hazards. These should therefore be specified. Also relevant types of initiating events, both external events and accident-initiating events in the system itself, like for example component failures, procedural faults, human error, dependent failure mechanisms which can cause accident sequences to occur, should be specified.

One should also take into account the potential hazards identified at the previous milestone, to get an idea about which of them may be relevant with the implementation of the planned system. The seriousness of these hazards will influence the scope of the hazard analysis.

Q6.7 Is the planned system assigned to any safety class or safety integrity level?

A6.7.1 Since this document only considers safety systems the safety class is pre-determined and the safety system must meet the requirements of these guidlines. With regard to safety integrity levels, this will be determined from the PSA but for the majority of safety systems the integrity level will be SIL 3 or SIL 4.

Q6.8 Is the physical equipment to be included in the scope of the hazard and risk analysis, as well as the subsystems which are associated with the hazards, specified?

A6.8.1 Since this is a safety system the physical equipment must be included in the scope of the hazard and risk analysis.

Q6.9 Have external events and accident-initiating events in the system itself, like for example component failures, procedural faults, human error, dependent failure mechanisms which can cause accident sequences to occur, been specified?

A6.9.1 External hazards such as fire, explosion, flood, missiles, aircraft impact, earthquake, toxic gases and electromagnetic interference should have been considered in the safety analysis and should be covered by the preliminary *Safety Review*. They should also form part of the safety systems environment and should be taken into consideration in the design if the non-functional requirements are to be met. As a fundamental principle, an external or internal hazard should not jeopardise the correct, and timely, action of a safety system designed to protect against that same hazard.

A6.9.2 With regard to component failures, procedural faults, human error, dependent failure mechanisms which can cause accident sequences to occur, these can only be specified in any detail once the equipment design (including the software) is known.

A6.9.3 However, failures in the functions that the safety system provides should be analysed to ensure that the required safety properties of the safety system are achieved.

Q6.10 How much will the potential hazards identified at the previous milestone influence the scope of the hazard analysis?

A6.10.1 The failure modes of the safety system plant actuators (valves, pumps etc) must be taken into account when considering fault sequences. Here it should be noted that the computer aspects of the safety system itself will contribute to these failures. It follows therefore, that the potential hazards identified at the previous milestone must influence the scope of the hazard analysis. Failures in the computer-aspects of the safety system should be seen as contributors to the failure rates of the safety system plant actuators. Also, the software may contribute to more complex failures of these plant items through acting as a common cause failure mechanism. This could mean that combinations of plant items might fail simultaneously due to software errors. A software hazards analysis is required to determine the potential for this occurrence. The software hazard analysis can obviously only be performed once the software is designed. A summary report of this analysis should be contained in the final *Safety Review*.

Questions about quality aspects

The following set of questions appear twice - once here and again in Chapter 8 - because the development of a specification and the hazard analysis are an iterative process which needs to be fully documented in the final *Safety Review* to demonstrate its completeness. At this stage in the life-cycle (IEC 61508's "Overall scope definition") there may only be a natural language specification with some mathematical algorithms specified. However, some measure of quality is required in the *Preliminary Safety Review*. Many of the replies have been repeated in Chapter 8 where some are more relevance and should be covered in the final *Safety Review*.

Comments:

Another aspect is the quality of the specification documentation, i.e. whether it has the quality necessary to be the basis for a safe development of the system. Experience tells that poor quality of the specification may easily lead to faults in the final system which are potentially hazardous. A good specification document should have the following attributes:

- *Correctness, .i.e. it should correctly express the intention with the target system. It should also be based on a correct interpretation of the plant and environment.*

Correctness is not directly measurable, but there are activities which enhances the confidence in the correctness of the specifications. A walkthrough of specification documents by an independent team is an effective way of revealing faults.

- Uniqueness*, i.e. there should be only one possible semantic interpretation of each requirement.*

Uniqueness is enhanced if there is a precise definition of all terms, in particular if normal language is used, where the terminology of real world items may have multiple meanings.

- Completeness*, i.e. no requirement or need of the customer is overlooked.*

To obtain completeness a system response should be clearly defined for every possible set of inputs. Textual completeness implies that all referenced terms are defined, and that all referenced tables, figures, sections etc. present and appropriately labelled in the document.

- Consistency*, i.e. there should be no internal contradiction between individual requirements, and terms must not have different meanings at different places within the document.*

Consistency can be observed through consistency checks. This can be automated if a formal specification language is used.

- Verifiability*, i.e. there should be a finite process in which it is possible to verify the final program vs. the specification.*

Verifiability is enhanced by the use of assertions (pre- and post conditions, safety invariants etc). in the specification. This is facilitated by the use of a formal specification language.

All these attribute are facilitated by the use of formal specification methods.

Q6.11 Has a walkthrough of the specifications been made to check correctness?

A6.11.1 A walkthrough of the overall requirements specification by an independent team (as part of the *Safety Review* process) will increase confidence in its correctness provided the team is constituted of members with the appropriate expertise and knowledge. The team should contain plant engineers (from the mechanical, electrical, civil, I&C and chemical engineering disciplines), safety engineers (again with knowledge of all disciplines), computer specialists and representatives of the operations staff (the end users). They should all be fully familiar with the requirements' specifications and other related documents so that they are able to contribute to the walkthrough discussion. The walkthrough should be guided by an experienced and impartial facilitator in the presence of the authors of the specification. All proceedings should be documented and form part of the *Preliminary Safety Review* .

A6.11.2 Such a walkthrough should be mandatory for safety systems since not only does it check correctness, it also checks the other attributes of completeness, consistency and uniqueness.

Q6.12 Has an animation of the specifications been made to validate their functionality?

A6.12.1 The effort required to animate a specification would suggest that it should ould follow at least the first iteration of the hazard and risk analysis. Nontheless, if the overall specification of requirements is sufficiently well developed then animation is recommended. But animation is not considered an essential part of an adequate safety demonstration for safety systems. Provided the requirements are described in a structured manner using mathematics, where appropriate, then the thorough application of the walkthrough technique will form an adequate safety demonstration. However, the walkthrough could be greatly assisted by the use of a suitable animated specification provided the test coverage is adequate and the tests are fully documented. Where an animated specification forms part of the validation process, all the participants in the walkthrough should be fully familiar with the specification language used. The animation process should be fully documented in the *Safety Review*, together with a justification for the chosen animation process, and the tests performed.

Q6.13 Have all essential terms in the specification been uniquely defined?

A6.13.1 This is an essential requirement for a safety system whatever the stage in its development. The licensee must provide an adequate demonstration that he has unique definitions for all the essential terms. Data dictionaries assist in this area. The *Preliminary Safety Review* should provide evidence of this unique definition.

Q6.14 Has the specification been made with tool assistance?

A6.14.1 Tool assistance gives greater confidence in the correctness, completeness, consistency and uniqueness of the specification since it reduces the amount of human error that may be introduced. The licensee can take benefit from this in the walkthrough phase of Q6.11 above since many of the processes of exercising paths through the specification, and recording the results, will be automated. However, it is essential that the tools are appropriately qualified. They should be subject to a QA regime equivalent to that of the safety system they are describing. They should have been subjected to extensive testing to demonstrate their correct operation. The output of any tool should be diversely checked, even if only manually, to ensure that there are no gross errors. There should not be blind faith in the tool. The *Safety Review* should provide a summary description of the tools employed together with a justification for their choice.

Q6.15 Are system responses specified for all possible input to the system from its environment?

A6.15.1 This property of completeness is essential for safety systems, and should be demonstrated to have been achieved, say during the walkthrough phase.

Q6.16 Are all reference terms defined, and all referenced tables, figures, sections etc present and appropriately labelled in the document?

A6.16.1 For safety systems this is an essential requirement for an adequate safety demonstration.

Q6.17 Have consistency checks been made?

A6.17.1 This property of consistency is essential for safety systems and should be demonstrated to have been achieved, say during the walkthrough phase.

Q6.18 Have safety assertions been specified?

A6.18.1 For safety systems, it is essential that all necessary safety assertions be included if the requirements are to be correct, complete and unambiguous. It is possible to establish assertions such as:

truth tables for logical operations;
ranges for input and output parameters for the different operating modes;
trip levels and alarm levels;
definition of unsafe states (e.g. detection of a LOCA with vented containment).

A6.18.2 Many of these safety assertions will be captured in the Technical Specifications required by SKIFS 1998:1 Chapter 5.paragraph 1 and should be developed in parallel with the requirements. The requirements for the safety system should be shown in the *Preliminary Safety Review* to meet the Tech Specs.

Q6.19 What is the degree of formality in the language of the requirement specifications (e.g. formal, semi-formal, structured text, informal text)?(Same as Q8.16)

A6.19.1 At the initial stage the specification may only be natural language supported by mathematical and boolean expressions. This should, however, be subjected to the review described above. This should be described in the preliminary *Safety Review*.

7 Hazard and Risk Analysis

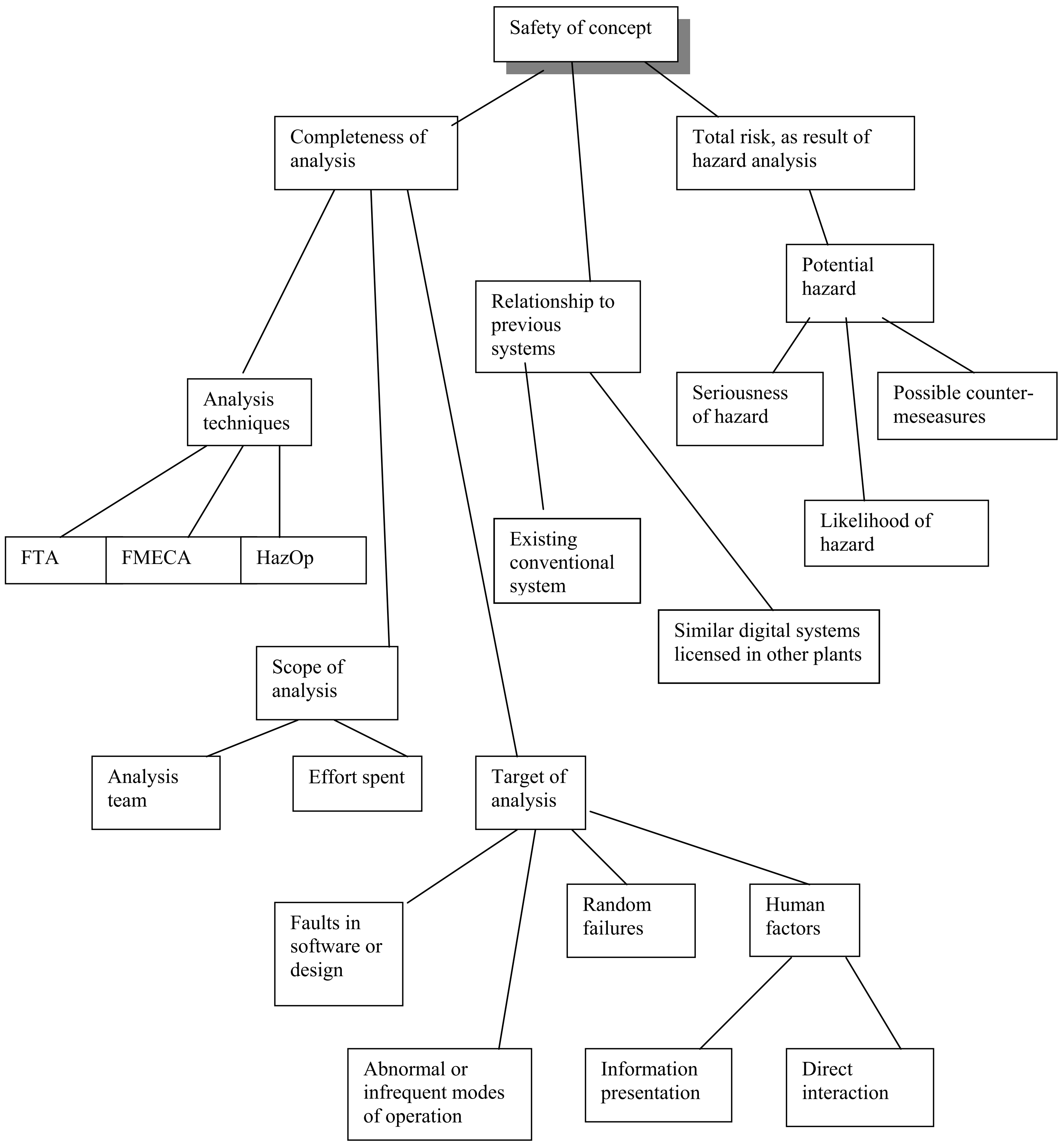
The objective is to determine the hazards and hazardous events the system may have on the plant for all reasonably foreseeable circumstances including fault conditions and misuse. The event sequences leading to these hazardous events should be determined.
A hazard and risk analysis shall be undertaken which shall take into account information from the previous phases. An identification and analysis of all aspects with potential safety consequences should be made before one starts to develop, as well as to assess, a safety related system. If decisions are taken at later lifecycle phases which may change the basis on which the earlier decisions were taken, then a further hazard and risk analysis shall be undertaken.

The hazard and risk analysis shall consider the following:
- each determined hazardous event and the components that contribute to it;
- the consequences and likelihood of the event sequences with which each hazardous event is associated;
- the necessary risk reduction for each hazardous event;
- the measures taken to reduce or remove hazards and risks;
- the assumptions made during the analysis of the risks, including the estimated demand rates and equipment failure rates – any credit taken for operational constraints or human intervention shall be detailed;

The analysis which can be performed at this stage in the project must be based on the functional requirement of the system as well as a deep knowledge of the plant process. For all threats to safety from potential system failures, the consequences should be identified and possible countermeasures suggested (e.g. how to bring the system into a fail safe state).
The goal at this milestone is defined as 'Safety of Concept', and the corresponding Influence net is shown in Figure 3.

The preliminary Safety Report and its Reviews should contain an initial hazard and risk analysis based on the preliminary specification of requirements (this is covered by Chapter 6). There should, however, be a final hazard and risk analysis in the final Safety Report and its Reviews based on the final specification of requirements (this is covered by Chapter 8). This is an iterative process. In this chapter reference to a Safety Review means both the preliminary Safety Review and the final Safety Review. The preliminary Safety Review should be reviewed by SKI before sanctioning continuation of the project (i.e. construction), as per SKIFS 1998: 1, Chapter 4, 2 para.

Figure 3, Influence net for *Hazard and Risk Analysis*



Questions about Completeness of the analysis

Questions about analysis methods

Comments:

One aspect concerning the completeness of the safety analysis is the analysis techniques which are used. Three examples are mentioned in Figure 3, viz. FTA, FMECA and HazOp, but there are several other applicable methods. For each technique one should identify whether it is applied, and whether it is appropriate. The appropriateness of a technique, and the extent to which it will need to be applied, will depend on a number of factors, including:

- *the specific hazards and the consequences;*
- *the application sector and its accepted good practices;*
- *the legal and safety regulatory requirements;*
- *the availability of accurate data upon which the hazard and risk analysis is to be based.*

The expertise of the analysis team can be evaluated on formal qualifications as well as of documentation of previous safety analysis activities.

Q7.1 Which analysis techniques were applied in the hazard and risk analysis?

A7.1.1 The analysis techniques applied in the hazard and risk analysis should be identified in the *Safety Review* with a description of each technique. Information that supports the technique's maturity should also be given.

Q7.2 Were these techniques appropriate?

A7.2.1 SKIFS 1998:1 in Chapter 4.paragraph 1 requires that a Safety Analysis be performed. The section on "Comments on certain paragraphs" indicates that the Safety Analysis should include a Probabilistic Safety Analysis (PSA) to level 1 and level 2 for a reactor facility. The techniques for producing these PSAs are well understood by the professional staff involved in this activity. These should still be applied since they will be appropriate.

A7.2.2 The software-based safety system will contribute to the overall plant risk through its failure to act on demand or through its spurious operation. These two aspects of the system have to be evaluated. At this stage in the development process, however, only the effects of the failure of safety functions can be assessed since neither the hardware nor software has been designed. Hence, the safety system, whether based on analogue or digital technology, will feature in the PSA as a risk reduction element in some fault sequences. However, the termination, or mitigation of the consequences of a fault sequence is performed directly by items of plant equipment such as pumps and valves. These plant items have their own failure mechanisms that need to be analysed and factored into the PSA. They might, for example, fail to actuate on demand (either completely or partially) and thus not terminate the fault sequence. In other cases they might in themselves be initiators of fault sequences through their spurious actuation. The software elements of the safety system (*protection*, *safety actuation system* and *safety system support features*) contribute to the frequency with which each of the failures of safety actuation systems might occur. Also the failure modes of the safety system software needs to be analysed (once the software has been produced - which means that this will be an iterative process) since they will determine the modes of failure of the safety actuation systems.

A7.2.3 Techniques such as Fault (or Event) Tree Analysis and Failure Mode, Effects and Consequences Analysis are the most appropriate techniques for developing the PSA for nuclear power stations. However, Hazard and Operability Analysis (HazOps) - a technique developed for the chemical industry - should be considered since its diverse approach may provide insights into other initiating events. Transient analysis should also be used to determine the fault sequences and the timing budgets for the safety system.

A7.2.4 As mentioned above, techniques for the development of a Safety Analysis and PSA for a nuclear power station are well established. The standard guidance for the Swedish nuclear industry should be followed for the assessment of the Safety Analysis and PSA. The *Safety Review* should provide adequate evidence that all initiating faults and hazards that could lead directly or in combination with other failures to a release of radioactive material from the plant have been included in the PSA. The PSA should also identify the safety functions that are required to operate for each of the Postulated Initiating Events and hazards, and specify the minimum level of performance needed.

Q7.3 Had the analysis team an adequate expertise to perform the analysis?

A7.3.1 The analysis team should contain members from the required engineering disciplines (nuclear, electrical, mechanical, chemical, civil etc) who have the required training in hazard and risk analysis. Their competence should be demonstrated through their qualification, training and experience records which should be summarised in the *Safety Review*. The training records should show that team members are competent to operate any computer codes that may be used in the analysis. The analysis team should have suitable documented guidance to ensure consistent performance when conducting this analysis. A summary description of this guidance should be provided in the *Safety Review*.

Q7.4 Does the documentation of the analysis show sufficient depth and completeness of the analysis?

A7.4.1 The standard guidance for the Swedish nuclear industry should be followed for the assessment of the depth and completeness of the Safety Analysis and PSA.

<u>Questions about Targets of analysis</u>

Comments:

The third aspect is the target of the analysis, i.e. what kind of failures or other malfunctions and their consequences the analysis has addressed. This means identifying types of failures or other malfunctions which may occur with the introduction of the programmable system, and what consequences these may have on safety. Examples of such targets are faults in software or design, random failures (typically hardware failures), failures due to human factors, failures in abnormal or infrequent modes of operation, etc. The objective is to confirm that the company has considered all relevant aspects of introducing a programmable system.

Inherent faults in the design of the software of a system may cause systematic failures during execution. Such failures occur at certain particular situations, as e.g. certain input data to a computer program. A problem with such faults is that they may be the cause of common mode failures.

Random failures in the modules of programmable systems are usually hardware failures, but there are certain characteristics of digital modules which distinguish them from conventional components.

One should in particular investigate the possibility of abnormal situations which occur infrequently. In this respect there are some aspects particularly related to programmable systems. Conventional equipment follows certain physical laws which limit the possibilities of abnormal operation, even in situations which have not been thought of beforehand. For a programmable system such an abnormal situation, e.g. due to sensor failures, may give rather arbitrary results, which may cause hazards. If any such possible abnormal situations have been identified during this investigation, the next question is whether any countermeasures have been proposed, like e.g. plausibility checks or safety assessments.

A further objective is to identify whether human interactions during operation will have any effect on the planned system, and whether any of these effects may cause a hazard. If so, the next question is whether there are any precautions proposed which can help the operator to correct the information or to help the operator to understand that information is incorrect. Another problem arises if the operator is supposed to directly interact with the system, e.g. through manual input of data (e.g. calibration data or set points). In such cases there should be precautions to check the correctness of these data.

An essential aspect to investigate here is whether countermeasures exist against the consequences of such failures. A particular problem is whether the planned countermeasures will prevent all hazards of such faults, i.e. are there some less serious hazards which are not covered by the countermeasures.

As an example: for a neutron flux based trip channel with a temperature/ pressure based trip as back up;if the neutron flux based trip channel fails then the slower response time of the backup system may result in damage in the core.

Q7.5 Have all different types of possible fault which can cause hazards been considered in the hazards and risk analysis?

- *Systematic failures in design or software.*
- *Random failures.*
- *Abnormal or infrequent modes of operation.*
- *Faulty human interactions.*
- *Incorrect information presentation.*

A7.5.1 The types of failure that need to be considered, as a minimum, in the *Safety Review* are:

Instruments out of range or inoperable;
Erroneous operator inputs;
Failures of output signals to actuators;
Failure of actuators;
Erroneous information displayed to operators;
Software attempting to divide by zero;
Software attempting to write outside an array.

The various countermeasures for these errors are covered by the responses to the next question.

Q7.6 Has the possibility for countermeasures against these types of failures been considered?

A7.6.1 The technique of incorporating error checking into software is known as defensive programming. The incorporation, into a program, of mechanisms which detect and respond in a predetermined safe manner to erroneous data values and control flow will ensure countermeasures have been considered. Defensive programming techniques "should cover both internally and externally arising exceptions, without adding unnecessary complexity to the software" (reference 8, section 2.3.2.4). Below are listed countermeasures that should be considered. The *Safety Review* should contain a description of, and justification for, the countermeasures chosen (see below some techniques taken from references 3, 8 & 12). These should be further developed (with a summary description in the *Safety Review*) by means of a software hazard analysis applied to the completed software.

A7.6.2 In general, the safety system's hardware and software architecture should be such that the system operates in a predefined safe manner in the presence of hardware and software faults. Techniques such as the use of alternate data sources and redundant architectures should be considered. (reference 3, section 6.19)

A7.6.3 Defensive programming techniques should be employed where known states and parameter ranges occur. For example, where the state of a valve can be either open or closed, a check might be made that, after a finite change-over time, one state or the other is achieved. This should be done by checking the valve states for *open AND not closed*, or *closed AND not open*. In the case of parameter ranges, if the output of a computation returns an unrealistic answer then an error should be flagged to the operator. The fact that the final actuated equipment has achieved the desired state in the required time should be verified. (reference 8, section 2.3.3.1.4 & 2.3.3.1.5)

A7.6.4 A hardware watchdog, serviced by a program operating at the base level of priority, should be used to detect problems such as programs locked in loops or deadlocked or non-running of the computer system. Software routines should be incorporated into the watchdog service routine that check the integrity of standard arithmetic operations. (reference 12, section 51)

A7.6.5 Data errors occurring in inputs and on transfer between modules should be trapped and alarmed; instrument readings should indicate when they are off scale. (reference 8, section 2.3.4.1.5)

A7.6.6 If complicated calculations are necessary, the program should be written so that - where feasible - a simple function or assertion will provide an error check and back-up action. (reference 8, section 2.4.3.6.1)

A7.6.7 Defensive programming should be employed with the objective of maintaining program behaviour within its specifications (anomalous behaviour should be detected, safe action taken and the operator should be informed). (reference 8, section 2.4.3.6.2)

A7.6.8 Known states and program properties, and parameter ranges should be verified, and variable values likely to cause illicit operations should be detected and safe actions taken, including the raising of an alarm (for example, division by zero - its detection might be via a program interrupt). Dynamic checking for overflow and underflow should be performed for arithmetic operations. Array bounds, lists, queues, buffers should be checked for out of range values. (reference 8, section 2.4.3.6.2 & 2.4.3.6.3)

A7.6.9 The compatibility between the precision used to represent numerical values should be guaranteed and if necessary verified dynamically. (reference 8, section 2.4.3.6.4)

A7.6.10 As a general rule, the system should be designed, as far as practicable, to reveal failures. Diagnostic software can be used to detect hardware failures. The memory should incorporate automatic (preferably hardware based) parity and checksum testing to detect bit change errors. Attempts to write to read-only or protected memory should be detected. Hardware failures which are not self-revealing should be deemed to exist until the next proof test that would reveal them. These proof tests should be defined. (reference 12, section 71)

A7.6.11 Correct operation of analogue-to-digital converters should be checked by reading reference voltages. (reference 12, section 73)

A7.6.12 Communications messages should incorporate checksums and message lengths. (reference 12, section 74)

A7.6.13 At initialisation the system should check its ability to write to and read from all installed memory and address all required hardware modules. (reference 12, section 75)

A7.6.14 Where possible, power supply voltage levels should be checked. Also, the ability of a signal line to perform its specified function should be tested automatically. (reference 12, section 76)

A7.6.15 Assertions should be inserted into the code and the system should be made as fault tolerant as possible. For example, data errors occurring in inputs and on transfer between modules should be trapped and alarmed; instrument readings should indicate when they are off scale; correct output address selection pre- and post-operation should be verified. (reference 8, section 2.3.4.1.5)

A7.6.16 When considering the use of a software-based, safety system (as with an analogue safety system), it is necessary to review the failure modes of the outputs of the system since these will determine the behaviour of the plant item. Digital outputs that activate safety plant items should employ at least 2-out-of-3 voting so as to improve the reliability of operation and reduce the number of spurious operations.

A7.6.17 Operators should be requested to confirm their actions. Security measures (see Q9.1.5, A11.16.2 and A11.16.2.1) should be in place to ensure that only authorised personnel perform actions that might have a hazardous consequence. Safety checks should be performed on all operator actions. Operators should not be able to over-ride correct safety-system action.

Q7.7 Will the planned countermeasures prevent all hazards of such faults?

A7.7.1 It is not possible to be 100% confident that the use of the above countermeasures will prevent all hazards of such faults but a high level of confidence should ensue. The safety analysis should incorporate the software-based safety system, and it should be shown in the *Safety Review* that consideration has been given to incorporating diverse means of achieving the safety functions. These should be analysed to ensure that they provide the required risk reduction for the particular fault sequences that include the failure-to-act of the safety system. Where spurious operation is concerned, the safety analysis should cover the spurious operation of a single safety actuation plant item or a whole system (such as a PWR's auxiliary feed to the steam generator being activated). In the trip situation, consideration should be given to the spurious operation of one of the safety actuation plant items not required to terminate or mitigate the consequences of the particular fault required. Whilst considering the diverse safety functions, consideration should be given to common-mode-failures since software can be a common-cause-failure mechanism - this might be significant where a particular safety action depends upon a feature of a software-based control and surveillance system (an analysis of potential software common cause failure mechanisms of the final software should be included in the *Safety Review*).

Questions about Relationship to previous system:

Comments:

A particular aspect is the relationship to previously analysed systems, e.g. if the system is a replacement of an existing system. If the hazard analysis is based on a previous analysis, a question is whether there are essential differences between the existing conventional systems and a new programmable system which have been reflected in the analysis. A particular problem to take into account is that a digital system takes a finite time to perform an action whereas an analogue system starts to act instantaneously, albeit with a particular transfer function.

A source of information which would be very useful for the safety evaluation of the target system would be previous safety reviews and rationales for approval of similar digital systems installed in other NPPs somewhere around the world.

Q7.8 Is the new system a replacement of an existing system?

If yes:

- *Q7.9 Is the hazard analysis based on the hazard analysis of the existing system?*

A7.9.1 The hazard analysis for a software-based safety system should be based on the existing system if the safety functionality is the same. However, it should take into account that the replacement utilises software, and that the human interface might be different

- *Q7.10 Have the essential differences between the existing conventional systems and the new programmable system been reflected in the analysis?*

A7.10.1 The essential differences between the existing, conventional systems and the new programmable system must be reflected in the analysis for the reasons given in Q7.5 above. The *Safety Review* should provide a specific discussion of these issues in this case.

Q7.11 In particular, does the fact that a computation takes time introduce any problems?

A7.11.1 The transient analysis should indicate the time responses for the safety functions. These time responses should, when the dynamic performance of the associated safety plant items are taken into account, enable the safety system's timing budgets to be determined. The response of the safety system should be within these timing budgets. A second consideration is the sampling rate of the inputs. Account should be taken of the frequency spectrum of the input, and the rate of change, when determining the sampling rate. The sampling rate should be such that signals are not aliased or trip and actuation levels are not significantly breached before detection. These aspects and the solutions offered should be discussed in the *Safety Review*.

Q7.12 Have similar digital system(s) been installed in other plant(s) around the world?

A7.12.1 The fact that similar digital systems are installed elsewhere gives some comfort to the regulator but it has to be remembered that the operational profile in these other plants may not match that of the target installation. A software error can lie dormant in a system for many years simply because the input conditions have been such that the program path containing the error has never been executed. This can then give a false indication of its reliability. Then, when the software is applied to another facility the particular path is traversed and the software fails. This point must always be borne in mind when considering previous use of the software.

Q7.13 If yes:

- *Have they been approved for safety critical application by independent authorities?*

A7.13.1 The fact that an independent authority has approved a similar digital system(s) for safety critical applications should not mean that a software-based safety system is accepted for safety duties on a nuclear power plant without a thorough review (to the standards of this guide). The approval by an independent authority is a valuable contribution to the safety demonstration but the regulator must still be satisfied that all the documentation is available for review and that it proves to be adequate. The *Safety Review* should provide a full description of the approval procedures and should show how the software meets the requirements of this guide.

Q7.14 Is the documentation of the underlying safety reviews of the system(s) available?

A7.14.1 An adequate safety demonstration for software-based, safety systems in NPPs requires that all the documentation of any previous safety review should be available. If it is not available, then the fact that a safety review has been performed should only be used in support of a safety system requiring a low reliability. This, of course, assumes that the safety review has been performed by a reputable body. For a safety system, the totality of the documentation should be described in the *Safety Review* and its acceptability justified.

<u>Questions about the Results of the analysis:</u>

Comments:
The result of the analysis is the identification of potential hazards to safety that failures in the new system may introduce. For each of these hazards the analysis should identify how serious the consequences of the hazard will be on the plant, people and environment.(the risks). One should also estimate the likelihood of these consequences, and the possible counter-measures which may be done to prevent serious consequences, for each of these hazards.

For each of the identified potential hazards identified as result of hazard analysis the following questions should be considered:

- *Q7.15 How serious are the consequences that the hazard can lead to?*

A7.15.1 Each identified potential hazard should be considered a potential initiating event which leads, in combination with other plant faults as appropriate, through a fault sequence to a particular consequence. The Safety Analysis should cover this, and it should be discussed in the *Safety Review*.

- *Q7.16 How likely is it that these consequences will happen in connection with the hazard?*

A7.16.1 The answer to this question is given by the PSA.

- *Q7.17 Is it possible to specify any countermeasures to these consequences?*

A7.17.1 These should have been considered in the Safety Analysis (see response to Q7.7) and discussed in the *Safety Review*.

8 Requirement Specifications

This milestone is where the result of the safety analysis is incorporated in the requirement, to make a more detailed requirement specification where the required safety and integrity level is reached. This milestone corresponds to lifecycle phase 4, and partly to phase 5 of IEC 61508. To quote from Part 1, section 7.5.2: Requirements: "The safety functions necessary to ensure the required functional safety for each determined hazard shall be specified. This shall constitute the specification for the overall safety functions requirements. The safety functions to be performed will not, at this stage, be specified in technology-specific terms since the method and technology of implementation of the safety functions will not be known until later."… "The necessary risk reduction shall be determined for each determined hazardous event. The necessary risk reduction may be determined in a quantitative and/or qualitative manner."

The goal for this milestone, dependable specification, is a measure of the degree to which the results from the safety analysis is incorporated and conform with the general safety and reliability requirements. This is thus a measure of the 'safeness' of the specified system. As the specification is the basis both for the development and verification of the system, it is important to ensure that the safety target at this milestone is reached.

There are two main aspects that will influence the decision on whether the specification satisfies the requirements of a safe system, viz. the safety aspect and the reliability aspect, as indicated on the influence diagram at Figure 4. The first relates to whether the specified system is safe, i.e. whether all aspects identified in the safety analysis are properly specified to the required safety level. The reliability aspect is a measure of the evidence, at this milestones, which supports the view that the development activity has produced a system which performs its specified functions with low failure probability.

Provided that there is an acceptable project plan (as discussed in Chapter 9), from this stage on it is assumed that the Preliminary Safety Report and its Reviews have been accepted by SKI, and that construction (i.e. the project) is being allowed to commence (as per SKIFS 1998:1 Chapter 4, 2 para). There will need to be further refinements of the requirements, and further hazard and risk analyses (on an iterative basis) but these should be covered in the final Safety Report and the final Safety Reviews. All the subsequent chapters will be regarded as applying to the final Safety Review.

Figure 4 is the Influence net.

Questions about Safety Aspects:

Comments:
During the three first phases in the assessment, an evaluation of the safety of the concept is sufficient to accept further development. A positive answer to this is required.

This also implies that the principle of 'defence in depth' is properly considered in the specification (see reference 10, sections 2.9 to 2.11 and sections 4.1 to 4.4 for guidance on 'defence in depth'). This implies again that safety defences are specified for the most serious hazards identified for the previous milestone.

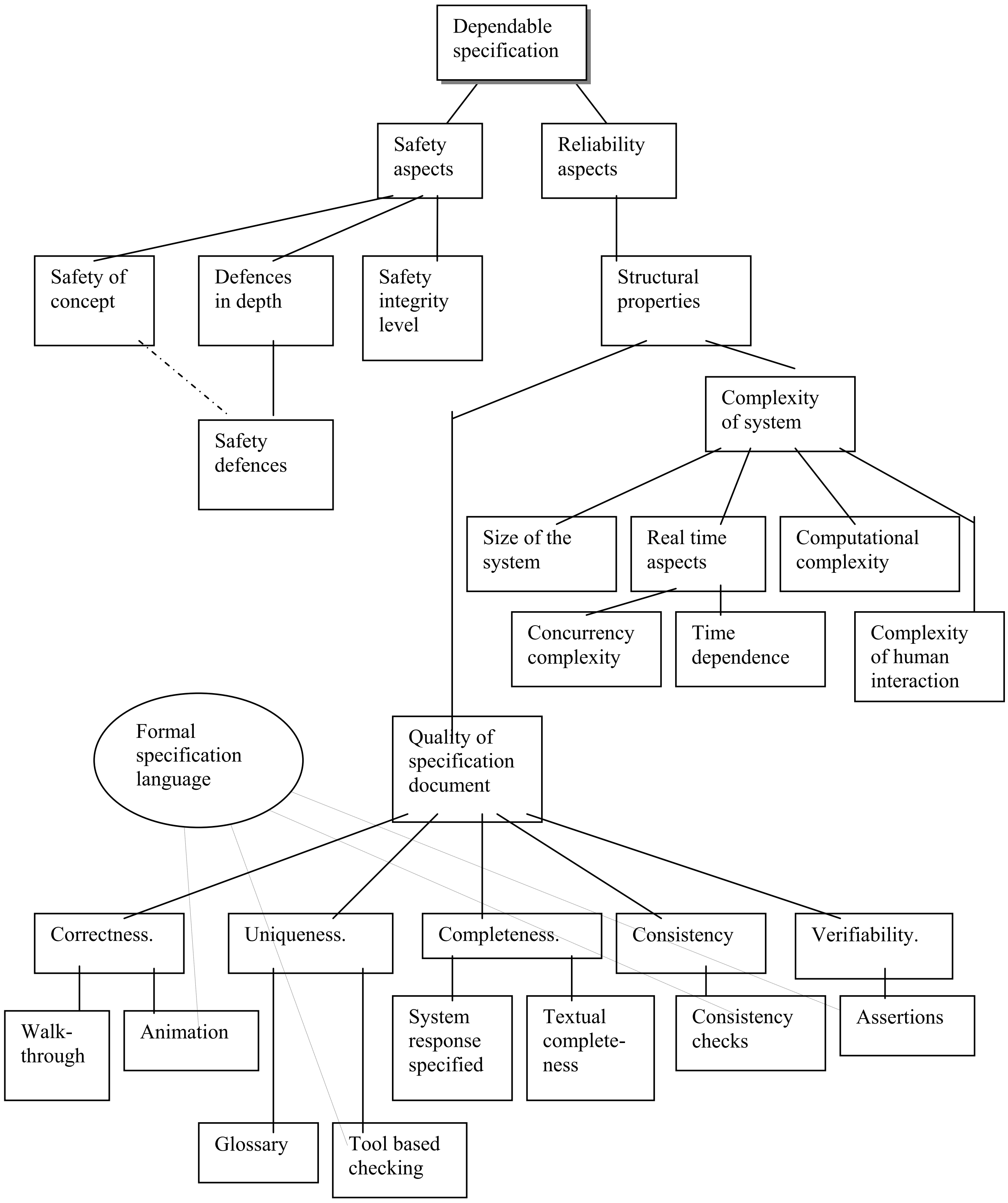


Figure 4 Influence net for milestone *Requirement specifications.*

The safety integrity level is required for this system, according the Company's classification scheme (see milestone 1) should be given.

Q8.1 Is the safety of the concept, as evaluated in previous milestones, acceptable?

A8.1.1 The earlier phases must be revisited if the safety of the concept is not acceptable. In fact, SKI should have given its consent for the project to commence by this stage.

Q8.2 Are appropriate safety defences required?

A8.2.1 Appropriate safety defences will be required since the Safety Analysis (required by SKIFS 1998:1, Chapter 4, para 1) will have identified events, event sequences and conditions that can lead to a radiological accident - this should also include events arising from failures in the safety system itself. The safety functions to be provided by the safety system should be such that events and event sequences are either terminated or their consequences satisfactorily diminished. The purpose of the safety system (protection system) is set out in reference 11, section 1.1. The determination that this is the case will be through the safety analysis (see reference 10, sections 5.69 to 5.73, "Safety analysis"). Further guidance on the elaboration of the functional requirement of the safety system by means of the safety analysis is given in reference 11, section 4, "Design basis". Reference 11 indicates that this elaboration is an iterative process throughout the plant design stages. Importantly, upon completion of the design, the recommendation of reference 11 that the design bases should be documented according to the requirements of reference 11, section 11, "System documentation" should be followed. The *Safety Review* should contain evidence that the safety analysis and the system documentation (as defined in reference 11 above) are comprehensive, correct and complete.

Safety systems in nuclear power plants must, in themselves, contain design features with adequate safety defences (these can be seen as deterministic design principles relating to the safety system - see reference 11, section 5, "Causes of protection system failures"). With regard to computer- based safety systems, the *Safety Review* should demonstrate that they meet, as a minimum, the following principles.

i) The single failure criterion (SFC), as defined in reference 10, sections 5.34 to 5.39 inclusive, should be met (see also reference 11, sections 6 & 7.5). The SFC is assumed to apply to single *random* failures. Faults in software are considered to be systemic faults and should be seen as a source of common cause failure in redundant systems running common software.

ii) At least two functionally diverse methods should be used to detect a fault sequence "preferably by the use of different variables and in the initiation of the safety systems action to terminate the sequence.." (see reference 13, principle P182 and reference 11, section 7.7(1)).

iii) The frequency of spurious safety system operation that might degrade safety, directly or indirectly, should be minimised by design (see reference 13, principle P197, reference 10, sections 3.8 & 5.36 and reference 11, section 7.4, "Spurious operation"). This latter principle should apply during maintenance and testing of the safety system at all times when it is performing any of its safety duties.

iv) No fault, internal or external should disable the safety system that is intended to protect against that fault (reference 13, principle P73, reference 10, section 5.37(1) and reference 11, section 3(2)). The safety system should also meet the requirements of paragraph 6.82 of reference 10.

v) Safety systems and control systems should be physically separate and should not, so far as is reasonable practicable, share equipment (including software) or services (reference 13, principle P73), i.e. safety systems should be independent from the control system as defined in reference 10, Appendix II. The protection and control systems should meet the requirements of reference 10, section 6.86 and reference 11, section 7.8.4. This avoids a potential common-cause failure mechanism.

vi) There should be adequate physical separation)(using, for example, distance, geometry and barriers as appropriate) in the layout of the safety system so as to minimise the effects of internal and external hazards or any failures in other systems. Reference 10, Appendix II, "Redundancy, Diversity and Independence" and reference 11, section 7.8.2, "Physical separation" give guidance. In particular, the requirements of reference 10, Appendix II.12(2) and II.13 should be met.

vii) Safety systems should act automatically in normal circumstances. There should be no necessity for human action for approximately the first 30 minutes following a demand for protective action (see reference 13, principle P77 and reference 11, section 7.2, "Initiation of protective action"). Notwithstanding the above, plant personnel should be able to activate any safety function and perform any necessary safety actions that they perceive to be in the interests of safety (see reference 10, section 6.84 which indicates that the safety system should not be able to "negate correct operator actions in design basis accidents"). But at no time should they be able to negate correct safety system action. This principle is supported by reference 10, sections 4.1(4), 5.28, 5.56 and 6.84.

viii) Safety systems should be designed, so far as is reasonably practicable, to fail in a safe manner (see reference 13, principle P196). The requirements of reference 10, section 5.40, "Fail-safe design" should be met. Some additional guidance is given in reference 11, section 7.15, "Fail safe design".

ix) All plant parameters used to invoke a safety function should be shown to be appropriate and sufficient for that duty (see reference 13, principles 191 &192). The requirements of reference 11, section 7.13, "Sensing of variables indicative of PIEs" should be met.

x) There should be adequate provision for monitoring plant safety and taking the necessary safety actions in a central control room and at an emergency location (see reference 10, section 5.29 & 5.30). The requirements of reference 10, sections 5.48 to 5.55 on "Design for optimal performance" should be met. For the safety system, the requirements of reference 11, section 8, "Safety system monitoring" should be met. Guidance on the equipment to be provided at any emergency location is provided by reference 11, section 7.8.3, "Supplementary control points".

xi) Safety system actions, and the associated alarms, should not be self-resetting irrespective of the subsequent state of the initiating fault (see reference 13, principle P187). This is designated as "seal-in" by reference 11, section 7.2, "Initiation of protective action".

Guidance on resetting the safety system is given in reference 11, section 7.3.3, "Resetting of the protection system".

xii) There should be a direct, known, timely and unambiguous relationship between the plant parameter(s) used by the safety system and the fault sequence it is designed to detect and protect against (see reference 13, principle P191. Where this is not achievable, the parameter chosen should have a known relationship with the fault sequence (see reference 13, principle P192). Again the requirements of reference 11, section 7.13, "Sensing of variables indicative of PIEs" should be met.

xiii) Faults in the safety system itself should be revealed at the time of their occurrence (see reference 13, principle P196 and reference 11, section 8.2, "Status displays"). The requirements of reference 10, section 3.8 should be met.

xiv) A safety system should be dedicated to performing its safety functions (see reference 13, principle P193). If other functions are incorporated, then the whole system should be classified as a safety system. The safety functions should not be jeopardised by the other functions. This is further amplified in reference 11, section 7.8.4, "Interaction between safety systems and other systems".

xv) Changes to the safety system during operation should be only by suitably engineered means under strict administrative control (see reference 13, principle P194).

xvi) Connection of systems external to the safety system should only be via suitable isolation features (see reference 13, principle P195).

xvii Unnecessary complexity should be avoided (see reference 13, principle P196). This is particularly important when considering a computer-base safety system.

xviii) During low power and shutdown states there should be a limit on the unavailability of the safety system as required in reference 10, section 5.25 and 5.26.

xix) Severe accidents due to multiple failures of safety systems should be reviewed as required by reference 10, section 5.31.

xx) Reasonable on-line maintenance and testing of the safety system without the need to shut down the plant should be possible, as required by reference 10, section 5.42. The requirements of reference 10, section 6.83 should be met. Guidance on the testing requirements is given in reference 11, section 7.10, "Testability". Guidance on safety system maintenance requirements is given in reference 11, section 7.16, "Maintenance, repair and calibration".

xxi) A suitable emergency power supply for the safety system should be provided as per reference 10, sections 6.88 and 6.89.

xxii) The safety system should perform its protective tasks in all modes of plant operation. This may require the use of operational bypasses. These operational bypasses should meet the requirements of reference 11, section 7.11, "Operational bypasses".

xxiii) The safety system actions should have priority over unsafe control system actions (see reference 10, section 6.80(3) and reference 11, section .8.4(5).

Q8.3 Which safety integrity level is required for this system?

A8.3.1 The PSA will indicate the safety integrity level required. The fault sequence that requires the greatest risk reduction will determine the reliability of the safety system in terms of failures per demand. In addition, the PSA can be used to determine an acceptable failure-rate for the safety system through a sensitivity analysis. This can then be translated into SIL level by referring to IEC 61508. However, the claimed safety integrity level of the safety system should be shown to be conservative as required by reference 10, section 6.78. Notwithstanding the SIL level determined as required by this approach, the software and hardware of the safety systems of NPPs should satisfy the requirements of this guide.

Questions about reliability aspects:

Questions about the complexity of the system

Comments:
In general the expected reliability is negatively correlated to the complexity of the system. There is therefore a need to define a complexity metrics based on the specification of a system, This metrics should be based on several different types of data about the system, like the size of the system (not always a good measure, see A6.4.3), computational (algorithmic and logical) complexity of the system, real time aspects like concurrency complexity (i.e. whether many functions shall be executed simultaneously and communicate with each other), whether timing is crucial etc. A particular aspect is whether the systems contain man-machine interfaces which implies complex human interaction.

There are no well known metrics based on the specification document. However, some guidance might be obtained from Ebert, C and Oswald, H, "Complexity Measures for the Analysis of Specifications of (Reliability Related) Computer Systems", Proc. of the IFAC SAFECOMP, London, 1991.

A particular problem is that a specification often is in a textual form without a fixed syntax (not like a programming language). But it should anyhow be possible to measure quantities as 'number of lines', 'number of decisions', 'number of concurrent processes' etc. An alternative may be to make judgement directly on the complexity nodes based on predefined scales (e.g. large complexity, medium complexity, low complexity, etc.)

Q8.4 What is the size of the system?

A8.4.1 The system has not yet been defined according to the introduction to Chapter 8, so no accurate size can be given. See the reply to Q6.4 for an estimation technique. It should also be remembered that only unnecessary complexity should be avoided.

A8.4.2 The main issue here is that the difficulty of making an adequate safety demonstration increases with increased complexity. However, the system must be fit for purpose in all respects. These two aspects have to be borne in mind when choosing a particular implementation.

Q8.5 What is the computational complexity of the system measured in the number of decisions?

- *Algorithmic complexity*
- *Logical complexity*

A8.5.1 The functional specifications of the safety systems of nuclear power plant do not exhibit a great deal of algorithmic or logical complexity since they are mostly ascertaining whether or not a parameter has moved beyond a particular safe level, and then operating safety actuation plant items such as pumps and valves, in a relatively simple sequence, to achieve a safe plant state. There are some parameters such as *departure from nucleate boiling ratio* in a PWR that require a complex algorithm but these are few.

A8.5.2 In theory, each safety function could be performed by a separate channel containing its own computer operating in a deterministic, sequential mode. This is often, however, not cost effective.

A8.5.3 The *Safety Review* should provide these metrics since it indicates that potential degree of difficulty of achieving a satisfactory safety demonstration.

Q8.6 Does the requirement imply the necessity of concurrent processing?

A8.6.1 The functional requirements of a safety system should not imply the necessity for concurrent processing since, as stated in Q8.5 above, each safety function could be achieved by means of a single computer. Voting could be achieved through conventional relays. Any requirement for concurrency should be removed from the functional specification.

Q8.7 Does the requirement imply the time dependence of some processes?

A8.7.1 The requirement must give the time dependencies for safety actions in order to ensure that the safety system is fit for purpose. See response to Q7.11.

Q8.8 Will the system contain man-machine interfaces which implies complex human interactions?

A8.8.1 A safety system must have a man-machine interface since the operator requires information on plant states, alarms, trip conditions and must be able to initiate safety actions. System maintenance also requires a suitable interface.

A8.8.2 Operator actions required for the safety functions should be subject to task analysis to determine the feasibility of performing the tasks. The interfaces and the procedures should be designed in accordance with human capabilities. Proper design of the human-machine interface plus adequate procedures and training should ensure that complex human interactions are avoided. The *Safety Review* should contain evidence that a full task analysis has been performed which demonstrates the feasibility of performing the tasks (IEC 60964, section 3.2 gives some indication of requirements). Evidence is also required of the use of suitable procedures and training through documented walkthrough exercises.

A8.8.3 A general principle worth following (see A8.2.1(vii) above) when considering operator actions is that no operator action should be necessary for a specified period of time at the start of an incident (i.e. following a trip). The operator should, however, be able to perform actions during this time which move the plant to a safer state. However, the operator should not be able to over-ride safe protective action (see Appendix A3, "Defence in depth" in reference 9, standard IEC 61513, for guidance on the role of automation and prescribed operator actions in the management of anticipated operational occurrences and accident conditions).

A8.8.4 The human-machine interface design should comply with the requirements of section 5.3.1.2 of standard IEC 61513 (reference 9).

Questions about quality aspects:

Comments:
Another aspect is the quality of the specification documentation, i.e. whether it has the quality necessary to be the basis for a safe development of the system. Experience tells that poor quality of the specification may easily lead to faults in the final system which are potentially hazardous. A good specification document should have the following attributes:

- *Correctness, .i.e. it should correctly express the intention with the target system. It should also be based on a correct interpretation of the plant and environment.*

Correctness is not directly measurable, but there are activities which enhances the confidence in the correctness of the specifications. A walkthrough of specification documents by an independent team is an effective way of revealing faults.

- *Uniqueness, i.e. there should be only one possible semantic interpretation of each requirement.*

Uniqueness is enhanced if there is a precise definition of all terms, in particular if normal language is used, where the terminology of real world items may have multiple meanings.

- *Completeness, i.e. no requirement or need of the customer is overlooked.*

To obtain completeness a system response should be clearly defined for every possible set of inputs. Textual completeness implies that all referenced terms are defined, and that all referenced tables, figures, sections etc. present and appropriately labelled in the document.

- *Consistency, i.e. there should be no internal contradiction between individual requirements, and terms must not have different meanings at different places within the document.*

Consistency can be observed through consistency checks. This can be automated if a formal specification language is used.

- *Verifiability, i.e. there should be a finite process in which it is possible to verify the final program vs. the specification.*

Verifiability is enhanced by the use of assertions (pre- and post conditions, safety invariants etc). in the specification. This is facilitated by the use of a formal specification language.
All these attribute are facilitated by the use of formal specification methods.

Q8.9 Has a walkthrough of the specifications been made to check correctness? (see Q6.11 for same question)

A8.9.1 Evidence of a walkthrough of the specification must be provided in the *Safety Review* (see Q6.11).

Q8.10 Has an animation of the specifications been made to validate their functionality? (see Q6.12 for same question)

A8.10.1
See response at A6.12.1.

Q8.11 Have all essential terms in the specification been uniquely defined? (see Q6.13 for same question)

A8.11.1 This is an essential requirement for a safety system. The licensee must provide an adequate demonstration in the *Safety Review* that he has unique definitions for all the essential terms. Data dictionaries assist in this area.

Q8.12 Has the specification been made with tool assistance? (see Q6.14 for same question)

A8.12.1 Tool assistance gives greater confidence in the correctness, completeness, consistency and uniqueness of the specification since it reduces the amount of human error that may be introduced. The licensee can take benefit from this in the walkthrough phase of Q6.11 above since many of the processes of exercising paths through the specification and recording the results will be automated. However, it is essential that the tools are appropriately qualified. They should be subject to a QA regime commensurate with that of the safety system they are describing. They should have been subjected to extensive testing to demonstrate their correct operation. The output of any tool should be diversely checked, even if only manually, to ensure that there are no gross errors. There should not be blind faith in the tool. The *Safety Review* should provide a summary description of the tools employed, together with a justification for their choice.

Q8.13 Are system responses specified for all possible input to the system from its environment? (see Q6.15 for same question)

A8.13.1 This property of completeness is essential for safety systems and should be demonstrated to have been achieved, say during the walkthrough phase. A traceability matrix could be used to link system responses to the appropriate inputs to demonstrate completeness. The *Safety Review* should demonstrate that all possible inputs from the safety system's environment have appropriate responses specified.

Q8.14 Are all reference terms defined, and all referenced tables, figures, sections etc present and appropriately labelled in the document? (see Q6.16 for same question)

A8.14.1 For safety systems this is an essential requirement for an adequate safety demonstration; the *Safety Review* should demonstrate that this has been achieved. The use of a Data Dictionary should be considered. Documented evidence of a review process should be sought.

Q8.15 Have consistency checks been made? (see Q6.17 for same question)

A8.15.1 This property of consistency is essential for safety systems and should be demonstrated to have been achieved, say during the walkthrough phase. Evidence that consistency checks have been successfully made should be provided in *the Safety Review*.

Q8.16 Have safety assertions been specified?(see Q6.18 for same question)

A8.16.1 The safety assertions used should be described in the *Safety Review*, and their choice justified.

Q8.17 What is the degree of formality in the language of the requirement specifications (e.g. formal, semi-formal, structured text, informal text)? (see 6.19 for same question)

A8.17.1 For safety systems, there is a mandatory degree of formality that precludes the sole use of informal or structured text. However, it is of great importance that all aspects of the requirements' specifications are fully understood by all parties involved in their production and validation. The degree of formality needs to be such that the requirements are unambiguous and accurate (i.e. complete and correct). Even when there is a formal definition of the requirements these should be supported by informal or structured text. As a minimum, mathematical and logical expressions should be used which are supported by diagrammatic formalism such as finite state machines or Boolean diagrams. The choice of language for the requirements' specification should be justified in the *Safety Review*.

A8.17.2 For an adequate safety demonstration, the licensee should provide training records of the personnel involved in the specification-of-requirements process to demonstrate that they have the requisite expertise. This is particularly important where a requirements' specification language is used. These should be summarised in the *Safety Review*.

9 Project Planning

It should be noted that the planning phase is not restricted to one milestone in the lifecycle, but runs in parallel throughout the whole system development. IEC-61508 distinguishes between Overall planning' and more specific planning in the system realisation phase. It is, however, the overall planning which is the topic of this section. IEC-61508 also distinguishes between three types of overall planning: Operation and maintenance planning, Safety validation planning and Installation and commissioning planning. These are, however, different targets of the planning, whereas this section emphasises the structure of the planning, and therefore treats the overall planning as a unity, covering all further activities in the system lifecycle.

The assessment objective at this milestone is to show that the company can demonstrate that it has established overall plans for all further activities in the system lifecycle which fulfil the requirements identified at the previous milestone.

In addition to planning what should actually be done, the plans should also contain details about how Configuration management and Quality assurance and control should be performed. Another aspect is structural quality, i.e. whether the plans are easily Comprehensible and that the Quality of the work schedule implies that the set of tasks is possible to fulfil. A particular quality attribute is traceability, i.e. how the different elements of the development can be traced back to the plans.

The observations of the 'observable' nodes in this diagram will to a large degree be based on judgement. However, by breaking them down into more detailed sub-observations, one can get a better basis for a judgement which is more “objective”.

Figure 5 is the corresponding Influence net.

<u>*Questions about Completeness of plans*</u>

Comments:

One aspect to investigate about these plans is their completeness, i.e. whether all phases in the development are well planned. This means that all tasks required in each of the planned lifecycle phases given by any standards or guidelines used by the Company. The plans should also be explain how the safety integrity is preserved through all lifecycle phases.

In addition to planning what should actually be done, the plans should also contain details about how configuration management and quality assurance and control shall be performed.

Careful planning of all future phases in the safety system's lifecycle and clear evidence that the process has been followed is an important contributor to an adequate safety demonstration. "Project planning should be documented in a comprehensive and specific plan for the software-based safety system, or as a set of plans that cover all aspects of the project…… Modifications of the system should not be excluded, and provisions should be made for possible further iterations of the safety analysis." (reference 3, section 4.1)

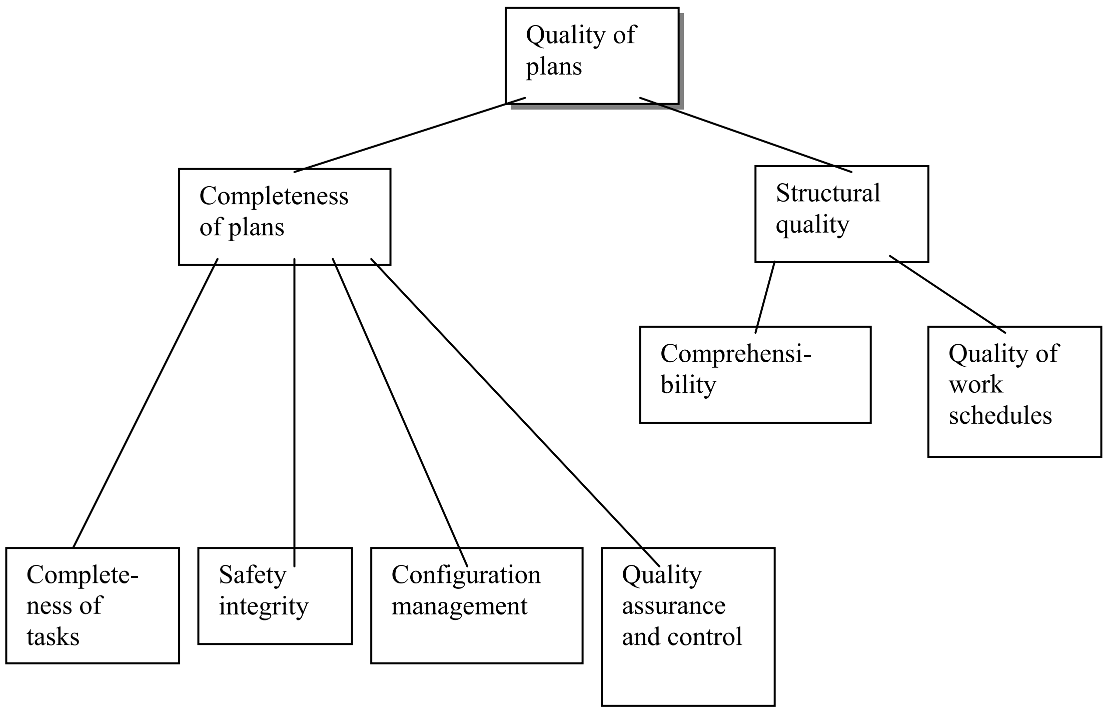


Figure 5 Influence net for milestone *Project planning*

Other aspects of the project which must be planned are quality assurance, verification and validation, configuration management, and commissioning and installation. Further guidance on planning, specifically directed at NPPs, is given in IEC 61513, section 5.4, "Overall planning" and section 6.2, "System planning".

Q9.0 Is the system planning adequate?

A9.0.1 The preliminary *Safety Review* should demonstrate that adequate plans have been drafted. The final *Safety Review* should provide evidence that they have been followed. The adequacy of the plans should be judged against the responses to the following questions.

A9.0.2 Figure 9.1 shows an outline of a typical process for the development of a software-based system showing the relationship of verification, validation and commissioning to the various phases. The future phases that are being discussed in the remainder of this document start at the Software-based System Requirements phase that equates to *Requirements specifications* of Chapter 8. Other lifecycles are possible such as those involving prototyping; nonetheless any life-cycle should include the basic processes shown in figure 9. For instance, any software developed for a prototype could be considered as pre-existing software if it is incorporated into the final system. Also, the software produced by any code-generating, graphical tool which is used in the safety system should be considered to be pre-existing software, and should be subjected to the full requirements of these guidelines.

A9.0.3 The system planning should be shown to meet the requirements of IEC 61513, section 6.2, "System planning" (reference 9).

Q9.1 Are all future phases in the system lifecycle included in the plans, including:

- *Q9.1.1 System realisation?*

A9.1.1.1 The system realisation plans should cover at least the following phases of figure 9.1 even if the particular lifecycle model shown is not followed:

software-based system design;
software requirements;
software design;
software implementation;
software-based system integration.

A9.1.1.2 There should also be a Verification plan (usually included in a V&V Plan) for performing the *verification* and *system integration verification* activities. The techniques to be used to verify the software should be stated in the Verification plan. Explicit procedures for these techniques should be identified and their suitability for the safety systems should be justified. It is expected that they will include a combination of techniques including both static examination of documents and dynamic execution of the implementation. The personnel performing the verification activities should be identified in the plan, and for safety systems they should be independent from the designers of the system. (see reference 3, sections 4.15 and 4.17)

- *Q9.1.2 System validation?*

A9.1.2.1 System validation demonstrates that the software-base safety system has achieved its overall safety and functional requirements. There are two distinct steps of validation (see figure 9.1). The first step is the validation of the Software-based Safety System Requirements against the Concepts, Overall Requirements and Hazard & Risk Analysis. The second step is the validation of the Software-based Safety System implementation against the Software-based Safety System Requirements. Techniques and explicit validation procedures should be identified in the System validation plan. The personnel performing the System Validation should be identified in the plan. (see reference 3, sections 4.13 and 4.17)

- *Q9.1.3 Installation and commissioning?*

A9.1.3.1 After the computer parts of the safety system has been constructed and validated as stand-alone systems, it is integrated with safety actuation systems & safety system support features and tested within the real plant environment. This process of installation and commissioning must be carefully planned to co-ordinate the proper transition from development to use, and the hand over from the developers and verifiers to the users and maintainers. (reference 3, section 4.25)

A9.1.3.2 The *installation and commissioning plan* should cover (see reference 3, section 4.26):

the activities for the integration of the computer aspects of the safety system into the plant for all modes of operation (the interfaces between: the 'Other Components' and the new and/or existing plant systems, and the tests needed to check the proper functioning of each interface should be identified);

the production and execution of the commissioning tests needed to confirm proper functioning of the safety system in the plant environment for all modes of operation (the records and reports that will be generated to describe the results of commissioning should be identified);

the training of the users and maintainers;

the transfer of the configuration management and change control process from developers to maintainers:

the personnel performing the commissioning;

the required interactions with the SKI, including any approvals or hold-points that may be required before the safety system can be put into operation.

A9.1.3.3 See reference 9, IEC 61513, section 5.4.3, "Overall integration and commissioning plans" for additional requirements.

- *Q9.1.4 Operation and maintenance?*

A9.1.4.1 The operation phase of the software-based safety system follows installation, commissioning and any regulatory approval for use. The system becomes part of the nuclear power plant and is operated by the licensee. The operation phase of a particular system continues until it is removed or replaced (possibly by a modified version). Operation of the software-based safety system will involve some maintenance activities to keep the equipment in good condition, and to fix any failed components. It should be noted that the processes that supports the safety systems of nuclear installations during their operational use have the potential to jeopardise that safety system's fitness for purpose if they are not of a sufficiently high integrity. Therefore it is important that they are well planned and controlled. (reference 3, sections 14.1 and 14.2)

A9.1.4.2 The Operation and Maintenance Plans should cover the planning of:

a probationary period;
the security arrangements;
start-up arrangements for the various modes of operation;
constraints on the operation of the safety system;
actions to be taken in the event of failure of the safety system;
operational and calibration data handling;
periodic testing;
software maintenance arrangements;
arrangements for recording operational experience.

A9.1.4.3 See reference 9, IEC 61513, section 5.4.4, "Overall operation plan" and section 5.4.5 "Overall maintenance plan" for additional requirements.

Q9.1.5 Security Plans?

A9.5.1 SKIFS 1998:1, Chapter 2, paragraph 5 requires that a "physical protection plan" be documented. It also requires that this plan be subjected to a *Safety Review*. The requirements of SKIFS should be met through an 'Overall security plan' and a 'System security plan' for the safety system, and this should only be *noted* in the preliminary *Safety Review*. The details of this security plan should be covered by a security classification that restricts the personnel who have access to it to those with the required security clearance and authority. Such information should only be issued on a "need-to-know" basis.

A9.5.2 The 'Overall security plan' should meet the requirements of IEC 61513 (reference 9), section 5.4.2 and the 'System security plan' should meet the requirements of section 6.2.2.

A9.5.3 Responses A11.16.2 and A11.16.2.1 provide more detail on the security requirements.

Q9.2 Is the safety integrity preserved through all lifecycle phases?

A9.2.1 The preservation of the safety integrity through all lifecycle phases is achieved through the correct application of verification, validation and commissioning. Verification checks a product against the output of the preceding phase; validation checks a product against higher level requirements and goals; and commissioning checks that the plant (including the installed system) performs to the overall requirement. There should also be an adequate change control process (see response to question Q9.4.4)

A9.2.2 The documented V&V and Commissioning results provide evidence that all planned activities have been performed. These documents should include a mechanism for recording all non-compliances found during the analysis with their proper resolution via the change control process. The records must also demonstrate the traceability of all V&V and commissioning work to the relevant source documentation. This is part of the safety demonstration that safety integrity is preserved.

A9.2.3 To maximise the preservation of safety integrity, the personnel performing V&V and commissioning should be independent from the personnel performing the safety system's realisation. This independence should be in terms of (see reference 3, section 4.17):

> "*technical independence* - done by different people, preferably using different techniques and tools;
>
> *management independence* - led and motivated by different people. The validation team and the development team should have different management lines. Official communication between independent teams should be recorded;
>
> *financial independence* - having a separate budget with restrictions on transfer of financial resources between development and V&V."

A9.2.4 The application of an adequate quality system will ensure that the functional behaviour is traceable throughout the system lifecycle and that the behaviour has been adequately demonstrated through testing to the required standards. An adequate quality system will also ensure that the V&V and commissioning are properly managed and recorded.

Questions about Quality assurance

Comments

Proper plans for conducting software quality assurance should be established in order to provide for easier approval of software based products. The quality assurance policy of a company should at least contain the following:

An overall quality assurance policy document

Qualification routines for software based safety related systems

A detailed quality assurance document for managing software projects.

The quality assurance plan should state clearly to which degree it conforms to any relevant international standard or guideline

Only well-reputed vendors should be chosen for the system development. To qualify a vendor, it should be mandatory that it is able to supply relevant information on previous delivery projects within the same kind of environment.

An external QA control should be performed by an independent team, i.e. persons not directly involved in, and with no direct responsibility for the system development.

To quote from reference 8, section 1.6.1: "It is widely accepted that the quality of software cannot be guaranteed by inspection or by testing of the product alone. In general, the application of a good Quality Assurance system provides essential evidence of the quality of the end product. Software is no different: it is felt that the final software-based system will have a higher degree of *dependability* if the techniques of Quality Assurance are used. This is particularly true for any non-trivial software product since it cannot be exhaustively tested and interpolation between test inputs is not possible due to the digital nature of the system (a situation that does not pertain with analogue systems)."

Additionally for safety systems, the quality system should cover all aspects of the system life cycle.

The licensee's quality assurance programme should be available for regulatory review (and possibly approval) before the project begins. This should be in line with the IAEA's "Quality Assurance for Safety in Nuclear Power Plants and other Nuclear Installations", Code and Safety Guides Q1-Q14, Safety Series No. 50-C/SG-Q, IAEA, Vienna (1984).

A software quality plan should be produced at the outset of the project. This plan should cover any external suppliers and should at least include (as indicated in reference 3, section 4.11, "Quality assurance plan"):

> "identification of the hardware and software items to which the QA programme description applies, and of governing standards, procedures and tools to be used on the project;
>
> who must review and approve each document to be produced for official release;
>
> a description of the project organisational structure, including assurance of the independence of QA auditors;
>
> a description of competence and training requirements for personnel involved in the project;
>
> a mechanism for identifying, reporting and disposing of non-conformance to standards and procedures;
>
> identification of all necessary plans: e.g. Development plan, Verification plan, Configuration management plan, System validation plan, Commissioning & installation plan, and Operation & maintenance plan;
>
> an indication of the number and scope of QA audits;
>
> the procedures for qualifying the tools;
>
> a mechanism for checking the quality of components from external suppliers. If this mechanism relies on external certification procedures (e.g. type testing) the description of these procedures should also be identified."

Q9.3 Do overall quality assurance plans exist for all phases with:

- *Q9.3.1 QA documentation for managing software projects?*

A9.3.1.1 Some guidance is given in the Technical Report series No. 282, "Manual on quality assurance for computer software related to the safety of nuclear power plant", Vienna 1988.

A9.3.1.2 In judging the acceptability of the QA documentation, the QA plan should cover the requirements of reference 3, section 4.11 quoted above and there should be a compliance statement should be obtained from the licensee against either:

i) IEEE Std 730.1-1989, "Standard for Software Quality Assurance Plans";

or

ii) ISO 9001

and

ISO 9000-3 "Guidelines for the application of ISO 9001 to the development, supply and maintenance of software".

- *Q9.3.2 QA in project management?*

A9.3.2.1 SKIFS 1998:1 indicates that the quality system should follow IAEA Safety Series No. 50-C/SG-Q, Vienna 1996. The licensee must provide a compliance statement, as part of the *Safety Review*, which shows that the requirements of the IAEA guide have been met. The statement should be supported by an audit report.

- *Q9.3.3 Qualification routines for software systems?*

A9.3.3.1 For each software system there should be tests that are systematically defined and documented to demonstrate that adequate test coverage has been achieved. There should also be evidence that the tests are traceable to the *system requirement specification*s. Where type testing is offered as evidence of adequacy of a software item, there should be sufficient documentation available to enable a review to be performed to the level of the requirements of this guide, i.e. there should be an adequate safety demonstration. A summary report (with appropriate referencing) should be provided in the *Safety Review*.

- *Q9.3.4 Qualification assessment of vendors?*

A9.3.4.1 "Well structured and effective organisations, i.e. licensees, their subcontractors and suppliers, are regarded as necessary in the management of all aspects of the safety system lifecycle to reduce the *faults* introduced into a system, and to ensure that those *faults* which are found are handled properly. Vendors must demonstrate that, throughout the lifecycle, they will maintain clear delineation of responsibilities and effective lines of communication plus proper recording of decisions will minimise the risk of incorrect safety system behaviour. An additional, important aspect of an effective management system is the development of a safety culture which at all levels within the organisation, emphasises safety within the organisation. By the use of managerial supervisory and individual practices and constraints, the safety culture sustains attention to safety through an awareness of the risk posed by the plant and on the potential consequences of incorrect actions." (from reference 8, section 1.5.1) The *Safety Review* should demonstrate that this has been achieved. In particular, the *Safety Review* should provide evidence that the requirements of A9.3.4.4 to A9.3.4.7 have been met.

A9.3.4.2 Each vendor should have a demonstrably good track record in the appropriate field. Such companies should only use staff with the appropriate qualifications and training for the activities in which they are engaged, and they should demonstrate that they have sufficient resources available. The *Safety Review* should provide written evidence that this is the case. The vendor's QA arrangements should be to the level of this section, including the use of an adequate configuration management system

A9.3.4.3 For software-based safety systems, the vendors and subcontractors should, therefore, each provide evidence of the following (which should be summarised in the *Safety Review*, together with appropriate referencing)- see reference 8, section 1.5.3.2:

> a written safety policy, demonstrating a commitment to a safety culture which enhances and supports the safety actions and interactions of all managers, personnel and organisations involved in the safety activities relating to the production of the safety system;
>
> a written statement of the responsibilities of, and relationships between, all staff and organisations involved in all aspects of the safety system - this to be clearly understood by all concerned;
>
> adequate levels of staffing (demonstrated, for example by means of appropriate work planning techniques) to ensure that the safety system's fitness for purpose is not put in jeopardy through lack of staff effort. There should be an appropriate balance between the numbers of hired staff and full time employees;
>
> the appropriateness of the level of training and experience of the staff for the tasks they are undertaking within the safety system lifecycle. Suitable evidence should be available for inspection.

A9.3.4.4 Adequate procedures should be in place for controlling contract documentation between customers and vendors to ensure that all specification and safety-critical contractual requirements are properly controlled. (reference 8, section 1.5.4.1)

A9.3.4.5 Vendors should have appropriate security arrangements in place.

A9.3.4.6 The vendor of tools should have adequate procedures for maintaining the tool and update the experience feedback. They should be committed to informing users of all anomalies discovered post delivery of the tool. (see reference 8, section 1.4.3.6)

A9.3.4.7 The licensee should also ensure that the computer/software vendors are prepared to release all proprietary information that would be needed for the licensing process. (see reference 3, section 2.14)

Questions about Configuration management plans

Comments

The Configuration Management plan should document the method to be used for identifying software product items, controlling and implementing changes, and recording and reporting change implementation status. The plan should be applied to the entire software life cycle and is especially important in environments where software could impact on safety measures.

The Configuration Management plan should be used both on software and the associated documentation to be sure that both are being updated in parallel and coincide at any time.

The decision about which items (documents, software etc.) are to be placed under configuration control must be decided at an early stage within the project. Changes to these items have to be made according to strict procedures.

Documents and software being placed under configuration control must be stored in a safe place. By a safe place one means for instance that backups are taken, diverse backups stored at different sites, security measures are implemented, and that correct version labelling is followed.

The configuration management organisation needs to keep track, not only of the specific produced software, but also of the environmental software products. Complete revision control procedures stating how to cope with new releases of the different software products must be established and strictly handled.

To quote reference 8, section 2.7.1: "A software-based safety system is formed from many different items of software and hardware and includes many documents that describe these items. Hence it is important that there is a full index of the items involved in the construction of a system, and that the status of each item (including changes made) is established and tracked so that faults are not introduced due to the incorrect versions of these items being used. A well ordered configuration management system provides the means to ensure this, as well as playing a significant role in a properly managed change control process."

Hence, all the items of software including documents, data (and its structures) and support software should be covered by a suitable, readily understood and fully documented configuration management system (CMS) throughout the lifecycle.

Q9.4 Are there acceptable configuration management plans for all phases with:

A9.4 As indicated in the response to question Q5.11 written procedures should be available which for instance comply with the following standards:

IEEE 1042-1987, "Guide to Software Configuration Management".

IEEE Std 828-1990, "Software Configuration Management Plans".

The use of these procedures should be demonstrated by means of a suitable audit. *The Safety Review* should contain a summary report of the compliance with these standards together with a review of the audit. Reference should be made to the source documentation. Specifically, the *Safety Review* should adequately address the points raised in A9.4.1.1 to A9.4.2.4.

- *Q9.4.1 Proper organisational structure?*

A9.4.1.1 The organisational structure should include a software librarian who controls the issuing of software versions together with the associated documentation. This should also cover the issuing of all project documentation and data. An item of software or documentation should not be accessible (to persons other than those responsible for its design and verification) until it is approved and under configuration control. (see reference 8, section 2.7.3.4.1)

A9.4.1.2 There should be a library or storage area designated to hold all items that are under configuration control so that it is easy to find and retrieve any identified item in any of its existing versions. There should be a procedure describing when and how a developed or acquired item will be placed under configuration control. There should be a mechanism so that at particular points on the schedule, a set of configuration controlled items can be identified which represent the 'baseline' for subsequent work. Each item should have a record containing relevant information such as when it was complete, what changes have been incorporated since the previous version, its current approval status, and the persons responsible for creating, reviewing and approving it. There should be approval procedures, tied to the quality assurance plan. (from reference 3, section 4.20)

A9.4.1.3 At any time, a QA or regulatory auditor should be able to request and directly receive any items of the set that constitute the most complete and current description of the system and project (the current baseline). This set of items should be identical to the ones currently being used as a basis for on-going development or analysis, with the exception of work in progress that has not yet been approved or released. (from reference 3, section 4.22)

A9.4.1.4 Program modifications should be controlled by change control procedures in which modifications are evaluated, approved and properly documented. Access to the software should be strictly controlled so that a programmer cannot make an unauthorised change at any stage in the lifecycle (especially post V&V). (reference 12, section 40)

A9.4.1.5 After delivery of the software and its documentation, the same level of configuration management should be maintained at the site where the delivered software is stored. In particular, a configuration audit should be performed on the safety system software prior to loading to establish that the correct items and versions have been included in the system. Following system loading, the loaded version should be verified as being uncorrupted. (reference 8, sections 2.7.3.4.4 and 2.7.3.4.5)

Q9.4.2Proper procedure for identification of items?

A9.4.2.1 The procedure for the identification of items should ensure that all software copies, including any pre-existing software that is being used, should be clearly and uniquely labelled with at least title, version number and creation or acquisition date. Provision should be made for the inclusion in the source code listing of information on changes made, approval status, and authors, reviewers and approvers names. The identification and version number should be included in the code so that this can be checked by other software items. (see reference 8, section 2.7.3.4.3)

A9.4.2.2 To aid backward traceability, the configuration management system should maintain lists of items, and their changes that were included in previous baselines. These lists should include names of individuals, and organisations to whom configuration items have been distributed. (see reference 8, section 2.7.4.4)

A9.4.2.3 Appropriate links are required between the software and the hardware configuration management procedures and databases. (see reference 3, section 4.21)

A9.4.2.4 All items of software development such as compilers, development tools, configuration files and operating systems should be under configuration management control. All identifiable items, such as documents, or components of the software, or data structures, should be uniquely identified including a version number. This should include both developed items and existing items that are being re-used or re-applied. (see reference 3, section 4.20)

- *Q9.4.3 Acceptable procedure for secure storage?*

The secure storage should be included in the Security Plans of Q9.1.5 - see also A11.16.2. and A11.16.2.1.

A9.4.3.1 As part of the need to maintain strict configuration control of the computer system, the software-based safety system supplier should identify how intentional or inadvertent corruption of the computer system functionality (e.g. unauthorised access, unauthorised code, viruses) will be prevented. This should include procedural or other controls on how changes to the system are made and verified, and how unauthorised changes are prevented. There should be an analysis of the threats to security together with a justification of the level of security implemented. These procedures or controls should be summarised, with appropriate referencing, in a separate safety (security) review document as required by SKIFS 1998:1, Chapter 2, para 5. The security arrangements should be shown to meet the requirements of A9.4.3.2 to A9.4.3.5. (see reference 3, section 6.38)

A9.4.3.2 The storage system should be protected by a security system that prevents unauthorised access. Users of the CMS should only be able to access parts of the CMS for which they have authority. The security system should be designed to a high standard and be regularly reviewed for possible breaches. The storage system should not be linked to an off-site network unless it is via an adequate "firewall" that has been demonstrated to be secure against all perceived threats. (see reference 12, section 37)

A9.4.3.3 During all stages of the software lifecycle software should be stored in a secure place and the media clearly identified as to its status. Strict procedures should be employed to ensure that the correct version is issued and that it has not been subjected to unauthorised modification either accidentally or intentionally. The issued version should be verified against a master copy. (reference 12, section 38)

A9.4.3.4 Following delivery to site, suitable arrangements should be made for the secure storage of copies of the software that is to be loaded in the event of a system failure. Due regard should be given to all possible means of corruption of the software. Particular attention should be paid to the secure storage of pre-programmed memory devices if they are used. Means should be provided to verify that the software has not been corrupted prior to its use both before and after loading. Automatic means are preferred. (see reference 12, section 101)

A9.4.3.5 The configuration management system's security arrangements should meet the requirements of British Standards Institution "Code of Practice for Information Security Management", BS 7799 (1995) or equivalent.

- *Q9.4.4 Change control procedures acceptable?*

A9.4.4.1 All modifications to a software-based safety system are subject to the requirements of SKIFS 1998:1, chapter 4, paragraph 6, "Modifications". In particular, there should be a review of the Safety Analysis. The Safety review should provide a summary description, with appropriate referencing, of the software change process procedures. The software change control procedures should be shown to comply at least with the requirements of A9.4.4.3 to A9.4.4.10.

A9.4.4.2 To quote from reference 8, Chapter 2.7: "Changes to the software of *safety systems* used in nuclear plant have the potential to affect significantly the safety of that plant if incorrectly conceived or implemented. Any alteration to the software of these systems at any phase in the system lifecycle, whether due to an enhancement or a need to adapt to a changing environment (resulting in a modification to the system requirements) or to correct an error, must be conceived and implemented at least to the same standards as the original implementation. This point is particularly important in the case of changes made after delivery. Furthermore, there should be procedures in place to ensure that the effects of such changes, on all parts of the system, are assessed to reduce to acceptable levels the potential for faults to be introduced.

"A software-based safety system is formed from many different items of software and hardware and includes many documents that describe these items. Hence it is important that there is a full index of the items involved in the construction of a system, and that the status of each item (including changes made) is established and tracked so that faults are not introduced due to the incorrect versions of these items being used. A well ordered configuration management system provides the means to ensure this, as well as playing a significant role in a properly managed change control process."

A9.4.4.3 Once an item has been placed under configuration control, it should not be changed except according to a well defined procedure that includes impact analysis, and that procedure will result in a new version of the item rather than a replacement of the existing identified item. The change control procedure should maintain records of the problems that were identified to necessitate the changes, how the problems were analysed, what items were affected, what specific changes were made to fix the problem, and what versions and/or baseline were produced to resolve the problems. The change control procedure may also identify responsibilities for approving changes if these are in addition to, or different from, the basic item approval mechanism. In general, a change should involve a repetition of all processes used to originally create the item, including all analyses, starting from the highest level item affected. A regression analysis should be used to identify the tests required to maintain the V&V record. The regression analysis procedures and results should be documented. (reference 3, section 4.23)

A9.4.4.4 After completion of the development (and delivery to site) a different change process may be applied since the organisation and people responsible for maintenance may be different from the original developers. This is discussed further in Chapter 13 "Maintenance and Modification Procedures". Prior to the application of the maintenance change process, a change procedure should be documented and it should consider the need to repeat the analyses undertaken during development. Any decisions not to repeat any of the analyses should be documented and justified. (see reference 3, section 4.24)

A9.4.4.5 The software change procedure and documentary process should apply to all elements of the software system including its documentation. This procedure should apply equally to system functionality enhancements, environmental adaptations (resulting in a modification to the system requirements) and corrections of implementation errors.

A9.4.4.6 No distinction should be drawn between major and minor software changes since a wrongly implemented minor change could challenge safety. (see reference 8, section 2.7.3.1.2)

A9.4.4.7 As specified in reference 8, section 2.7.3.1.5, the software change procedures should contain the following basic elements:

- the identification and documentation of the need for the change;

- the analysis and evaluation of the change request, including: a description of the design solution and its technical feasibility; and its effect on the safety of the plant and on the software itself;

- the impact analysis of each software change: for each software change, the implementors of the change should produce a software impact analysis containing a short description of the change plus a list of software parts affected by the change plus the effect on non-functional system properties as, e.g. *dependability*, response time, system accuracy, hardware performance. Objective evidence that the full impact of each software change has been considered by the implementor for each software part affected should be provided. As a minimum this should consist of a summary description of the change to be implemented, plus documentary evidence of the effect of the change on other software parts. The software impact analysis should also list data items (with their locations - scope of the data item) that are affected by the change plus any new items introduced.

- the implementation (consistent with the standards employed in the original production process), verification, validation (as appropriate) and release of the changed software or document item. The V&V phases may make use of the software impact analysis to perform regression testing.

A9.4.4.8 Faults should be analysed for cause and lack of earlier detection. Any generic concern should be rectified and a report produced. (reference 8, section 2.7.3.1.6)

A9.4.4.9 A correction to a wrongly implemented software change, if found at the stage of site testing (i.e. following installation of the software on the plant), should be processed as though it were a new software change proposal. (reference 8, section 2.7.2.3.1)

A9.4.4.10 The *commissioning* team should use the software impact analysis and the factory V&V report to develop their own series of tests in the form of a site *commissioning* test schedule. (reference 8, section 2.7.3.2.2)

- *Q9.4.5 Good revision control?*

A9.4.5.1 Procedures should be set up for version control and the issuing of correct versions. Provision should be made for informing all relevant personnel of pending changes and approved modifications. (see reference 8, section 2.7.3.4.2) The revision control should be to the standards set down in the response to all parts of question Q9.4.

<u>*Questions about Structural quality*</u>

Comments:
Aspect of the structural quality of the plans is whether they are easily comprehensible and that a work schedule is made which shows that the set of tasks is possible to fulfil.
A particular quality attribute is traceability, i.e. how the different elements of the development can be traced back to the plans

Q9.5 Is comprehensibility of the plans acceptable?

A9.5.1 It should be possible for the licensee to explain the plans to the assessor, showing how all the following requirements are met.

- *Q9.5.1 Are the plans well structured?*

A9.5.1.1 A proven, proprietary, computer-based project management tool should be used for a safety system project. This will ensure that the plans are well structured.

- *Q9.5.2 Is the language understandable?*

A9.5.2.1 A proven, proprietary, computer-based project management will ensure that the language is understandable.

- *Q9.5.3 Is the terminology consistent?*

A9.5.3.1 A proven, proprietary, computer-based project management tool will ensure that the terminology is consistent.

Q9.6 Concerning the work schedule:

- *Q9.6.1 Are all tasks included?*

A9.6.1.1 A project review by the licensee together with a QA audit should provide sufficient evidence that all tasks are included. The regulator should sample the work schedule, the project review and the QA audit.

- *Q9.6.2 Is the timing realistic and consistent?*

A9.6.2.1 This requires specialist knowledge of the vendors staff productivity. The licensee should provide evidence in support of the effort and duration of each activity of the work schedule in the preliminary *Safety Review*. This should be based on the analysis of past projects of a similar nature from which productivity figures for designer, programmer etc. can be derived. Estimates of the work content of each activity are also required - again based on earlier projects.

- *Q9.6.3 Are milestones identified?*

A9.6.3.1 The milestones that need to be identified will relate to the particular lifecycle model chosen for the development. The licensee should justify, in the preliminary *Safety Review*, the milestones chosen in the work schedule with reference to the particular lifecycle model being used. As a guide the lifecycle model of Figure 9.1 shows a typical series of phases and activities across these phases. A milestone should be present for the end of each of the phases and for each of the separate verification, validation and commissioning phases. In addition milestones should be present for the completion of the following:

reviews of impact on Safety Analysis, Tech Specs, Maintenance, Surveillance & Testing schedules, and Operating procedures;

project reviews;

QA audits;

licensing activities (e.g. submissions to SKI such as Safety Reports & Safety Reviews);

security arrangements;

staff training;

review of quality system and configuration management system.

Q9.7 Can all items in the plan be traced to corresponding items in the realisations of the plans?

A9.7.1 The licensee should provide, in the preliminary *Safety Review*, a cross-reference between the items in the plan and the corresponding items in the realisations of the plan (these will depend on the life-cycle model chosen but should contain the elements of figure 6). This should be structured such that any omissions will be clearly visible.

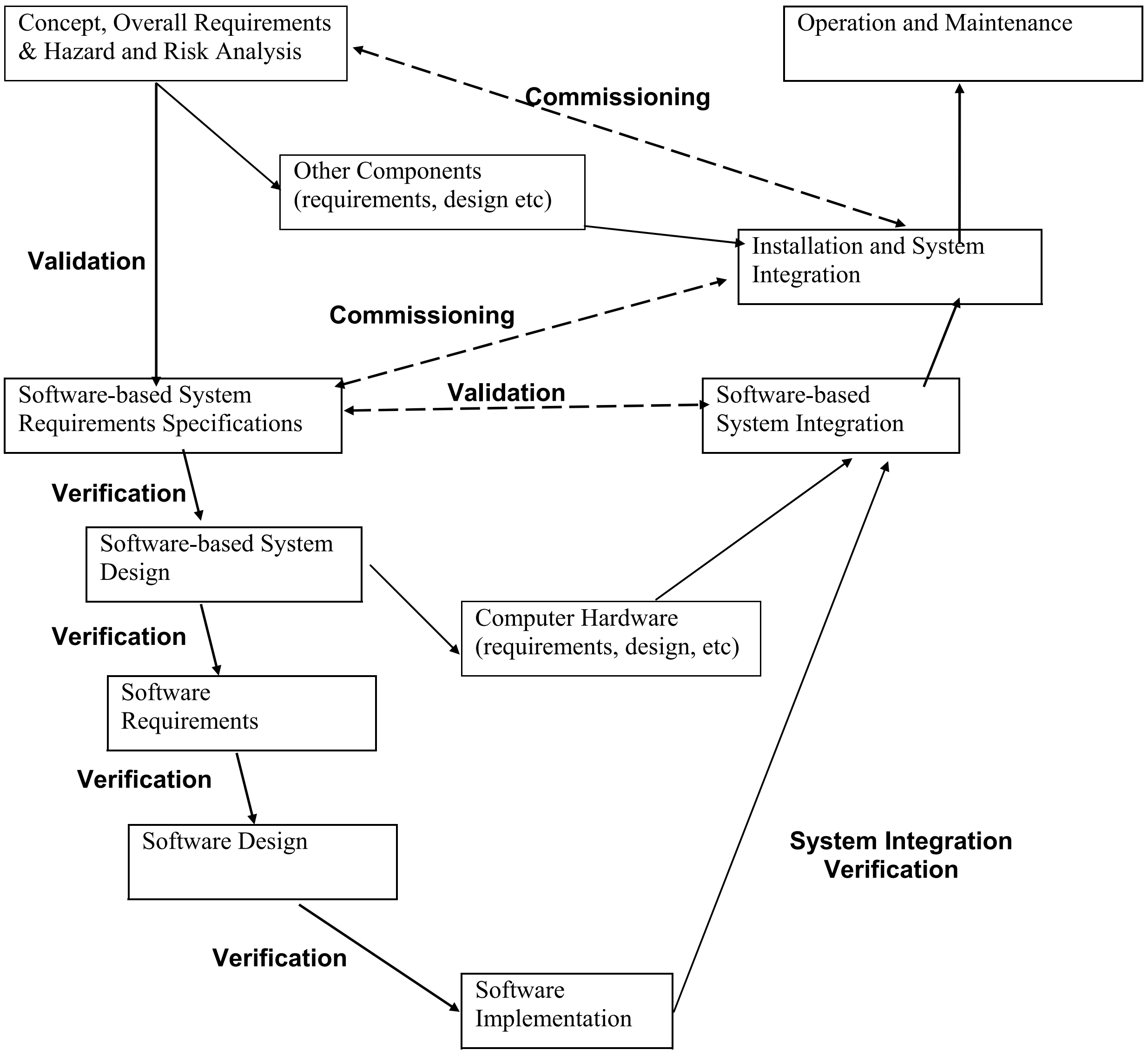

Concept, Overall Requirements & Hazard and Risk Analysis
Operation and Maintenance
Commissioning
Other Components (requirements, design etc)
Validation
Installation and System Integration
Commissioning
Software-based System Requirements Specifications
Validation
Software-based System Integration
Verification
Software-based System Design
Verification
Computer Hardware (requirements, design, etc)
Software Requirements
Verification
Software Design
System Integration Verification
Verification
Software Implementation

FIG. 6 Typical lifecycle model showing how verification, validation and commissioning relate to the project phases.

10 Overall System Design

At this milestone the company shall document how the system shall be made to fulfil the requirements. This means a completeness of the design with respect to the functional requirements and fulfilment of safety requirements. An additional aspect to observe is the design methodology. This is illustrated in Figure 7a - c.

<u>*Questions about Completeness of design*</u>

Comments:
The completeness of the design means that all tasks specified in the requirement should be included in the design document.
It also means that all relevant aspects of the system to be designed are considered in the design document. IEC-61508, part 3, sect. 7.4.2.2 identifies such aspects are functionality, information flow between components, sequencing and time related information, timing constraints, concurrency, and data structures and their properties

Q10.1 Are all tasks specified in the requirements also included in the design?

> *It is possible that not all the tasks specified in the requirements will be implemented via software - some functional and non-functional requirements might be achieved through hardware. This has to be borne in mind when tracing requirements through the lifecycle. Infact. the adequacy of the non-software-based designs should be subjected to an equally searching assessment.*

A10.1 The *Safety Review* should contain a description of the design (with appropriate referencing to the detailed design documents) plus a *traceability matrix* or a similar representation showing which parts of the software and hardware design meet the tasks specified in the requirements. All the tasks should be covered by the design. Here tasks are defined as the operations that the operators and maintainers must perform. There should also be a demonstration of the correctness of this *traceability matrix* through verification (see response to question 12.5 on manual inspections in Chapter 12).

Q10.2 Does the design document reflect the intended functionality of the system?

A10.2 The *Safety Review* should contain a *traceability matrix* which confirms that all the functions in the requirements specification are met by specific aspects of the design, i.e. there is an identifiable aspect of the design which can be linked to a particular function. The verification and validation activities will provide auditable evidence that the design meets the functional requirements.

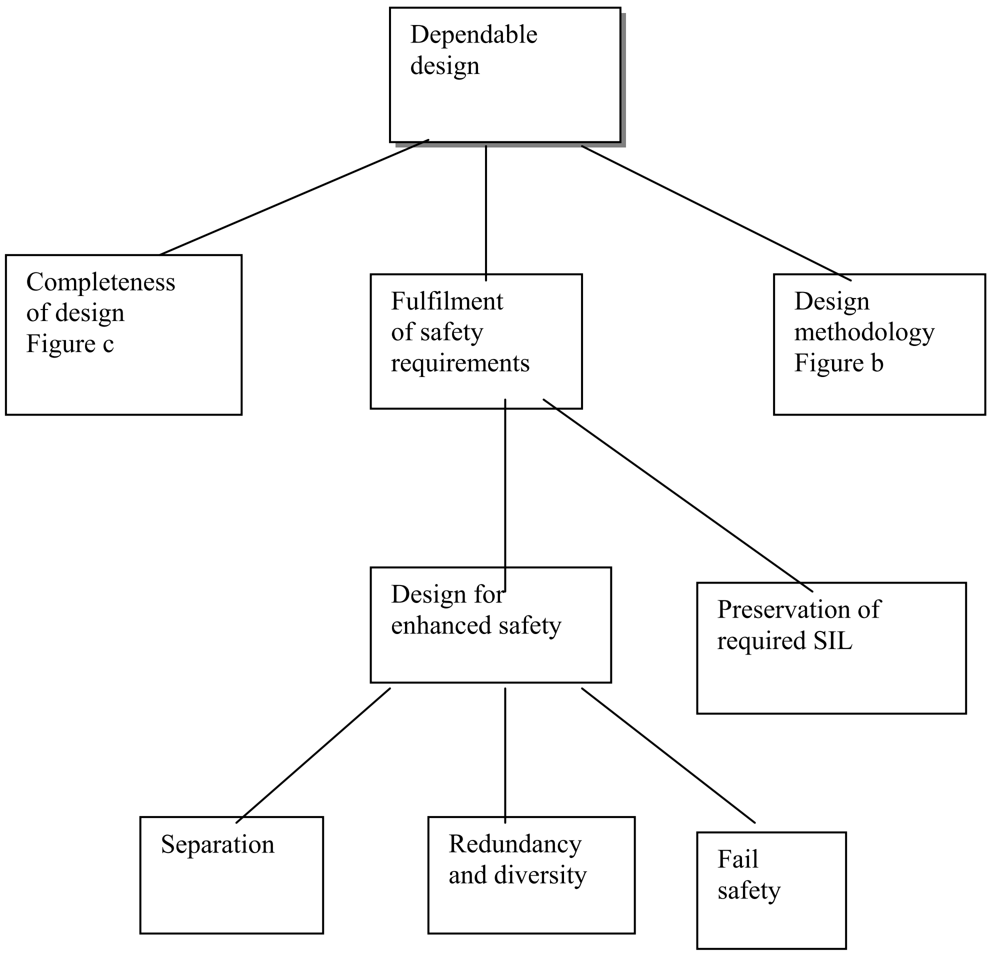

Dependable design
Completeness of design Figure c
Fulfilment of safety requirements
Design methodology Figure b
Design for enhanced safety
Preservation of required SIL
Separation
Redundancy and diversity
Fail safety

Figure 7 a Influence net for milestone *Overall system design*

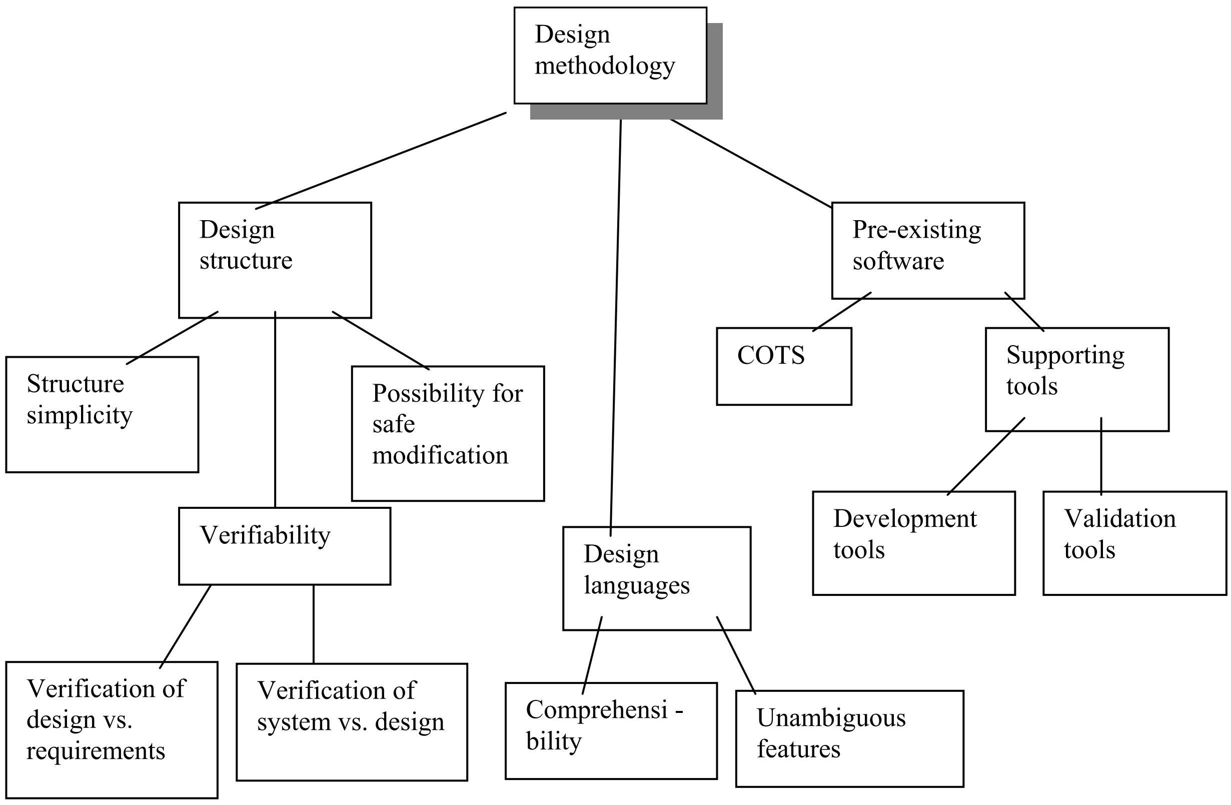


Figure 7b Influence net for milestone *Overall system design (cont.)*

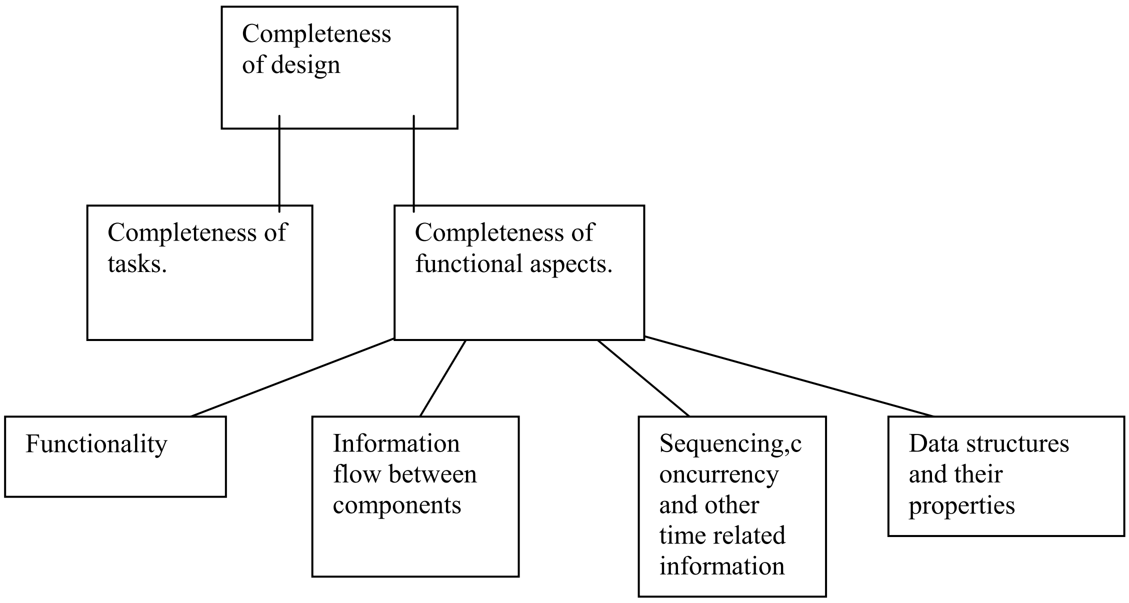

Completeness of design
Completeness of tasks.
Completeness of functional aspects.
Functionality
Information flow between components
Sequencing,c oncurrency and other time related information
Data structures and their properties

Figure 7c Influence net for milestone *Overall system design (cont.)*

Q10.3 Does the design adequately describe the information flow between components?

A10.3 A standard technique should be used to describe the information flow between components. The approach adopted should be described in sufficient detail in the *Safety Review* to enable the design to be understood. Techniques such as data flow diagrams (part of the Structured Analysis and Structured Design approach promoted by Edward Yourdon - see "Modern Structured Analysis", Yourdon E, & Edward J, Printice Hall, 1989) that show the data that is held by, and transferred between the software components of the design. This is a top-down process that progresses through levels of refinement to create a more detailed picture as the components are further decomposed. Other techniques for describing the data flow are permitted but should be shown to have the same clarity of presentation as data flow diagrams.

Q10.4 Does the design address sequencing, concurrency and time related information?

A10.4.1 The *Safety Review* should describe how the real-time performance requirements and any constraints have been met, and through a *traceablity matrix* show the links to the individual non-functional requirements (again "address" is taken not to include evidence that the requirements have been correctly implemented - that is a V&V activity). For a safety system, each of the trip functions (e.g. Pressuriser Water Level High for a PWR) and safety actuation system sequencing (e.g. Auxiliary Feedwater Pumps Start, again for a PWR) described in the functional requirements should be shown to have a link to a particular design aspect. Cycle times, signal sampling and processing functions should be shown to have been considered and that overall response times meet the specified timing budgets. The frequency of sampling of input parameters, and the required digital filtering (if applicable) should be included. It should be demonstrated that proper consideration has been given to ensuring that varying measurements are truly represented within the system, and events are observed within sufficient time to take the necessary actions.

A10.4.2 The design's sequencing and concurrency aspects should be supported by a suitable model. A summary description of the model chosen should be provided in *the Safety Review*, together with a justification of the suitability of the modelling method. Yourdon's state transition diagrams (a form of finite state machine) are useful in this context as are Timed Petri Nets. Even the standard flowcharts can be used to good effect. Whichever method is used, the traceability matrix should show how these link to the functional requirements.

Q10.5 Does the design adequately describe the data structures and their properties?

The paper "Data for software systems important to safety" by D Welbourne and N P Bester, GEC Journal of research, Vol 12, No 1, 1995 provides some useful insights.

A10.5.1 The *Safety Review* should describe the data structures and their naming conventions, and demonstrate that they have been used uniformly throughout the whole system. Each data structure identifier should consistently reflect its type (array, stack, linked lists, variable, constant, etc.), its scope (local, shared, external), its kind (input, output, internal etc.). A data dictionary should be developed which ensures that there is consistent use of each data object. (see reference 3, section 9.21)

A10.5.2 It should be shown that each input and output type such as pressure measurement, valve position etc have been included. And that all other related information associated with the data item is present, e.g. for an input - building location, equipment room, cubicle, rack number, terminal number, input address, engineering units, range, required sample rate, filtering requirements, accuracy, precision, alarm levels, associated algorithms, display screen etc. Other variables and constants such as trip levels, calibration constants, time constants etc. should be included. Note: plant inputs and outputs should be assigned unique identifiers that reflect their purpose.

A10.5.3 The *Safety Review* should demonstrate how it is ensured that ambiguity in the data properties is removed and that all aspects of the data structures are adequately described. This might be done by modelling the data to show the relationships and hierarchies. The most appropriate technique uses the entity-relationship diagram originally proposed by PP Chen ("The Entity Relationship Model - Toward a Unified View of Data", ACM Trans on Database Systems, Vol 1, No 1, March 1976, pp 9 - 36).

<u>*Questions about Fulfilment of safety requirements*</u>

Comments:

The fulfilment of safety requirements means that the safety and integrity level (SIL) is preserved through the design phase. It also means what is done to design for enhanced safety. IEC-61508 contain several recommendations about this:

'If safety functions and non-safety functions are to be implemented in the same design, or if safety functions of different safety integrity levels are to be implemented in the same design, then the following shall be documented during the design:

- *the method of achieving the independence;*
- *the justification of the method.'*

'As far as practicable the design shall minimise the safety-related part of the software. Where the software is to implement both safety and non-safety functions, then all of the software shall be treated as safety-related, unless adequate independence between the functions can be demonstrated in the design.'

'Where the software is to implement safety functions of different safety integrity levels, then all of the software shall be treated as belonging to the highest safety integrity level, unless adequate independence between the safety functions of the different safety integrity levels can be shown in the design. The justification for independence shall be documented.'

'Select and justify an integrated set of techniques and measures necessary during the software safety lifecycle phases to satisfy the specification of requirements for software safety at the required safety integrity level. These techniques and measures include software design strategies for both fault tolerance (consistent with the hardware) and fault avoidance, including (where appropriate) redundancy and diversity.'

Q10.6 Do the design documents show how the actual design preserves the safety integrity level given in the requirement specification?

A10.6.1 The *Safety Review* should describe how the design achieves the required safety integrity level (SIL 3 or SIL 4). Appropriate selections should be made from IEC 61508-3 Annexes A to C for SIL 4 (preference should be given to the 'HR' techniques) unless it has been demonstrated though a suitable PSA sensitivity analysis that a lower SIL level is acceptable. Notwithstanding the selection of a lower SIL level, the safety system should still meet the requirements of this guide. The lower SIL level will only apply to protection channel that can be demonstrated to be "independent" from the rest of the safety system. The *Safety Review* should contain a full justification for this approach. Also, the detailed design documents should be such that it is possible to demonstrate (through the traceability matrix) that each functional and non-functional requirement has been met, and to verify that the design is correct with respect to these functions. It is suggested that this aspect be samples by the SKI assessor.

A10.6.2 The *Safety Review* should also provide a commentary showing how the selected design methodology, the proposed software tools, the chosen languages (including the compiler), the verification and validation techniques conform with the requirements set out in IEC 61508-3, Section 7.4.

A10.6.3 The *Safety Review* should show how the following principles (mostly extracted from the UK NII's Assessment Guide 3 - reference 12) have been met.

i) The design of safety systems should avoid unnecessary complexity. The simpler the design the easier it is to show how the design documents preserve the safety integrity level. It also gives greater confidence that the software will be fully understood.

ii) There should be a demonstration that any protective action claimed against the failure of a control function does not use common hardware or software. Where this cannot be demonstrated, then the *Safety Review* should demonstrate that adequate diverse provision has been made to enable the NPP to be brought to a safe state in a timely manner. (reference 12, section 6)

iii) The software architecture should be such that overall control flow is in a fixed sequence pattern and should not generally employ interrupts. Where interrupts are used they should be justified in terms of demonstrably simplifying the design. Their usage and masking during time and data critical operations should be checked for correct operation in-service and be well documented. The hardware and software should be designed to detect both unserviced and missing interrupts. All interrupt vectors should be initialised whether used or not. Unused interrupts should point to an error reporting procedure and should initiate protective action. (reference 12, section 49)

iv) The system and procedures should be designed so that only an approved program as a whole can be loaded into the system. (reference 12, section 100)

v) No program modifications should be necessary as part of system operation and no facilities should be provided for this purpose. Protection system software should not be altered to overcome operational or maintenance difficulties, other than on an off-line system using agreed modification procedures. Engineered facilities should be provided where there is a need to change specific system parameters such as trip settings, calibration constants etc. for operational reasons. The available range should be limited to values supported by the safety case[2]. A display of current values or states of such parameters should be provided. (reference 12, section 103)

[2] *In this case, the Safety Report and the Safety Reviews.*

vi) Facilities should be incorporated for an operator to determine the state of each input variable and output function. (reference 12, section 24)

vii) The software design should include an automatic test arrangement to enable in-service functional checks of the system hardware to be made from the input variables to the output action. A frequency for applying this test should be proposed with a justification based on the type of faults detected and their rate of occurrence. The insertion or removal of any test equipment should not require manual action to reinstate the system; however, there should be confirmatory feedback to the operator. (reference 12, section 25)

viii) The software architecture should ensure that the programs, together with their stored constants, are stored in a secure, reliable and permanent manner such that timely automatic restart is possible following power supply failure or any such detectable and recoverable failure. (reference 12, section 26)

ix) The design aim should be to prevent the unintentional operation of a safety device due either to operator error or to a fault in the equipment. Unsolicited changes in plant status should be detected and where possible prevented. (reference 12, sections 27 & 28)

x) The need for manual intervention (e.g.: for calibration or adjustment) should be kept to a minimum. (reference 12, section 29)

xi) The use of features not specified by the manufacturers (i.e. undeclared instructions or clock rates) should be prohibited unless their use is fully justified. (reference 12, section 30)

xii) The design should be consistent and complete, containing no contradictions and no ambiguities. The description of the interfaces between modules should be complete. Both sides of each interface between modules should match with, as far as possible, a consistent partial ordering and use of variable names between module input and output interfaces. (reference 3, section 8.11)

Q10.7 Are features for enhanced safety designed into the system:

- *Q10.7.1 A clear separation in the design between safety critical and not safety critical parts of the system?*

A10.7.1.1 The separation between safety-critical and non-safety-critical functions for the safety system should have been established in the *Overall requirements* phase. These non-safety-critical functions should be clearly identified in the *Safety Review*, and a *traceability matrix* should demonstrate that the system and software architecture clearly separate them out.

A10.7.1.2 The *Safety Review* should demonstrate compliance with the following principles.

i) Hardware diagnostic software and operator communications' software should be clearly separated from the essential safety kernel. Also, the provision of software for the detection and reporting of hardware failure (hardware diagnostic software) should not be allowed to complicate the system. (see reference 12, section 20)

ii) Concurrent interactions with other systems should be avoided. (reference 12, section 20)

iii) The structure of the operating system needs to be considered. A software architecture that separates the initialisation from the main program loop in the operating system should be considered. Also, a distributed architecture will prevent the interaction of software modules.

iv) Point-to-point communications are preferred to a shared communications bus. Electrical isolation via say opto-isolators should be considered. Other than the special maintenance systems, there should only be one-way transmission of data from the safety system to other systems such as alarm display systems. This one-way transmission of data to other systems should be on a regular basis, not as a result of demands from the other systems.

v) It is recommended that software-based safety systems not be designed with the flexibility to cover a wide range of non-protection duties through the use of customising data. With this proviso, sufficient features should be included to enable a reasonable degree of variation in the requirements to be achieved. (reference 8, section 2.2.4.2)

- *Q10.7.2 Measures for fault tolerance, like diversity or redundancy designed into the system?*

A10.7.2.1 The *Safety Review* should demonstrate how the principles of redundancy, diversity, separation between safety and non-safety functions, physical isolation and segregation have been applied to the design of the safety system architecture. The contribution made by each principle to meeting the requirements, including design criteria such as the single failure criterion should be documented.

A10.7.2.2 Functional diversity, where the same safety action is fulfilled by different means, should be used in safety systems of the highest integrity. The *Safety Review* should demonstrate, and justify, the level of functional diversity achieved. In particular, the detection of unsafe reactor states should be via different physical parameters, e.g. reactor coolant (RC) pressure low and RC cold-leg-temperature low to detect a LOCA in a PWR. Diverse hardware and software should also be considered for safety systems of the highest integrity. The claimed level of diversity, whether functional, equipment or software should be described and justified in the *Safety Review*.

A10.7.2.3 In the case of software diversity, the chosen design should be justified in the *Safety Review*, as should the method of implementation. For example, independent programming teams, diverse designs, and different working environments, tools and run-time support systems (operating systems, compilers) can be used. The justification should cover all these aspects and techniques, and should demonstrate that the added complexity does not negate the benefits of using diversity.

A10.7.2.4 The *Safety Review* should justify the method of arbitrating between the different systems. In general, the voting should be performed as far down the decision chain of each separate sub-system as possible in order to maximise sub-system diversity. It should also include a demonstration of the acceptability of the variations in the tolerances of the instrumentation and the analogue-to-digital converter (including its bit count), the timing differences (skewness) and the distance between measurements points.

A10.7.2.5 The *Safety Review* should provide descriptions and justifications for the following requirements (extracted from reference 8, section 2.2.4.1 and reference 12).

i) Before fault tolerant mechanisms are considered, a general principle should be fault avoidance through the use of highly reliable design elements. If a fault can be identified and avoided, it should be designed out first before fault tolerant mechanisms are considered. Thus, fault tolerance should not be used to compensate for deficiencies in the design. It should be used only if the deficiencies are well understood, unavoidable and specific of a fault type. The *Safety Review* should justify the need for each fault tolerant feature. (see reference 8, section 2.2.4.1.2)

ii) The efficiency of fault tolerant mechanisms depends on the knowledge of the faults to be protected against. Also, it works best when reliable components of design are employed, when the error rate is already low, and the fault mechanisms well identified. In addition it must be recognised that a fault tolerant mechanism can become a source of failures in its own right. These potential failures should be identified, and their impact on the fault tolerance capabilities should be assessed in the *Safety Review*. (see reference 8, section 2.2.4.1.1)

iii) The fault tolerance should cover both internally and externally arising exceptions, without adding unnecessary complexity to the design. (see reference 8, section 2.3.2.4)

iv) The *Safety Review* should demonstrate how the design has achieved its level of fault tolerance, i.e. the capability to mitigate the effect of errors, so as to prevent them from turning into failures that jeopardise system safety. A comprehensive list of faults should be included with a justification for its completeness. The mechanisms for the detection of these faults and their control should be included.

v) The design should address the results of the Safety System Failure & Hazard Analysis. The design should include the necessary supervision features to detect and report hardware failures. They should also include the necessary supervision features for the software. (reference 8, section2.3.3.1.3)

vi) Failure of one part of a redundant scheme should not adversely affect other parts of the system. (reference 8, section 2.2.3.4)

vii) Where redundancy is claimed the hardware should be physically and electrically segregated. (reference 12, section 67)

viii) Information transmission between redundant parts of a protection system should be avoided. Where this is not fully practicable then the system should be arranged so that malfunction of any redundant part does not adversely affect the other parts. The integrity of communication lines and multiplexers should be checked periodically to establish 'liveliness'. (see reference 12, section 68)

ix) Redundant portions of the system should not be operated synchronously unless it can be established that there are safety advantages in so doing. (reference 12, section 69)

x) All messages should be tagged with a serial number to enable the loss of a message to be identified. (reference 12, section 62)

xi) It should be demonstrated that the power-fail / restart procedures do not produce an unsafe state in redundant systems due to initial conditions and phasing. (see reference 12, section 53)

xii) Programs and fixed data (including operational data) should be held in read only memory so that they cannot be changed on-line either intentionally or due to a software error. Changes to in situ ROM should be subject to strict controls to avoid incorrect data entry (program logic should NOT be changed in this manner); consideration should be given to the use of special error checking software within the protection system, and within the modification device for off-line use, to check the correctness of any necessary fixed data changes. (reference 12, section 61)

xiii) The protection system should automatically record, in a secure manner, all fixed data changes within the system for subsequent analysis. (reference 12, section 63)

- *Q10.7.3 Ways to bring the system into a fail-safe state in case of identified failures during operation?*

A10.7.3 It should be demonstrated in the *Safety Review* that, for each identified failure, a course of action has been adopted which moves the NPP into a safe state, and that, as soon as practicable, the condition will be reported to the operator (see reference 12, section 52).

For example:-

i) A failed protection system train should be set to a trip state in a redundant architecture. Valves and pumps should either move to a safe state (either closed or open) or remain in an unaltered state depending on the most safe action (this is of particular concern when power supplies fail).

ii) Appropriately engineered facilities should be provided to ensure that operation of any testing facilities, manually or automatically initiated, should neither trigger spurious operation of tests, nor protective action nor degrade protective system reliability. Where testing causes inhibition of protective action, testing facilities should be regarded as part of the safety system and should be designed and engineered to the same standards. (reference 12, section 5)

iii) Data errors occurring in inputs and on transfer between modules should be trapped and alarmed; instrument readings should indicate when they are off scale. (see reference 12, section 42)

Questions about Design methodology

Comments:

The design should have a structure that facilitates abstraction, modularity and other features which control its complexity. It should also support verification methods, both forward and backward. This means that it should be possible to demonstrate the correct conformance of a design element to the corresponding requirement, and that the design element should have a form which makes it possible to verify, by testing and/or analysis, that the final system is in accordance with the design. As far as practicable, the design shall include software functions to execute proof tests and all diagnostic tests in order to fulfil the safety integrity requirement *of the system. A further aspect is that the design should support the possibility for safe modifications if needed at a later stage.*

The design representations shall be based on a design language which is comprehensible by developers and others who need to understand the design; and also defined or restricted to unambiguously defined features.

If pre-existing is to be used as part of the design then it shall be clearly identified. The software's suitability in satisfying the specification of requirements for software safety shall be justified. Suitability shall be based upon evidence of satisfactory operation in a similar application or having been subject to the same verification and validation procedures as would be expected for any newly developed software.

Pre-existing software also includes. tools to support both for development of the design document, and to validate aspect of the design. These tools should be of such a high quality that they may not have any negative effect on the safety integrity level of the system.

Q10.8 Is the design structure simple and comprehensible?

A10.8.1 The *Safety Review* should describe the design in terms of its set of interacting modules, and the specifications of those modules and of their interfaces. The design should be well-structured and understandable to the implementors, maintainers and testers. The design should demonstrably cover all requirements and should not contain any unnecessary items. (see reference 8, section 2.3.2.2) The *Safety Review* should contain an evaluation of the design structure by the developers and the users to demonstrate this understanding. The procedure that might be followed is described in a Software Engineering Institute (Carnegie Mellon University, Pittsburgh, USA) technical report, CMU/SEI-96-TR-025, January 1997, "Recommended Best Industrial Practice for Software Architecture Evaluation".

A10.8.2 The design should have at least two levels of design, namely, the architecture of the software and its detailed design specifications. The software architecture should relate as nearly as possible to the *Overall system requirements*, and should avoid complex and intricate solutions to design problems. The chosen architecture should employ modularity, information hiding and encapsulation as far as is practicable. (see reference 3, sections 8.4, 8.5 & 8.7)

A10.8.3 The software specifications should be traceable to the *Overall system requirements*, both functional and non-functional. The reliability, availability and security requirements should be carried forward into the software specifications. (see reference 3, section 3.26)

A10.8.4 The documentation should be adequately clear and precise so as to aid comprehension. The design should be documented by a diagrammatical representation which is supported by natural language descriptions. The structure should show a progressive decomposition of the design starting at a high-level overview and moving down in a hierarchy of more detail which finally results in a set of basic modules with their specifications.

For example the design might be based on the Yourdon Structured Analysis and Structured Design (SASD) approach which is functionality-based and covers:

Environmental Model - defines the boundary between the system and its environment.

Statement of Purpose - what the system does.
Context Diagram - shows external objects and interfaces.
Events List - events that trigger a response from the system.

Behavioural Model - defines the required behaviour of the system.

Data Flow Diagrams
Process specifications
Data Dictionary
Entity Relationship Diagrams
State Transition Diagrams

Implementation Model - defines the software specifications for the coders.
Structure Charts
Software specifications

Other modelling approaches include Object Oriented Analysis and Design (OOAD) methods - see "The Object Primer, the Application Developer's Guide to Object-Orientation", S C Ambler, SIG Books/Cambridge University Press, 1995 - which are object-based. However, these are not so readily suited to real-time process control type applications. A more recent development is design approaches based on the Unified Modelling Language (UML) which combine the benefits of SASD and OOAD. UML is a notation, or language, for visualising, analysing and designing object-oriented software systems. It is a combination of methodologies and was developed to merge previously competing modelling notations. UML is now an open standard controlled by the Open Management Group (OMG). To be of use in the design of realtime systems, UML needs to be extended to cover this aspect (see for example "Real-time UML" by Bruce Powel Douglass, Addison-Wesley, 1998). However, reservations have been expressed about the use of this approach since the semantics are said to be poorly defined.
.

UML is based on the following graphical diagrams:

- *Use-case diagrams*
- *Class diagrams*
- *Behaviour diagrams*
- *State diagrams*
- *Activity diagrams*
- *Sequence diagrams*
- *Collaboration diagrams*
- *Implementation diagrams*
- *Component diagrams*
- *Deployment diagrams*

With extensions that cover non-functional requirements and general *constraints (Constraints diagram*) and the *System Architecture diagram.* The *Safety Review* should provide a justification for the analysis and design methodology used. This justification should take into account that this is a safety-critical application with a need for rigorous verification and validation.

A10.8.5 The *Safety Review* should provide an analysis of the degree of *coupling* and *cohesion* between modules (these are measures of the understandability of a design - see "The Practical Guide to Structured System Design", Page-Jones, Meilir, Prentice Hall, 1988). *Coupling* is a measure of the interdependency between modules. Low coupling or interdependence usually indicates good, understandable design. Ideally, modules should only be linked by parameter passing. Module links that employ devices such as control indicators or common areas represent poor design structure. *Cohesion* is a measure of the degree to which the various elements of a module (data, instructions, subroutine calls) have in common with each other. A high degree of cohesion, say through the dedication of a module to a particular function, is indicative of good design structure - and is, therefore, more comprehensible.

A10.8.6 The *Safety Review* should provide a summary description of the interface between any system software and the application software.. Reference should be made to the more detailed documents that completely define and document this interface.

A10.8.7 The software architecture should also take into account the unavoidable changes that may occur during the lifetime of the system, so that software maintenance and upgrades can be performed conveniently. But this aspect of the design should not be such that it increases the complexity of the software and, consequently, decreases the understanding. (see reference 3, section 8.15 & 8.5)

A10.8.8 The *Safety Review* should demonstrate that the operating system runs in a fixed sequence pattern and does not generally employ interrupts (see reference 12, section 49 on interrupts). The *Safety Review* should justify the use of any interrupts in terms of demonstrably simplifying the program.

It should also demonstrate that:

- interrupt usage and masking during time and data critical operations is checked for correct operation in-service and that these aspects are well documented;
- the hardware and software is designed to detect both unserviced and missing interrupts;
- all interrupt vectors are initialised whether used or not;
- unused interrupts point to an error reporting procedure and should initiate protective action.

Q10.9 Is the design verifiable with respect to the requirements?

A10.9.1 In order to enable the design to be verified against the requirements, the requirements must be sufficiently detailed to enable a link to be made (through a traceablity matrix) between each element of the design and the specific functional and non-functional requirement. The use of graphical techniques can aid understanding and hence verification. It is preferred that the design should be described formally using a graphical technique with a well-defined syntax and semantics, and with explanations in natural language. The design should contain no contradictions and no ambiguities. The *Safety Review* should describe how the chosen design aids the verification against the requirements. The characteristics described in this response provide some guidance. (reference 3, section 8.7)

A10.9.2 To aid verification, the software architecture chosen should be deterministic. A design should be selected that makes the operation of the software predictable in terms of response to inputs and the time to produce a response. A fixed, repeated sequence of operations (e.g. polling) is generally recommended over the use of interrupts. Communication protocols should be deterministic and should not depend on the correct operation of other, external systems. (reference 3, section 8.10)

A10.9.3 The safety system software architecture should be organised on a modular basis in a manner that minimises the risk of design faults and also facilitates verification. The appropriate use of hierarchical modularization, encapsulation and information hiding should be justified in the *Safety Review* to show that the design of a module is restricted to one clearly identified function that requires only minimum interaction with other functions (simple interfaces) and minimises the impact of changes. The interfaces between the various modules shall be simple, completely identified and documented in the *Safety Review*. (reference 8, section 2.3.3.2.1)

A10.9.4 The description of the interfaces between modules should be complete. Both sides of each interface between modules should match with, as far as possible, a consistent partial ordering and use of variable names between module input and output interfaces. (reference 3, section 8.11)

A10.9.5 Modules should be defined in a way which makes it possible to reason about and test a *module part* (program) in isolation from other module parts and from other modules. The contextual information which is required for these activities should be part of the module documentation. The modules should include processes, abstract data types, communication paths, data structures, display templates, etc. (reference 8, section 2.3.4.2.2)

A10.9.6 To facilitate traceability and verifiability, each design element such as a module, a procedure, a sub-routine or a file should have a unique identifier. Also, the naming and encoding of data structures, input, output and internal variables should be meaningful in the system environment (for example, names such as set-point, temperature, trip, should be used). Names should be used consistently throughout the whole software. (reference 8, section 2.3.4.1.3)

A10.9.7 If during the design process, shortcomings in the requirements specification are identified, then these should be reflected as changes in the functional requirements documents using a formal change process which requires a re-approval of these additional requirements. The application of this approach should be documented in the *Safety Review*.

Q10.10 If yes, has the design been verified?

A10.10 For safety systems, the design must be verified against the requirements. Evidence that this has been satisfactorily performed should be included in the *Safety Review*.

Q10.11 Has the design document a structure which facilitates verification of the finally realised system?

A10.11.1 The *Safety Review* should provide evidence that the design documentation of a software-based safety system conforms to the following:

i) Technical information on the overall architecture of the software and on the specifications of all software modules should be referenced. Relevant implementation constraints should also be specified. (reference 3, section 8.13)

ii) A *traceablity matrix* should be available showing the decomposition into modules and the mapping of this decomposition onto the requirements. This should be supported by the graphical and natural language descriptions together with appropriate justification. The existence of interfaces between the various software modules, and between the software and the external environment (as per the Requirements Specifications) should be identified and clearly specified in the documentation. (see reference 3, sections 8.15 & 8.16)

iii) If the safety system includes multiple processors and the software is distributed amongst them, then the description of the software architecture should specify which process runs on which processor, where files and displays are located and so on. (reference 3, section 8.17)

iv) The implementation constraints should be described and justified and be traceable to higher level requirements or constraints. Such constraints include the methodology for ensuring software diversity, and the attributes required of the programming language(s), compilers, subroutine libraries and other supporting tools should be described. (see reference 3, section 8.19)

v) Documented evidence should be given that, on failure detection, the actions taken in terms of recovery, halting procedures and error messages maintain the system in a safe state. (reference 3, section 8.20)

vi) Each software module identified in the software architecture should be described in the detailed design. The particular content of the description will depend on the module type. In general, a module description should completely define its interface with other modules, and should completely define the module function and its role in the overall software. (reference 3, section 8.21)

vii) The common techniques for describing design such as data flow diagrams, structure diagrams or graphical methods, including flowcharts should be used as long as the meaning of the elements of the diagrams are well defined. (see reference 3, section 8.23)

Q10.12 Does the design facilitate future modifications in a safe and orderly way?

A10.12.1 The *Safety Review* should describe the information hiding and encapsulation aspects of the design and demonstrate that their level of use is sufficient to facilitate future modification. The description should also cover the module interfaces to show that they are simple, such that the consequences of expected changes are confined to a single or to a small number of modules. (see reference 8, section 2.3.4.2.1) The demonstration of the simplicity of the interfaces should be included in the *Safety Review* by reference to the degree of module *coupling* and *cohesion* (see A10.8.5).

A10.12.2 A modification is more likely to be conceived and implemented correctly if the system's design is well structured, modular, well documented and containing features which aid changes (see above response). However, it has to be recognised that a system that has been designed to be easily maintainable may, in fact, require an overly complex design whose safety may be difficult to demonstrate. (see reference 8, section 2.3.2.6) The *Safety Review* should justify any increased design complexity added by features that have been included in the system to facilitate modification.

A10.12.3 The *Safety Review* should describe how it has been ensured that no program modifications are necessary as part of system. It must be ensured that safety system software need not be altered to overcome operational or maintenance difficulties, other than on an off-line system using agreed modification procedures. Engineered facilities should be provided where there is a need to change specific system parameters such as trip settings, calibration constants etc. for operational reasons. The available range should be limited to values supported by the safety analysis. A display of current values or states of such parameters should be provided. (reference 12, section 103)

Q10.13 Have tools been used to validate that the design is correct in conformance with the requirements?

A10.13.1 The *Safety Review* should provide a justification for any design methodology that does not use a tool(s) to validate that the design conforms to its requirements. For example, formal methods that are applied purely manually require the involvement of very well trained personnel, and despite this, are still highly error-prone. Hence, these formal methods should be supported by tools. (see reference 8, section 1.4.4.2)

A10.13.2 The *Safety Review* should describe the tools used to validate that the design conforms to the requirements, and should demonstrate that they are stable, well-supported and have been shown to have been designed, validated and maintained under a quality management system equivalent to that required by this guide.

A10.13.3 The *Safety Review* should describe the integrated project support environment that incorporated the tools, or demonstrate how adopted approach ensured proper control and consistency. For example, the *Safety Review* should show how the adopted approach provided the facilities of a data dictionaries that is able to check for inconsistencies in the database, ensure the correctness of values of items entered and enable data to be interrogated. In addition, the *Safety Review* should demonstrate how the exporting and importing of data to and from other sources was supported. Tools should also be used for configuration management to ensure consistency. (see reference 8, section 1.4.4.3)

Q10.14 Is the design language comprehensible by developers and others who need to understand the design?

A10.14 The *Safety Review* should demonstrate that the design language is adequately supported by comprehensive documentation and suitable training courses that fully explain the syntax and semantics. It should also describe examples of the use of the design language in safety-critical applications. Training records for the developers and others should be summarised in the *Safety Review*, together with a commentary on the developers and others views on the comprehensibility of the design language.

Q10.15 Has the design language a well defined syntax and semantics with unambiguous terminology?

A10.15.1 According to IEC 61508, 7.4.2.5, "The design representation shall be based on a notation which is unambiguously defined or restricted to unambiguously defined features." Therefore, the *Safety Review* should demonstrate that the chosen design language has a well-defined syntax and semantics with unambiguous terminology.

> *Languages such as VDM, Z, LOTOS and OBJ are considered to be design languages by IEC 61508. These all have well defined syntax and semantics but are not suitable for all aspects of the design process. For example, neither Z nor VDM cover temporal or concurrency issues. Other languages with well-defined syntax and semantics are B and Timed Petri Nets. Although B does not cover temporal aspects, Timed Petri Nets do. Different design languages can be used to analyse different aspects of the design. For example, LOTOS is ideal for describing and analysing the behaviour of communications protocols; and CSP is similarly useful for aspects of the design which involve a degree of concurrency such as with shared buffer stores.*

A10.15.2 Graphical representations are useful, since they provide a more readily understood picture of the system. The *Safety Review* should demonstrate that the chosen graphical representation has a rigid syntax in terms of connection types, and a well-defined semantics in terms of the functions performed being mathematically or logically defined (including their restrictions and limitations). (see reference 8, section 1.8.4.2.2 & reference 3, section 8.7))

Q10.16 Have development tools been used which improve the quality of the design document?

A10.16 The *Safety Review* should justify the lack of use of any development tools to generate the design documentation (see reference 3, section 4.7). The justification should include a demonstration of the adequacy of the manual method of document verification for accuracy, completeness, consistency and lack of ambiguity (see reference 3, section 10.5).

The tools that are used should be described and justified in the *Safety Review*. For example, does the tool provide appropriate summaries and textual facilities to amplify the design notations; and are word processing facilities provided which have the ability to import diagrammatic information and data tables generated by other tools.

Q10.17 Does the design specify or indicate the use of pre-existing software, as e.g.

- *COTS*
- *Supporting tools*
- *Development tools*

Pre-existing software is also covered in the responses to questions Q11.10, Q11.11, Q11.12 and Q11.13.

For COTS

A10.17.1 The *Safety Review* should indicate which elements of the design are implemented using pre-existing modules (COTS) and which require new software.

For Supporting and Development Tools

A10.17.2 The *Safety Review* should describe, and justify the use of the chosen software development tools, whose output becomes part of the program implementation, and which can therefore introduce errors (code generators, compilers and linkers are examples) and the chosen software verification tools, that cannot introduce errors but may fail to detect them (static analysis tools and test coverage monitors are examples). (see reference 3, section 10.29)

A10.17.3 The justification should include arguments supporting the adequacy of the definition of the tool's functionality. For a software development tool, the arguments should demonstrate that the domain of applicability is precisely known; and for a software verification tool, the demonstration should be that the analyses or checks it performs are well-defined. (see reference 3, section 10.30)

A10.17.4 The justification should show that the development tools are of sufficient quality to ensure they do not jeopardise the safety of the safety system. Therefore, a design tool whose output is used without further review should be of the same integrity level as that of the safety system. This requirement may be reduced if its output is subjected to verification or if reverse engineering is used (see note with A11.10.1). For a software verification tool the requirements may also be reduced somewhat, on the grounds that its output will be closely scrutinised. (see reference 3, section 10.31)

A10.17.5 Examples of methods of validating a tool, i.e. justifying its use, are (see reference 8, section 1.4.4.1):

- verify the output by inspection;
- run the output of the tool on the target system or on a simulator of this system and verify this against the expected behaviour;
- certifying that the tool is processing correctly all valid statements and combinations thereof contained in the definition of a programming language;
- using appropriate profiles and reviewing a sufficient amount of previous operational experience.

11 System realisation

At this milestone all parts of the system are made, but not yet installed in the plant. All standard hardware and software are purchased, and all tailor made programs have been made. At this stage all documentation and information required for further evaluation should also be available. The influence diagram at this milestone is shown in Figure 8

The milestone goal here is dependable system. This is influenced by three aspects: The conformance of the system to the design requirements, what is done to obtain high quality for the system, and what measures are made to obtain fail safety. A fourth aspect also belongs here, viz. system validation, as that should be an integral part of system realisation. This is, however, treated as a separate milestone, in agreement with IEC-61508.

<u>*Questions about Conformance with design specification*</u>

Comments:

The conformance of the system with the design requirements implies both that there is a clear correspondence between the system solution and the design by forward and backward traceability, and that all parts of the design are correctly implemented in the final system

Q11.1 Is there a clear correspondence between the system solution and the design?

A11.1.1 The *Safety Review* should contain a *traceability matrix* showing the links between the elements of the software specifications for the software modules and their interfaces (the outputs of the *Overall System Design* milestone) and their associated code fragments. These modules and interfaces should form part of an overarching design structure that covers the software architecture, namely the information and control flow and the data structures. The *Overall System Design* in turn links back to the functional requirements specification (covered by the *Overall Requirement* and the *Requirement Specification* milestones). This should be confirmed by the verification phases between the Software-based System Design, Software Requirements, Software Design and Software Implementation activities of Figure 9.1, "Typical Lifecycle Model".

Q11.2 Are all parts of the design correctly implemented in the final system?

A11.2 The verification and validation activities assessed in Chapter 12 will enable the correctness of the final system to be determined. Obviously, the reply to this question must be *YES* provided the requirements of Chapter 12 are met. The V&V activities should be fully discussed in the *Safety Review.*

<u>*Questions about System quality*</u>

Comments:

System quality features are those features which contribute to making the system free of failures and thereby reliable. One can distinguish between three types of system quality:

structural quality, hardware quality, software quality and the quality of the man-machine interfaces.

The structural quality refers to how the system is build up by different modules. IEC-61508 states:

The software should be produced to achieve modularity, testability, and the capacity for safe modification. For each major component/subsystem in the description of the software architecture design, further refinement of the design shall be based on a partitioning into software modules (i.e. the specification of the software system design).

The use of comprehensible and verifiable methods in this refinement process should be favoured. Formal development method facilitates this process.

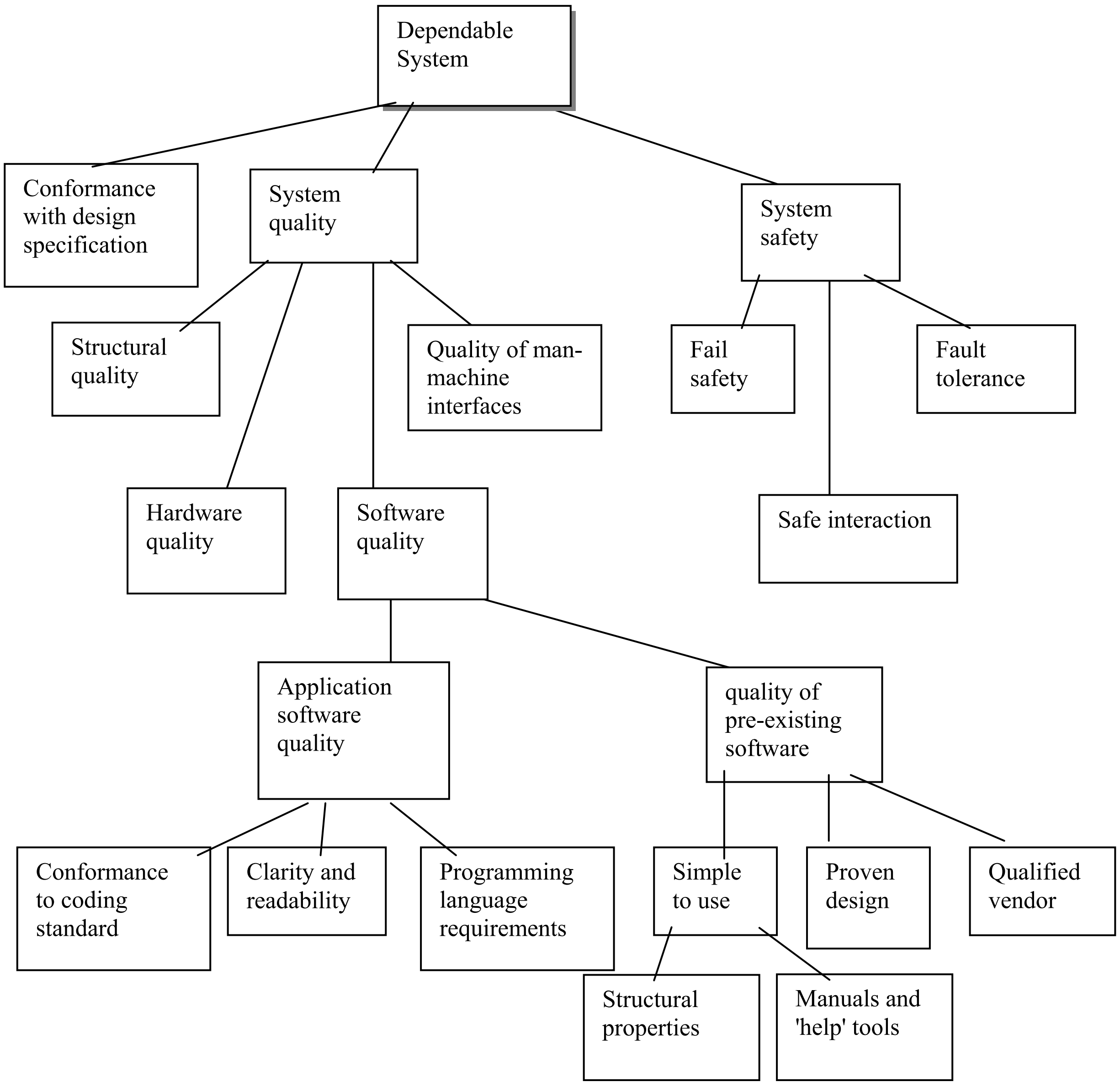


Figure 8 Influence net for milestone *System realisation.*

Q11.3 Is the transfer from the design to the realised system made in a comprehensible and verifiable way?

A11.3.1 Reference 8, section 2.4.2.1 states:-

"The instruction sets of digital computers allow, through the use of branch instructions (including computed branches), index registers (used as address pointers) and indirect addressing, extreme complexity to be introduced into the code (both through poor structuring or the addition of unnecessary functionality). This complexity in coding can result in the introduction of faults during the coding phase and when making software changes. It also makes the detection, during the *verification* and *validation* phases, of any fault introduced more difficult. To address these problems, strict coding rules need to be defined and enforced. These coding rules should impose a stringent discipline on the writing of the safety system software and ensure that code legibility is given priority over ease of writing."

The *Safety Review* should contain an analysis of the applicability of the coding standard used for the software language chosen for the development.

A11.3.2 Well-structured programs employing good programming practices will, in the main, be more understandable, and hence more dependable, than those that do not follow these practices. Information hiding and modules that perform only a limited number of related functions (encapsulation) contribute significantly to the quality of a program's structure. Programs should not have unreachable code or unused variables, should have (as far as possible) single entry and exit points and should not use uninitialised variables. The modules themselves should also exhibit good internal structure. This can be measured by treating the program as a directed graph consisting of nodes linked by arcs. The nodes represent branches in the program and the arcs represent a set of statements between two nodes. The program is said to be well-structured if its directed graph can be reduced to a single entry node and single exit node linked by an arc through applying Dijkstra's and Hecht's reduction rules (see "Software metrics - a rigorous approach", N Fenton, Chapman and Hall, 1991). The *Safety Review* should contain a justification for the level of use of *information hiding* and *encapsulation*, and should justify the structure of any module that does not reduce satisfactorily.

A11.3.3 To aid the verification stages reference 3, sections 9.28 and 9.29 states that "the code of each program of a module should be displayed in a specific section of a document together with the contextual information necessary to verify the correctness of this program to its specification. The contextual information provided should be sufficient to serve as a basis for understanding and testing a program of a module in isolation from the other module's programs and modules. This module contextual information should contain a replicated description of, or a reference to the relevant fragment of the software design specifications, the logical assertions, or the pre- and post-conditions that should be satisfied by every program of the module. It should also identify the other programs and modules that call upon, and are called upon, and the program input and output variables and parameters, together with their range of validity. To facilitate the reviewing of the code, this document should be no longer than one or two pages."

The *Safety Review* should provide a description of the software documentation and show how it complies with this requirement.

Q11.4 Have any formal or semi-formal development methods been used to facilitate this process?

A11.4.1 The development of the code is a progressive process which is shown in outline in the life-cycle model of figure 9.1. This can be supported by the use of formal development methods such as the use of VDM, Z, B, LOTOS, CSP etc. or by semi-formal methods such as structured program development of which Yourdon's and Jackson's methods are examples. The use of such methods improves software quality, understandability and maintainability, and is thus a desirable feature. In fact, the development process may need to use a combination of formal and semi-formal methods to achieve the complete design and coding (see "Evaluation of YSM and VDM for Time-critical Systems", S Goldsmith, Computing and Control Engineering Journal, March 1990, pp 87 - 94). The process of progressive refinement means that the design is taken to the stage where sufficient detail is available to enabel the coding to begin. The *Safety Review* should contain a summary report on this process with a justification for the level of formality employed. (See reference 8, section 1.8, "Formal Methods" for a general review of this aspect.)

Q11.4.1 If yes, has this development been supported by tools?

> *There are a number of formal and semi-formal tools for software development available on the market which cover the full development of a software-based system. Some of these employ the semi-formal technique of structured design enabling developers to produce graphical representations of the system which can be validated through animation. Others use a formal mathematical method such as Z, VDM or B which can be formally verified and validated. The tools that are of particular interest at the realisation stage use automatic code generation. The use of such validated tools is recommended for safety systems as they relieve programmers from some of the clerical error-prone programming and manual verification tasks of the code production process. A suitable set of validated tools will include translators (compilers and assemblers) and code generators, debuggers, linkers, testing and other verification tools.*

A11.4.1.1 The *Safety Review* should provide a summary description of the tools chosen together with a justification for the particular selection (for general guidance see reference 8, Chapter 1.4, "Tools").

In particular the summary and justification should cover (but not be limited to):

> the validation of the translator, or through additional analysis and verification, the demonstration that the translation is correct;
>
> the compatibilty with other tools used in this, and other phases of the development.
>
> the error detection capabilities of the compiler of the programming language used to write the software (reference 8, section 2.4.3.7.3;
>
> the none use of the same version of the compiler for all parts of the software - all software should be re-tested if a new version of the compiler has to be used,

unless it can be demonstrated that the machine code has not changed (see reference 8, section 2.4.3.7.4);

any compiler code optimisation: code optimisation should be simple and provably correct; the same optimisation options should be used on all the software; array bound checking should be included (see reference 8, section 2.4.3.7.5).

Q11.5 Is the quality of the hardware in the system adequate?

A11.5.1 For an adequate safety demonstration, the *hardware system architecture* that is proposed should be traceably derived from the Overall Requirements with documented arguments for the choices being made. This aspect should be discussed in the *Safety Review*.

A11.5.2 The *Safety Review* should also discuss the following:

i) the design of the hardware which enables all hardware faults that may affect safety or reliability to be detected and reported. The *hardware system architecture* should ensure that the safety system is periodically testable in service, either by auto or manual testing, without degradation of reliability (test periodicities should be derived from the system reliability and availability requirements, and hardware faults should be postulated, even those that will remain undetected; these faults should be taken into account in the consideration of the required hardware redundancy and tests periodicities) (reference 8, section 2.2.3.8);

ii) how, in the presence of faults, the system moves to a subset of states that can be demonstrated to satisfy the system safety requirements - this should be true even in the presence of an external fault like a loss of electrical power to the system or to its input/output devices (see reference 8, section 2.2.3.8);

iii) the hardware design which ensures that failure of a part of a redundant safety system does not adversely affect other redundant or common parts of this safety system (reference 8, section 2.2.3.4);

iv) the *hardware system* Failure and Hazard Analysis which demonstrates that the possible failure modes of the *architecture* that may compromise the safety functions have been taken into account, and that adequate exception handling mechanisms and hazard mitigating functions have been included in the design (both software and hardware) (reference 8, section 2.2.3.6);

v) the demonstration, by analysis, that the hardware system meets the reliability requirements

.

vi) how the correct hardware configuration is maintained - e.g. through (see reference 12, section 70):

clearly labelling hardware modules and bin positions with a unique part number and serial numbers;

taking special precautions where modules are configurable through only using reliable means of configuring such as soldering or wire wrapping;

providing features which detect (or prevent) the insertion of an incorrect module, absence of a required module or incorrect insertion of a correct module;

providing features that prevent the incorrect connection of cables due to either wrong orientation or location.

A11.5.3 The *Safety Review* should include a compliance statement for the hardware design and components against the following well establish nuclear standards:

i) IEC 60987, "Programmed Digital Computers Important to Safety for Nuclear Power Stations", 1989;

ii) ANSI-7-4.3.2, "IEEE Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations", 1993;

iii) IEEE 603, "IEEE Standard Criteria for Safety Systems for Safety Systems for Nuclear Power Generation Stations", 1991;

iv) IEEE 627, "IEEE Standard for Design Qualification of Safety Systems Equipment Used in Nuclear Power Generation Stations", 1980;

v) IEC 60709, "Separation within the reactor protection system", 1981;

vi) IEC 60780, "Qualification of electrical items of the safety system for nuclear power generating stations", 1984;

vii) IEC 61225, "NPPs I&C systems important to safety. Requirements for electrical supplies", 1993.

A11.5.4 The *Safety Review* should also include a compliance statement for the hardware against the relevant standards on electromagnetic interference (EMI) and radio-frequency interference (RFI) immunity. For example,

IEC 61000-41: 1993; -2: 1995; -3: 1995; -4: 1995; -5: 1995; -6: 1995, "Electromagnetic compatibility".

A11.5.5 The software and trip settings should be held in non-volatile memory (such as EPROM) that can only be changed through the software modification procedure. Calibration data should be held in non-volatile memory (such as EEPROM) that can be changed on-line but does not need to be reloaded on re-start and cannot be corrupted by software errors. Any variation from this requirement should be justified in the *Safety Review* by showing that the proposed solution ensures that re-start is reliable, and the software and data cannot be corrupted. (see reference 8, section 1.7.3.2.2)

Q11.6 Are the man-machine interfaces clear and comprehensible?

A11.6.1 The *Safety Review* should show how the man-machine interfaces are linked to the requirements; and discuss how the interfaces have been developed from, say, a task analysis covering the postulated fault sequences. The task analysis will enable the determination of the required information for each fault sequence, and the required controls to enable the operator to perform the required tasks. The *Safety Review* should contain an analysis of the control room provisions to show that they meet the safety system design requirements. It should also include the provisions required in the event of total loss of the safety system computers.

A11.6.2 These controls should meet the appropriate international standards such as:

i) IEC 60694, "Design for Main Control Rooms of Nuclear Power Plants", 1989;

ii) IEC 61227, "Nuclear Power Plants - Control Rooms - Operator Controls", 1993;

iii) IEC 60965, "Supplementary control points for reactor shutdown without access to the main control room", 1989.

The *Safety Review* should contain a compliance statement against the chosen standards, together with a summary discussion of the task analysis and design process.

Questions about the application software:

Comments:
Application software means software which is written particularly for the actual system. IEC-61508 states:

The source code shall:
- *be readable, understandable and testable;*
- *satisfy the specified requirements for software module design;*
- *satisfy the specified requirements of the coding standards;*
- *satisfy all relevant requirements specified during safety planning.*

Each module of software code should be reviewed.

For user application programming using a limited variability language, for example ladder logic and function blocks, detailed design should still follow good programming practices. Hence, the software should be designed in a structured way, including: organising the software into a modular structure that separates out (as far as possible) safety-related parts; including range checking and other features that provide protection against data input mistakes; using previously verified software modules; and providing a design that facilitates future software modifications.

To the extent required by the safety integrity level, the programming language selected shall:
- *have a translator/compiler which has either a certificate of validation to a recognised national or international standard, or it shall be assessed to establish its fitness for purpose;*

- *be completely and unambiguously defined or restricted to unambiguously defined features;*
- *match the characteristics of the application;*
- *contain features that facilitate the detection of programming mistakes; and support features that match the design method.*

If this cannot be satisfied, then a justification for an alternative language used shall be documented during software architecture design description (see 7.4.3). The justification shall detail the fitness for purpose of the language, and any additional measures which address any identified shortcomings of the language.

Coding standards shall be:
- *reviewed as fit for purpose by the assessor, and*
- *used for the development of all safety-related software.*

The coding standards shall specify good programming practice, proscribe unsafe language features (for example, undefined language features, unstructured designs, etc) and specify procedures for source code documentation. As a minimum, the following information should be contained in the source code documentation:
- *legal entity (for example company, author(s), etc);*
- *description;*
- *inputs and outputs; and*
- *configuration management history.*

Programming related recommendations are also given in section 6.5.2 in SKI-guide-94:9. In summary, the application software shall be clear and readable, to support verification, it shall conform to accepted coding standard, and it shall be written in a language which fulfils the requirements of a safe programming language.

Q11.7 Does the application software conform to the coding standard set for this system?

A11.7.1 For safety systems of the highest integrity, *ALL* project specific code (operating systems, I/O drivers etc. as well as application software) must comply with the coding standards. The method of demonstration of this compliance should be described in the *Safety Review*. Pre-existing software is covered elsewhere.

A11.7.2 The detailed recommendations given in Appendix B of IEC 60880 for the coding of software should be followed. All exceptions to these recommendations should be duly justified in the *Safety Review*. However, the recommendations contained in the response to this question must be followed under all circumstances.

Q11.8 Is the code clear and readable?

A11.8.1 The *Safety Review* should contain a justification for the choices made in the coding standard - the standard should cover the point made in the A11.8.2. The *Safety Review* should also describe and justify the method used to ensure that the coding standards are followed. A tool-assisted method is preferred.

A11.8.2 The following standards (taken from reference 8, chapter 2.4, "Coding and programming directives") should be met for coding, modules and subroutines, data structures and addressing, and documentation. *The Safety Review* should contain a compliance statement, with supporting evidence, against each requirement, or there should be a justification for non-compliance.

Coding (reference 8, section 2.4.3.3)

The code should be designed so as to facilitate static analysis, testing and readability by, as much as possible, running in a direct and fixed sequence pattern, i.e.:

- Computed branching should be prohibited;
- Branching into loops, modules or subroutines should be prohibited;
- Dynamic instruction changes should be prohibited;
- Interrupts should be avoided unless they lead to a significant simplification. Where interrupts are used, their usage and masking during time and data critical operations should be proven correct and should be well documented. The use of high level *synchronisation programming primitives* should preferably be used to deal with interrupts. The hardware and software should be designed so that every interrupt is either serviced or explicitly masked.
- Recursion - not amenable to static analysis - and re-entrancy should be avoided unless they are shown to produce simpler code than by other means;
- Nesting of loops should be limited to a specified maximum, and modification of loop control variables within the loops should be prohibited;
- All alternatives in switch or case statements should be explicitly covered by the code. Appropriate actions should be implemented to deal with default conditions;
- Indirect addressing should be used carefully so that computed addresses should be easy to identify.

Dynamic storage allocation should be prohibited.

Modules and subroutines (see reference 8, section 2.4.3.4)

Modules should not exceed a limit specified for the system (e.g. 50 or 100 statements, or the amount of coding that can be placed on one page) without written justification - simplification of the software architecture and module structure might serve as adequate justification for exceeding the limit.

Unnecessary code compacting, programming tricks and unnecessary optimisations that make the understanding of code more difficult should be avoided.

Operations involving different types of variables should be avoided unless conversions appear explicitly.

A maximum should be specified for the depth of nested subroutines.

Subroutines should communicate with their environment exclusively via their parameters.

Subroutines should have only one entry point and should return from only one point.

The coding phase should start only after the verification team has approved the design phase. (reference 8, section 2.4.4.1.1)

Evidence should be provided that structured programming techniques have been followed. (reference 8, section 2.4.4.1.2)

Backward branching and branching out of loops should be avoided, except in the case of error exit. (reference 8, section 2.4.4.1.3)

Data structures and addressing (reference 8, section 2.4.3.5)

Variables and data should be explicitly declared and assigned. Explicit initialisation of all variables should be made before their first use. A variable should not be used for more than one purpose.

Distinctly different symbolic names should be used to represent constants, variables, modules, procedures and parameters.

Equivalence statements, especially referring to a common or global area should be prohibited.

Arrays should have a fixed, pre-defined length. The number of dimensions in every array reference should be equal to the number of dimensions in its corresponding declaration. Dynamic computation of indexes should be prohibited.

Arithmetic expressions should reflect the equations they represent. They should be executed in binary fixed-point arithmetic, unless the use of floating-point arithmetic can be demonstrated to contribute to safety - in the latter case, the hardware and software used to implement floating point functions should be suitably qualified.

The use of global variables within subroutines should be avoided. (reference 8, section 2.4.4.2.1)

The symbolic names and types of variables, constants, modules, procedures and parameters should be meaningful at the application level. (reference 8, section 2.4.4.2.2)

Documentation (reference 8, section 2.4.3.9)

The source code should include comments to a level of detail that is consistent throughout the whole software. Comments should be consistent with the documentation of the software requirements and of the design. They should be a sufficient complement to this documentation to make the code readable and understandable.

In particular, comments, related to the code of every software module or program, should contain all the complementary information necessary for an independent and stand alone review and verification of this code against its specification.

Comments should also include any specific information required for maintenance of the code, in particular, its history and the designers' names.

If a tool has generated the code, the documentation standards described here apply to the language description of the tool input.

The code of each part of a module should be displayed in a specific section of the module documentation together with the textual information necessary to verify the correctness of this module part to its requirements

Q11.9 Is the programming language acceptable for a safety related system?

<u>High Level Languages</u>

A11.9.1 The choice of programming language(s) used should be justified and documented in the *Safety Review*. The justification should cover at least the points raised below in this response.

A11.9.2 For the application software, a problem-oriented language (including graphical languages) rather than a machine-oriented language should be used since this will facilitate the understanding of the system by all persons concerned in the development (see reference 8, section 2.3.4.3.1). Where this is not the case, the *Safety Review* should demonstrate how all persons involved gained an adequate level of understanding of the language chosen for the application's level.

A11.9.3 The programming language (or the subset which is used) should have a rigorously defined and documented syntax and semantics (see reference 8, section 2.3.3.3.2). The syntax should be defined in Backus-Naur Form - if this is not the case, there should be a justification for the method used, with reference to BNF. The semantics of the language should be defined by other than natural language (natural language can be imprecise and ambiguous). For example, formal approaches such as *denotational semantics* (constructs are mapped by a *valuation function* into their meanings - denotations), *operational semantics* (each construct is defined by a transition rule - abstract machine) and/or *axiomatic semantic* (constructs are defined by axioms using pre- and post-conditions) might be used to give the precise meaning of the language (see Glynn Winskel, "The formal semantics of programming languages - An introduction", Foundations of Computing Series, Cambridge, MA, 1994). This will enable compilers to be constructed and assist in formal verification. A summary description of the language, together with its syntax and semantics should be provided in the *Safety Review*.

> *Languages such as ADA have for a long time had a safe subset, but the full implementation of ADA is not now considered a useful language for safety-critical software because of the increased complexity. Conversely, C has always been considered unsuitable for safety-critical software but now, due to work by the UK's Motor Industry Software Reliability Association (MISRA), there is a MISRA C standard which is promoting the use of "Safe C" (MISRA do , however, still prefer the safe subset of ADA). Where a proprietary language is used such as Intel's PL/M-86 (used for the Primary Protection System on the UK's Sizewell B PWR), the syntax and semantics are defined by the supplier since they have produced the compiler.*

A11.9.4 Reference 3, section 9.22 reminds us that "most languages suffer from insecurities, which make it difficult or even impossible to detect violations of the language rules either by the compiler or by analysis of the program text. If these insecurities cannot be eliminated by discarding some language features, or by adding additional static-semantic rules, their use should be avoided or at least restricted, identified and thoroughly verified."

Hence, any programming language used should be restricted to its safe subset, i.e. that set of language features that are unambiguously defined and secure. A high level, strongly typed, programming language that prohibits the unsafe mixing of variable types should be used. Run time type checking should be used when available. (see reference 8, section 2.4.3.7.1) A justification of the safe subset, and of the safe outcome of any unsafe features used should be provided in the *Safety Review*.

A.11.9.5 The *Safety Review* should provide a justification for any programming language and its translator which, by their design, prevent the use of error-limiting constructs, translation-time type checking, run-time type and array bound checking, and parameter checking. These run-time checks, however, may be dispensed with if the implementation is mature and thoroughly trusted or if they would require excessive processor power at run-time. (see reference 3, section 9.9) However, the maturity or use of excessive processing power should be supported by documented evidence in the *Safety Review*.

Assembly Language

A11.9.6 Where assembly language is used, the *Safety Review* should demonstrate that its quantity has been kept to a minimum and it has been subject to the same strict coding principles and controls as for a high-level language (reference 8, section 2.4.3.7.2). In particular, assembler and machine code insertions into high-level language code should not be allowed without justification (reference 8, section 2.4.4.3.1).

A11.9.7 The use of features not specified by the manufacturer, for example undeclared instructions or clock rates, should be prohibited unless their use is fully justified in *the Safety Review*, say through written support from the manufacturer specifying in detail the precise behaviour of the instructions or declaring that the clock rate will not jeopardise the reliability of the processor. (see reference 8, section 2.2.4.3)

A11.9.8 The *Safety Review* should demonstrate that, where assembly language is used, it is fully justified by real-time performance or compatibility constraints and that the modules are of limited size and functionality. (see reference 3, section 9.8)

A11.9.9 The assembler language used should be demonstrated (in the *Safety Review*) to have sufficient facilities for supporting modular programming and for structuring the code in terms of procedures, sub-routines or sub-programs with specifications and calling sequences distinct from their code bodies. (see reference 3, section 9.8)

Questions about Pre-existing software:

Comments:
Pre-existing software is standard software which is already developed, and which is used either as software modules in the system(COTS), software in standard hardware components (firmware) or software which forms an operating system. Software which is used in the generation of the system (compiler, linker, loader etc.) are known as transformation tools and need to be subjected to an adequate qualification process.

As pre-existing software is generally developed by other people that the user, it is a requirement that the way to use the system is well communicated from the developer to the user. A good structure of this software will ease this communication and make it less error prone. Another important factor is to have a good user manual and 'help' tools to explain the use of this software.

A pre-existing software system which shall be used in a safety related application should also show its quality through proven design, i.e. through an extensive usage by a multitude of users. An indication on the quality of the pre-existing software system may be obtained through an investigation of the producer and/or vendor of the system.

Q11.10 Is this software well defined, unambiguous and easy to understand, so that incorrect use is unlikely?

For COTS
(The following requirements have been extracted from reference 8, chapter 1.3, "Use and validation of pre-existing software (PSW)")

A11.10.1 The *Safety Review* should demonstrate that the functions performed by the COTS meet all of the requirements expressed in the safety-system software requirement specification. The functional and non-functional behaviour of COTS should be clearly defined and documented using, where appropriate, a formal specification language; and the impact of the COTS on safety should be evaluated. The interfaces through which the user, or other software invoke the COTS modules should be clearly identified. A summary description of these aspects of the COTS should be provided in the *Safety Review*.

For safety systems of the highest integrity, the COTS should be subjected to the full rigor of the assessment process defined in this guide. The COTS should not be used if there is insufficient information available to perform this assessment.

It may be possible for COTS consisting of small amounts of code (e.g. compiler library routines) to determine its functionality (i.e. the full specification) and demonstrate its correctness through *reverse engineering*. If this approach is adopted then this must be justified in the *Safety Review*.

> *Note (from Reference 3, section 10.32): "reverse engineering involves applying in reverse (or inverting) the translation process. Its feasibility depends on the original translation process being well-defined and traceable, rather straight-forward and (uniquely) invertible. This inversion has been performed satisfactorily, and even mechanised, for output of generators of code from designs. The technique has also been used successfully in reconstructing source code from machine code where the source has been fairly low-level. The technique cannot be rigorously applied to reconstruct high-level language programs, for which the mapping to executable code is extremely complex."*

A11.10.2 It should be demonstrated in the *Safety Review* that the COTS functions cannot be invoked (even inadvertently, for example through erroneous inputs, interruptions, and misuses) by the safety system, by other software or by the users in ways that are different from those which have been specified and successfully tested. This may necessitate the use of pre-conditions, locking mechanisms or other protections, A11.10.3 The COTS components to be used should be clearly identified including their code version(s). (reference 8, section 1.3.3.2)

<u>For Transformation Tools</u>

(The following requirements have been extracted from reference 8, chapter 1.4, "Tools")
A11.10.4 The *Safety Review* should provide a precise definition of the transformation tools' functionality and their domain of applicability. This should cover tools such as (see reference 8, section 1.4.4.5):

- tools for editing and type-setting;
- tools for syntax-checking, data type-checking and *consistency*-checking;
- tools to prove properties of specifications and discharge proof obligations in refining specifications;
- tools to animate formal specifications;
- tools to derive usable software components automatically from formal specifications.

A11.10.5 The transformation tools maturity should be justified in the *Safety Review*.

Q11.11 Is this software supported with comprehensible manuals and/or other tools which facilitates the understanding?

<u>For COTS</u>

A11.11.1 The information required to evaluate the quality of the *COTS*, and of its assessment and development processes should be available in the *Safety Review* as required by A11.10.1. Any COTS which met these requirements will be well understood.

<u>For Transformation Tools</u>

A11.11.2 The manufacturer's quality assurance documentation for the tools used in the development of the safety system software should be reviewed in the *Safety Review*. The policy to be adopted when new versions of the tools have to be used should be precisely described, e.g. perform thorough review of tool and justify not re-applying it to all earlier software; repeat all earlier work.

A11.11.3 The integrated project support environment incorporating the tools should be described in the *Safety Review*. It should be shown how proper control and consistency is ensured. This environment should include suitable data dictionaries which check for inconsistencies in the data base, ensure the correctness of values of items entered and enable data to be interrogated. In addition, it should support the exporting and importing of data.

A11.11.4 An analysis of the transformation tools used in the software development process to identify the nature of the faults they can introduce in the target software, and their consequences should be included in the *Safety Review*. (see reference 8, section 1.4.3.1)

Q11.12 Has a wide user experience shown that this software has a high quality and reliability?

For COTS

A11.12.1 The evaluation of the quality and reliability of COTS software for safety systems should not be based on user experience. The *Safety Review* should describe how the following principles have been met.

i) For safety systems with the highest integrity requirements, COTS should be subjected to the same rigorous review (verification and validation - not of the production process) as for new software. The operational experience feedback should be seen as compensating for the lack of evidence of the quality of the development process.

ii) To be well defined the COTS should have been developed and maintained according to good software engineering practice and QA standards appropriate to its intended use. For example, a compliance analysis of the COTS design against the requirements of this guidance should be performed and reported in the *Safety Review*.

Operational Experience Feedback

A11.12.2 If credit is claimed for feedback experience(say for safety systems of lesser integrity), sufficient information on operational history and failure rates should be available. Feedback experience should be properly evaluated on the basis of an analysis of the operating time, error reports and release history of systems in operation. This feedback experience should also be based on use of the COTS under evaluation in identical operational profiles. This operating experience should be based on the last release, except where an adequate impact analysis shows that previous experience based on unchanged parts of the COTS is still valid because these parts have been unaffected by later releases. (reference 8, section 1.3.3.9) This information should be fully documented in the *Safety Review*.

A11.12.3 The information on operational history and failure rates provided for evaluation should consist of the version number of the COTS and its configuration, site information and operational profiles, demand rate and operating time, error reports and release history, and should be included in the *Safety Review*.

The site information and operational profile data should include (see reference 8, section 1.3.4.2):

- configuration of the COTS;
- functions used;
- types and characteristics of input signals, including the ranges and, if needed, rates of change;
- user interfaces;
- number of systems.

The demand rate and operating time data should include:

- elapsed time since first start-up;
- elapsed time since last release of the COTS;
- elapsed time since last severe error (if any);
- elapsed time since last error report (if any);

- types and number of demands exercised on the COTS.

The error reports should include:
- date of error, severity;
- fixes;

The release history should include:
- date and identification of releases;
- *faults* fixed, functional modifications or extensions;
- pending problems.

A11.12.4 If the available information of the type required above is not sufficient, then an analysis (risk assessment) of the impact on safety of a failure of the COTS should be performed and reported in the *Safety Review*. Special attention should be paid to possible side-effects and to failures that may occur at the interfaces between the *COTS* and the user and/or other software components. (reference 8, section 1.3.3.10)

Operating systems

A11.12.5 Only operating systems that comply with the clauses of this document should be used. . The justification for the use of the operating system should be provided in the *Safety Review*. The use of the operating system should be restricted to the indispensable functions. These functions should be identified in the *Safety Review* and should have well defined interfaces. Each particular function should be called always in the same way. Relevant and documented operating experience of the use of these functions should be available. There should be no unused code. Where this is not achieved, the *Safety Review* should provide evidence of the use of pre-conditions or interlocks or other suitable protection that will prevent the running of this unused code. The *Safety Review* should justify the precautions taken to ensure that the separation of safety and non-safety functions is not jeopardised by the use of a common operating system, other running support software or network communication software. (Extracted from reference 3, section 9.25 and 9.26).

For Transformation Tools

A11.12.6 The *Safety Review* should describe how the following principles (taken from reference 8) have been met.

i) The use of a transformation tool for safety system software without further review of its output is not acceptable unless special justification is provided. This requirement may be reduced if its output is subjected to verification or reverse engineering. (reference 8, section 1.4.3.5)

ii) A software tool should be independent from the operating system so as to minimise the consequences of having to use new versions of this system. (reference 8, section 1.4.4.4)

iii) The vendor of a tool should maintain and update the tool experience feedback and inform users of all anomalies discovered by users. (reference 8 section 1.4.3.6)

Q11.13 Has previous experience shown that the vendor produces high quality products?

A11.13.1 The *Safety Review* should contain evidence in support of the high quality of the similar products produced by all the vendors supplying software for the safety system development. It should include the name of the product and its version number, the date of its first introduction into the market place, the number of systems in the field and the total elapsed time of each, total number of software errors.

<u>Questions about System safety</u>

Comments:
The safety aspect includes both fault tolerance and actions to bring a plant into a fail- safe state in case of a failure in the system. What to look for concerning fault tolerance is which methods implemented into the system are used to work correctly even if random or systematic failures occurs. Concerning fail-safety one should look for methods implemented to detect failure, and if a failure is detected, whether the system is able to transfer into a safe state previously identified in the hazard and risk analysis.

Another aspect is operation safety. Intended operator actions, as part of regular activities, should always be performed in a safe way. There should in addition be built in a certain fault tolerance, so that the system can tolerate operator errors without jeopardising safety

Q11.14 Are methods implemented to detect failure and transfer the system into a safe state previously identified in the hazard and risk analysis?

A11.14.1 The *Safety Review* should contain a traceability matrix showing which particular software code fragment implements a particular fail-safe requirement arising from the plant hazard and risk analysis. The verification activity will demonstrate that it meets its specification.

A11.14.2 SKIFI 1998:1, Chapter 5, "Operation of Facilities", paragraph 3, "Maintenance, Surveillance and Testing" states that:

"Building components as well as other components, systems and other devices of importance for safety at a facility shall be inspected and tested on a *continuous* basis in order to control that they function in a safe manner and that there is no sign of damage."

As far as software-based safety systems are concerned, this legal requirement is fulfilled by the self-diagnostic features designed into the safety system. Hence, the *Safety Review* should describe how the software design includes the necessary self-supervision features to detect hardware faults that may occur at execution time. Software should also supervise its own control flow and data. It should be shown that the supervision features, that were not anticipated by the Software Design, have been included, and the proper requests for modification of the requirements made.

A11.14.3 SKIFS 1998:1, Chapter 5, paragraph 3 continues by stating that:

"In order to prevent abnormal events, incidents and deficiencies of importance to the safety of the facility, such parts and devices shall be maintained in accordance with special maintenance programmes which shall be documented."

The *Safety Review* should describe the means of meeting this legal requirement and should demonstrate that the tests (in combination with the automatic self-supervision) cover all postulated faults. The tests should verify periodically the basic functional capabilities of the safety system, including basic safety functions, major non-safety related functions, and special testing used to detect failures unable to be revealed by self-supervision or by alarm or anomaly indications. The testing should include applicable functional tests, instrument checks, verification of proper calibration, and response time tests. However, periodic testing should not adversely affect the intended safety system functions and, on termination of the tests, the safety system should be restored to its original status. The preferred method is to subject the software-based safety system to periodic tests by means of an automatic facility specially designed for that purpose and engineered as part of the safety system. It should be qualified to the same level as the safety system. (see reference 8, section 2.8.3.1.1)

Q11.15 Are methods implemented into the system which make it work correctly even if random or systematic failures occurs?

A11.15.1 The measures discussed in response to question Q10.7.2 of Chapter 10 will ensure that the safety system works correctly in the presence of random or systematic failures. For a safety system the safe action is to place the reactor in a safe shutdown state and hold it there. Hence, any detected failure, which might cause the system to fail to act on demand, should move towards reactor shutdown. For example, in a 2oo3 voting system any channel that has failed should be placed in a trip condition.

A11.15.2 The *Safety Review* should contain a hardware and software Failure Modes and Effects Analysis for the completed system to identify all potential failure modes and their effects on the system. The safe outcome of the failure should then be demonstrated either through design provisions within the safety system or other means. (see reference 8, section 2.2.3.6)

Q11.16 Has the system been realised in a way which minimises the likelihood that an operator action may change the system in an unsafe way?

A11.16.1 The *Safety Review* should describe how it has been ensured that no program modifications are necessary as part of the system's operation. It must be ensured that safety system software need not be altered to overcome operational or maintenance difficulties, other than on an off-line system using agreed modification procedures. Engineered facilities should be provided where there is a need to change specific system parameters such as trip settings, calibration constants etc. for operational reasons or to operate plant for maintenance purposes. The available range should be limited to values supported by the safety case. A display of current values or states of such parameters should be provided. (reference 12, section 103)

A11.16.2 SECURITY (extracted from reference 8, Chapter 1.7, "Security", sections 1.7.3 & 1.7.4)

With all safety systems there is a potential for the operator (or other persons) to perform, either intentionally or unintentionally, an action which might result in an accident due to gaining access to a part of the system for which they do not have the required. The restriction of physical access through the use of locked doors and key interlock systems has been the

traditional method, and still remains a major contributor to preventing unauthorised access to these safety-critical systems. However, with the introduction of software-based safety systems, there are now a number of security issues that are unique to the technology due to its programmability and communication facilities, and which need to be addressed.
In addition, there are other computer systems such as maintenance scheduling systems, fuel burn-up calculators and other computer codes which are used off-line in support of operational safety, computers used in support of emergency arrangements and dosimetry record systems, that could indirectly affect plant or individual health and safety, and that must be considered.

The objective of information and system security is to guarantee and preserve the dependability of safety systems by preventing security incidents or by minimising their impacts. In this context, security seeks to prevent unauthorised accesses to information, software and data in order to ensure that three attributes are met, namely:

(i) the prevention of disclosures that could be used to perform mischievous, malicious or misguided acts which could lead to an accident or an unsafe situation (confidentiality),

(ii) the prevention of unauthorised modifications (integrity),

(iii) the prevention of unauthorised withholding of information, data or resources that could compromise the delivery of the required safety function at the time when it is needed (availability).

So far as is reasonably practicable, there should be no significant increase in overall plant risk from computer-based systems important to safety in a nuclear installation due to:

- *intentional or reckless interference or misuse of these systems,*
- *well intentioned, but misguided, use by authorised persons of these systems,*
- *the security provisions themselves.*

Information can be stored in computer central memories, mass storage devices, tapes, diskettes, can be printed, and can be transmitted by various networks or movable media as tapes and diskettes. This means that there is the potential for intentional or unintentional impairments and misuses by company employees or members of the public, authorised or not. Additionally, specific threats also exist from hackers, viruses, software logic bombs, software time bombs, Trojan horses, and other similar mechanisms. Hence, the procedures and equipment for loading software into a software-based safety system and its support equipment should be included in any security evaluation.

Information has to be protected by taking into account the different storage and communication media which are used, not only at the plant but also at a supplier site.

A11.16.2.1 SECURITY

SKIFS 1998:1, Chapter 2, paragraph 5 requires that a "physical protection plan" be documented. It also requires that this plan is subjected to a *Safety Review*. For a software-based safety system, the *Safety Review* should ensure that the following provisions are met. The details of the security arrangements should be covered by a security classification that restricts the personnel who have access to these details to those with the required security clearance and authority. The details should only be available on a "need-to-know" basis.

i) As a minimum, the security provisions should incorporate the standard security measures that are regarded as good commercial practice against threats coming from unauthorised access, viruses, software logic bombs, Trojan horses, and other similar mechanisms. (reference 8, section 1.7.3.1.1)

ii) The security provisions that address the procedures and the environment of the design and development of a software-based safety system should be commensurate with the level of safety significance of the target system. (reference 8, section 1.7.3.1.2)

iii) Access should be restricted (for example through the use of password protection and key lock systems) to only those parts of a system to which a person has authority. (reference 8, section 1.7.3.1.4)

iv) A document on security should be drawn up so as to identify the security risks, threats and vulnerabilities of the safety system. This document should be used to derive the security design requirements of the computer-based safety system. (reference 8, section 1.7.3.1.5)

v) In this document (section iv above) allocation of information security responsibilities should be defined. In particular, the licensee should define security requirements that its contractors and service providers have to satisfy. (reference 8, section 1.7.3.1.5)

vi) As necessary, security incidents should be reported to the regulatory authority. (reference 8, section 1.7.3.1.6)

vii) Special security attention should be taken during vulnerable phases, for example, on the occasion of software or hardware changes. (reference 8, section 1.7.3.1.7)

Physical security measures.

viii) For safety systems, security threats should be avoided by design where possible. The safety system software (programs and fixed data, including operational data) should be held in suitable fixed read only memory. A systematic comparison of this read only memory code with the source code should be performed for the purpose of detecting unauthorised code. (reference 8, sections 1.7.3.2.1 & 1.7.3.2.2) Suitable arrangements should be made for the secure storage of copies of the software that is to be loaded in the event of a system failure. Due regard should be given to all possible means of corruption of the software. Particular attention should be paid to the secure storage of pre-programmed memory devices if they are used. Means should be provided to verify that the software has not been corrupted prior to its use both before and after loading. Automatic means are preferred. (reference 12, section 101)

ix) No inward data transmission link into the safety system should be allowed except the direct connection required by the specified local maintenance and testing equipment. These systems and other specified support systems should have restricted access, for example, through the use of passwords and key locks. (reference 8, section 1.7.3.2.3)

x) Direct links from the safety system to equipment outside the plant should be prohibited. (reference 8, section 1.7.3.2.4)

xi) All changes of constants and parameters should be automatically recorded in a secure manner for subsequent auditing. (reference 8, section 1.7.4.1)

xii) All authorised accesses should be recorded and all security attacks should be reported (automatically where possible) and investigated. (reference 8, section 1.7.4.2)

xiii) Education on the security policy should be imposed on all persons involved. (reference 8, section 1.7.4.3)

xiv) Where the safety system interfaces to computer-based support systems, for instance for the updating of calibration data or plant maintenance activities, then, as a minimum, good security practices should be observed. Examples of such practices are: user identification and password protection, the installation of virus protection software (if relevant), and the standard physical restriction of access to only authorised persons. (reference 8, section 1.7.4.4)

xv) Constant vigilance is required. The security policy should address the security issue through appropriate training programmes and specifically targeted campaigns. (reference 8, section 1.7.4.5)

xvi) For safety systems, security specialists in association with the safety specialists should evaluate the adequacy of the security provisions. (reference 8, section 1.7.4.6)

- *Q11.17 Are intended operations safe?*

A11.17.1 To ensure that the intended operator actions are safe, the *Safety Review* should demonstrate compliance with the following principles.

i) Appropriately engineered facilities should be provided to ensure that operation of any testing facilities, manually or automatically initiated, neither trigger spurious operation of tests, nor protective action nor degrade protective system reliability. Where testing causes inhibition of protective action, testing, facilities should be regarded as part of the protection system and should be designed and engineered to the same standards. (reference 12, section 50)

ii) Calibration data should be of a sufficiently high accuracy so as not to degrade the computer-based system's reliability. For safety systems, such data should be generated automatically by a system that has been developed to the same standards as the computer-based system. Where the data generation system has not been developed to this standard, the calibration data produced should be checked by a diverse method performed by an independent group. (From reference 3, section 14.7)

iii) Where data is loaded manually there should be a read-back facility that requires the operator to confirm the data entered before he/she is allowed to proceed to the next item. All entered data should be separately archived and checked by an independent party before the safety system is re-instated. (reference 8, section 2.8.3.3.1)

iv) Where the data is entered electronically, that set of data should be covered by a checksum which will enable the data to be verified. In addition, the data should be read back and verified automatically against the original. Printouts of the data entered and that held in the guardline should be provided and retained for auditing purposes. (reference 8, section 2.8.3.3.3)

v) A suitable test of the guardline concerned should be performed following the change of calibration data so as to demonstrate its correct operation. (From reference 3, section 14.9)

vi) The means of testing plant items should be engineered as part of the safety system. Such engineered test systems and their user interfaces should be designed to standards commensurate with their duty. They should provide a read-back facility, with confirmation by the operator, of the plant item selected before it is activated for test. (reference 8, section 2.8.3.5.1)

vii) It should not be possible to connect test equipment to more than one redundant train at any one time. The bypassing of actuation signals for the maintenance of output devices and components should be indicated by an alarm and be recorded as unavailable. (reference 8, sections 2.8.3.5.2 & 2.8.3.5.3)

Q11.18. Can the system tolerate operator errors without jeopardising safety?

A11.18.1 The *Safety Review* should demonstrate how it is ensured that:

i) the operator is not be able to negate correct protective action as determined by the specification;

i) test equipment activation is restrict to one plant item at a time (reference 8, section 2.8.3.5.2);

ii) a plant item is returned to its original entry status before another item is selected by the test equipment (reference 8, section 2.8.3.4.3);

iii) the bypassing of actuation signals for maintenance of any output devices and components is indicated by an alarm and that the output device or component is recorded as unavailable;

iv) the auto-test facility is run after any changes to establish that the system runs correctly;

v) the removal of a hardware unit or module from its operating position in a guardline results in an alarm in the main control room and the guardline being placed in a trip condition.

12 System validation

The objective at this milestone is to collect all information about verification and validation (V&V) activities during the development of the system, to use them as a basis for an assessment of the safety of the system. We will, however, concentrate on software related V&V. The goal at this milestone, validated system, is an expression of the degree the dependability targets required for the system is actually reached. As shown in the influence diagram in Figure9, there are three main sources of information of relevance in the validation of the system: testing, static analysis, and previous experience.

Figure 9 is the Influence net for *System validation.*

Questions about Test based assessment

Comments:

Testing means to execute a program (preferably on the target processor or a simulator) with selected test data to demonstrate that it performs its task correctly. Because of the digital nature of computer-based systems, exhaustive testing is impossible in practice, but ideally the test data should be selected so that they maximises the likelihood that all potentially residual faults are detected.

There are various strategies for testing. One is function testing, i.e. testing that all functional aspects of the requirement specification are correctly executed. This is known as 'black box' tests since it assumes no knowledge about the program structure. Typical examples are factory acceptance test (FAT). Coverage tests is a type of testing which aims at covering structural parts (branches, statements etc.) of the software and is known as 'white box' testing since it uses knowledge of the program structure to select data for optimal coverage. Stress tests aim at testing the program for how it handles abnormal data, heavy loads etc.

Another testing type is safety testing, which emphasises the safety aspects, i.e. selecting test data which simulates particular safety critical situations to show that the safety requirements are met. This test should check that particular safety enhancing facilities are executed correct. Simulation testing is a particularly important element in safety testing where the test data are generated by a simulation of the fault scenarios that the safety system is designed to protect against. Random testing is a particular form of simulation testing which is based on randomly generated data. Where the data sets are selected from the full operational input space with the operational probabilities, then a system reliability figure can be computed

A particular safety test is robustness testing, using the injection of faults during execution to check that the system behaves in a safe way.

The above testing is usually performed off-site. However, for the purposes of this assessment process we need to include the on-site activities of commissioning.

Reference 3 (paragraphs 13.1 to 13.3) provides the following explanation for the inclusion of this activity.

"Following delivery to site, the computer system has to be installed in the plant. This will be a progressive process involving the securing of the various items of equipment in their allocated locations, the connecting of power and the repeating of some of the tests performed at the manufacturer's premises in order to demonstrate that the computer system has not suffered damage during its transportation and installation. This phase is then followed by the connection of the plant cables to the computer system: these may be both communication cables and plant signals. After their connection, there will be a need to demonstrate that each cable has been connected to its correct terminals

Commissioning is defined ..... as: The process during which nuclear power plant components and systems, having been constructed, are made operational and verified to be in accordance with design assumptions and to have met the performance criteria; it includes both non-nuclear and nuclear tests. During this phase the computer-based system is progressively integrated with the 'Other Components' and other plant items ('Systems Integration'). Testing within the plant environment is an important part of the commissioning of computer-based systems. In particular, modes of operations and interactions between the computer based system and the plant which could not be readily tested at the validation stage should be tested at the commissioning stage.

Installation and commissioning activities can take place on either new plant or during a plant modification (e.g. retrofitting and upgrading). The recommendations given below apply to both cases here above."

Q12.1 Have all the following testing methods been applied?
- *Function testing*
- *Simulation testing*
- *Random testing*
- *Commissioning*
- *Coverage test as e.g. White box tests and Black box tests*
- *Stress tests*

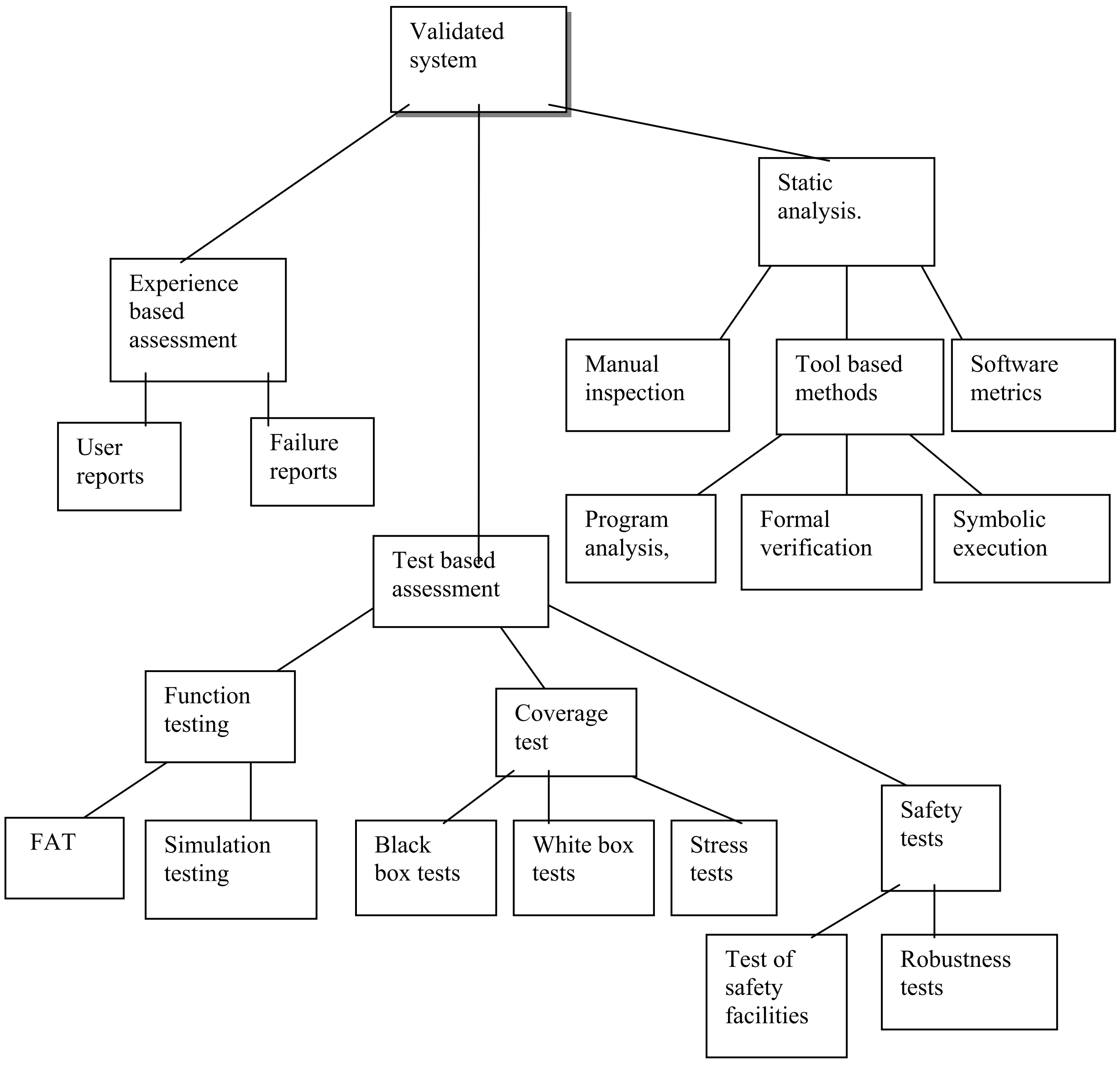


Figure 9 Influence net for milestone *System validation.*

A12.1.1 The testing strategy should be described, and the adequacy of the test coverage justified, in the *Safety Review*. It should be demonstrated that the conduct of the testing has orderly and disciplined. The ordering of the tests should either be from the top down or from the bottom up (the sequence is usually the latter, i.e. unit testing, integration testing, system testing). Where program stubs (i.e. subroutines that have only an interface with no code body) have been used it should be clearly demonstrated that the integrated entity containing the code body has eventually been properly tested. There should be a summary report of the test results, which should show that all aspects have been covered. It should be confirmed that the test results have been suitably archived for future reference.

A12.1.2 The *Safety Review* should provide evidence that the test procedures present the rationale for each test case, and enable the tracing of the test cases to the relevant source documents. The expected test results should be stated (with their method of derivation) in the test documentation prior to execution of the tests.

A12.1.3 A detailed description of a random selection of test results should be provided in the *Safety Review*. There should be a summary of which results met expectations, and which did not (anomalies); and evidence that all anomalies revealed by the various verification activities and analyses had been recorded and investigated. Evidence should be provided to show that records of all such anomalies have been archived including the means by which they were resolved.

A12.1.4 Evidence should be provided in the *Safety Review* that there is sufficient detail recorded in the test procedures to enable the tests to be repeated. This should cover, in particular, the equipment used to conduct the tests (including software test tools) which should be uniquely identified (including version and configuration, if appropriate). All outputs from the system under test should be monitored; any input or set-up parameters used should be noted in the procedures.

A12.1.5 The *Safety Review* should contain evidence that the personnel planning and conducting the tests have been given appropriate training in the use of any test tools, procedures and techniques. Personnel should be shown to be independent from the development team. And the procedures should be described which ensure that project related communication between the test team and the designers is recorded in writing. A summary report of how the test equipment has been calibrated against traceable standards should be included.

A12.1.6 The *Safety Review* should contain a summary description of the procedures for reviewing any modifications to the approved test procedures. All changes to approved test procedures should be recorded and subjected to re-approval.

A12.1.7 The set of tests should be justified in the *Safety Review*. As is stated in the IAEA guide (reference 3, section 11.12), in constructing test cases consideration should be given to:

- coverage of all requirements (including robustness tests and security features);
- coverage of full ranges (including out of range values for input signals);

- exceptions handling (e.g. demonstration of acceptable behaviour when input failure occurs);

- equivalence class partitioning and boundary conditions;

- timing related requirements (e.g. time response, input signal scanning, synchronisation);

- accuracy;

- all interfaces (e.g. hardware/ software during system integration and external interfaces during validation);

- stress and load testing (e.g. to reveal cliff edge effects);

- all modes of operation of computer system including transition between modes and recovery after power failure.

Evidence should be provided in the *Safety Review* that these points have been covered.

A12.1.8 The *Safety Review* should provide evidence that the system hardware used in the validation testing is fully representative of the final software-based safety system that will be installed at site.

A12.1.9 The *Safety Review* should provide evidence that the system operation and maintenance manuals have been validated during the testing phase.

A12.1.10 The *Safety Review* should include, as a minimum, the following information with regard to testing.

i) <u>Functional Testing (Black Box Testing)</u>

a) Using a traceability matrix linked to the test cases, it should be demonstrated that:

all functional requirements have been tested, including those that have been added during the later design phases;

all ranges of input variables;

all modes of system operation.

b) Since exhaustive testing is impractical, equivalence partitioning and boundary value analysis should be used to reduce the number of test cases required (see sections 9.3.2 and 9.3.3 of reference 4). The choice of equivalence partitions and the boundary values should be justified in the *Safety Review*.

ii) <u>Non-Functional Testing (Black Box Testing)</u>

a) Non-functional testing is a demonstration (via a traceability matrix) that all non-functional requirements implemented in software, such as numerical precision and performance, have been tested. (see reference 3, section 10.11) Evidence that this has been achieved should be provided in the *Safety Review*.

b) Performance testing (including stress testing) should cover all timing requirements, including (but not limited to) speed of response to inputs, time to detect and recover from faults, and capability to accept all specified input rates. (reference 3, section 10.12) The means by which this demonstration has been achieved should be described and justified.

iii) Structural Testing (White or Glass Box Testing)

a) Evidence should be provided that the test procedures have exercised all the interfaces (e.g. module/module, module's program/module's program, software/hardware, internal/external at system boundary) and all data passing mechanisms and the interface protocol. (reference 3, section 10.14) Again a matrix could be used with a summary description of the methodology adopted and the results.

b) The means of testing the exception conditions (e.g. divide by zero, out of range) - robustness testing - should be described and justified. This might include the use of specialist test tools (e.g. fault injection tools, in-circuit emulators). (reference 3, section 10.14)

c) The achieved targets for the structural coverage metrics of the testing (e.g. 100% statements and branch coverage, 30% linear code sequence and jumps or a basis set of test paths - see "A method of determining a basis set of paths to perform program testing", J Poole, NISTIR 5737 & "Structured testing: a testing methodology using the cyclomatic complexity metric", A H Watson & T J McCabe, NIST Special Publication 5000-235, National Institue of Standards and Technology, 1996) should be stated in the *Safety Review*, and justified. Where appropriate, the testing should be supported by tools to automatically confirm the test coverage.

v) Simulation Testing

a) The *Safety Review* should provide a summary description of the simulation testing. This should involve the exercising of all parts of the system using realistic accident scenarios. The accident scenarios should be based on an analysis of the plant transients induced by postulated initiating events. The *Safety Review* should provide evidence that the test profiles are representative of the expected plant parameter variations which would place demand on the safety system. The number of tests executed should demonstrated to be sufficient to provide confidence in the safety system's dependability. (see reference 3, sections 12.9 & 12.10)

vi) Commissioning

a) Evidence should be provided in the *Safety Review* that the safety system has not been damaged during delivery and installation at site by means of, say, a repeat of a selection of the factory acceptance tests.

b) The *Safety Review* should provide evidence to demonstrate that the commissioning has been performed to the same standards as the earlier testing discussed above. There should also be a summary report of all commissioning records and anomalies and test failures, including explanation of their resolution. The test coverage should be justified and it should be demonstrated that the external plant interfaces have been covered. The justification should include (see reference 3, section 13.11):

- the adequacy of any system integration testing performed after installation;
- the amount of repeating of the pre-installation testing (sometimes known as factory testing or acceptance testing;
- the commissioning test coverage including the tracing of tests to source requirements (safety system requirements).

c) The NPP's approach to the on-site probationary period should be described, and justified, in the *Safety Review*. The justification should demonstrate that the safety system will be capable of performing the appropriate safety duties during all phases of commissioning, particularly once radioactive material is in the reactor. The description of the probationary period should cover the operation, test and maintenance of the safety system, and should be as representative of the in-service conditions as possible. It may be appropriate to run the new system in parallel with the old system for the probationary period, i.e. until sufficient confidence has been gained in the adequacy of the new system. During this phase, the validation of operation and maintenance manuals should be completed, and the system should be subjected to routine test and maintenance activities. A log should be kept of any revealed faults and appropriate corrective action taken. (see reference 3, section 13.7)

d) The *Safety Review* should provide evidence that the commissioning team, who produce the commissioning tests, are competent and independent from the developers of the safety system. (see reference 3, section 13.9)

e) The configuration control regime for the safety system at site should be described, and justified, in the *Safety Review*; and it should be demonstrated that any changes required during this phase have been subjected to a suitable change control process (see chapter 13). (see reference 3, section 13.10)

Q12.2 Have reliability estimates been made based on any of these tests?

There has been a considerable amount of research undertaken into the estimating, based on testing, of a reliability metric for software. These result in reliability figures based on failure rates, probability of failure on demand and residual defects. Of these, the probability of failure on demand is the most important since it will have the greatest impact on safety since fault sequences will not be terminated or their effects mitigated. A measure of the failure rate is important since such failures may cause spurious trips. In the latter case the requirement should be that the safety system software is an insignificant contributor to the spurious trip rate. However, any estimate of the failure rate that did not take into account the type of failure (i.e. only those that caused a spurious trip) would be on overestimate of the safety system's failure rate. Defect density is more a measure of quality than reliability since no account is taken of the significance of the defects.

Estimating reliability based on the above tests is not a trivial exercise. Colorado State University has undertaken some work on estimating defect density from test coverage. They *suggest this method is more accurate than methods based on reliability growth models and testing time expended (see "Estimating defect density using test coverage", Y K Malaiya and J Denton, Technical Report CS-98-104, Colorado State University, CO, USA, 1998). Unfortunately, the number of defects is not a good measure of the reliability of a piece of software since there is no indication of the path in which the defect lies. For example, the path may only have a very low probability of being traversed in the lifetime of the system and may, in fact, never be traversed. Since the reliability of a software-based system is a function of the probability of defects being present in particular paths and the probability of that path being traversed then simply knowing that there are x faults remaining in the system gives no indication of the reliability.*

Software Reliability Growth Models work best when the testing is based on the operational profile of system usage (see the British Computer Society SIGIST group's draft software standard "Reliability Testing"). This means that the software is being tested as though it were in the field. Based on this approach various reliability growth models can be fitted to the data (see Appendix A3, "Reliability Growth Models", in "Software Reliability and Safety in Nuclear Reactor Protection Systems", J D Lawrence, NUREG/CR-6101, US NRC, 1993). Of note is that a reliability growth model can be specially selected and calibrated for the particular software production process, since each has its own modelling characteristics (see section 3.1, "Reliability growth assessment and prediction" in "The problems of assessing software reliability .... when you really need to depend on it", B Littlewood, Report no. CSR 874, Centre for Software Reliability). From the high-integrity-software point of view, there is, however, one fundamental drawback when considering reliability growth models for predicting reliability, namely, that the software has to contain errors in order to generate data for the models. Obviously, this is not desirable for a software-based safety system - one is looking for a quality system with, if possible, no errors. This means that for safety systems such methods are not acceptable.

The approach that is the most appropriate is to develop the software to the highest standards and subject it to a searching testing regime until it is felt that the residual errors are negligible. The system should then be subjected to a series of tests that mimic the operational profile, expecting failure-free working.

In order to measure the probability of failure on demand of a safety system, it should be subjected to a set of tests which mimic the accident scenarios (fault sequences) for the reactor. The probability of selecting a particular scenario should be based upon the estimated frequency of occurrence of that fault sequence. The statistics then allow the probability of failure on demand to be estimated based on no faults being found provided the following conditions are also met:

a) *the tests are independent;*
b) *for each test, the probability of failure is constant;*
c) *there are a large number of test runs.*

From these tests the probability of failure on demand (pfd) can be estimated at the 99% confidence level as pfd $< 10^{-k}$ *for* 4.6×10^{k} *tests (see section 9.3.7, "Random or statistical testing" in reference 4) which means that for a pfd of* 10^{-4} *there must be 46000 tests. This is a large number of tests and represents a limit on the reliability figure that can be realistically demonstrated by this method (or in fact any other method). It follows from this that only modest reliability claims should be allowed.*

These same arguments can be applied to failure rates. Taking a one-year interval where the inputs represent the normal operation under no-accident conditions, then for a probability of failure of 10^{-3} per year, the system has to run fault free for 4600 years at the 99% confidence level. Such testing is obviously not feasible; hence once again, only modest claims should be allowed. For example, intuition suggests that failure rates of once in ten years should be the limit to be applied for the safety system as a fault initiator.

A12.2.1 The *Safety Review* should describe, and justify, the system adopted for demonstrating the claimed reliability taking into account all the arguments given above. Evidence should be provided demonstrating that the method of error detection is adequate.

Q12.3 Have the safety enhancing facilities been tested?

A12.3.1 Since the "*safety enhancing facilities*" must be part of the functional requirements of the safety system, then they must be tested for an acceptable safety demonstration to be produced. The *Safety Review* should identify the features that have been included in the functional requirements to enhance safety, and should then demonstrate that these have been tested.

A12.3.2 The safety enhancing features will include:

i) the redundant and diverse channels with their voting arrangements;
ii) the human-factors interface checks for out-of-range values & re-instatement of the status of the safety system following maintenance activities;
iii) hysteresis bands on measurements to stop chatter around trip values;
iv) slugging of relay contact inputs;
v) all the fault tolerant provisions.

Q12.4 Has fault injection been used to test the robustness of the system?

A12.4.1 Since the software provided to make the safety system robust must be part of the functional requirements of the safety system, then they must be tested for an acceptable safety demonstration to be produced. The *Safety Review* should identify the features that have been included in the functional requirements to make the system robust, and should then demonstrate that these have been tested.

A12.4.2 For example, the automatic self-diagnostic features should be tested as fully as possible. This may mean testing at board and module level using special equipment such as an In-Circuit Emulator (ICE) and injecting faults into the hardware. The *Safety Review* should provide evidence that all these features have adequate test coverage in terms of all statements and all branches at least at the module level.

A12.4.3 As stated in the response to question Q12.3, all the fault tolerant features should be demonstrated in the *Safety Review* to have been tested.

A12.4.4 The robustness of the safety system should be tested by selecting combinations of inputs that are not included in its operational input space. For example,

i) instrument readings that are in the wrong range for the particular plant operating mode,

ii) higher than expected rates of change of inputs,

iii) linked parameters moving in directions relative to each other contrary to those expected,

iv) operators terminating procedures at unexpected points or attempting to perform operations that are not allowed

- all these should not result in an unsafe outcome.

The *Safety Review* should provide a comprehensive list of the particular tests performed, and demonstrate that each test has a safe outcome.

Questions about Static analysis

Comments:

It should be noted that static analysis is applied to a model of a program's behaviour; testing explores those aspects of a program's behaviour that a model does not capture. Static analysis is, therefore, a complementary verification method to testing. This includes manual inspection, like Fagan inspection, structured walkthrough etc. Manual inspection in general always used in the verification process, complementary to testing, and should be mandatory. It is also a possibility to use available tools for the analysis. There exists various program analysis tools, as e.g. reverse engineering tools. A particular type is tools used for formal verification, like program provers and model checkers. However, such tools are most effective when t a formal method have been used in the system development. Formal verification generally requires considerable expertise; and therefore, due consideration has to be given to the competence of the staff involved in this type of analysis.

A third type of tool is for symbolic execution, which means that a program is executed as an algebraic expression which may be compared with the specifications which are expressed in a formal language (formal code verification). Software analysis tools, because of their high cost of application, are not in general use, but have mostly been used on software with very high integrity requirements. But this can change in the future.

It is also a possibility to perform measures on the software according to different software metrics. However, the correlation between such software measures and software reliability is not high (see "A critique of software defect prediction methods", N Fenton & M Neil, DeVa project).

Q12.5 To which degree has manual inspection of the system been applied?

A12.5.1 The *Safety Review* should describe, and provide summary findings (referenced to the detailed reports) of, the manual techniques, such as reviews, walkthroughs, visual inspections or audits, that must be applied to all the all life-cycle phases as part of the verification activity. The whole of the documentation, software and data of the safety system should be subjected to manual inspection. The data should include any configuration, calibration and trip data used in the particular instantiation. Evidence should be provided of the traceability of the functional and non-functional requirements from the top-level specifications, through the design features, the code used to fulfil these requirements, to the tests used to demonstrate compliant behaviour. The procedures for controlling the manual inspections and the method of recording the results should be described, and justified. Where checklists are used the required response to a checklist item should be clear and not open to interpretation.

A12.5.2 The manual inspection can either be done by the supplier through an independent Verification and Validation team, and then the process reviewed by *the Safety Review* group, or, the whole operation could be done by the licensee's *Safety Review* group. In either case, the persons performing the manual inspections must have the necessary knowledge and expertise to perform the task. The manual inspection team should raise 'comments', in writing, which cover what they perceive to be anomalies (errors or shortcomings due to inconsistency or ambiguity etc.) in the documentation, code or data. These anomalies should be resolved, in writing, by the supplier's design and coding staff. The *Safety Review* should justify the level of independence of the groups to show that it is adequate.

A12.5.3 The *Safety Review* group would establish, as a minimum, that:

the design is understandable and has been decomposed into a logical hierarchy;

fail-safe and defensive programming techniques had been used;

all documents are comprehensive and follow the supplier's standards;

coding standards have been followed;

code is well structured and properly commented;

there is consistency in the data transfers between sub-systems;

redundant variables are identified;

the size of data arrays is correct;

through numerical analysis of calculations, the stability of solutions, the potential for divide by zero, numerical overflow and underflow and the correctness of any floating point operations;

where pointers are used in the code that they are not corrupted by side effects in a routine;

interfaces between software specifications and hardware specifications are correct, complete and unambiguous;

maximum response time allowances have been met.

Q12.6 Have tools been applied to analyse the software?

A12.6.1 The *Safety Review* should provide a summary description of the tools used to verify the code statically. There should, also, be a justification for the use of these tools in terms of their suitability (applicability, quality, ease of use and the skill levels of the users) for the chosen duty. A clear statement of their limitations should also be given.

A12.6.2 Of the manual inspections listed in A12.5.3, tools should be applied in the following areas:

- code structure using a control-flow analyser (see A11.3.2);
- data use using a data-use analyser;
- input/output relationships using an information-flow analyser;
- algorithmic relationships between inputs and outputs using an analyser that performs symbolic execution (see A12.8.1 for more detail).

A12.6.3 There are a number of industrial strength tools on the market that can be used to statically analyse the safety system's software, and they all usually possesses analysers to perform the tasks listed in A12.6.2. The following are examples of such tools.

Control-flow Analyser

A control-flow analyser treats the software as a directed graph and determines its reducibility as discussed in A11.3.2. It shows whether or not the software is well-structured in terms of entry and exit points, loops and branches; it also identifies unreachable code. Neither the presence of unreachable code, nor poor structure, mean that the software contains an error but the analyser highlights the potential for error.

Data-use Analyser

This particular analyser determines how variables and constants are used in the program. A variable may be used before it is initialised - this could mean that the old value stored in the computer's memory could be used erroneously. A variable may be declared but never used, or it may be assigned values at two different points in the program without the variable being used, i.e. the first assignment would be redundant. Again none of these necessarily represent program faults but indicate that there is a potential for error that needs further investigation.

Information-flow Analyser

The information-flow analyser identifies the inputs and conditional variables upon which each output depends - similar to a cause effect graph (see reference 4, section 9.3.4). Unwanted or unexpected interactions for all possible program paths are revealed.

Semantic Rule Analyser

This analyser checks that the semantic rules of the programming language have been met. For example, it checks for items such as 'aliasing', i.e. the use of the same memory location by two different variable where one value can be unintentionally overwritten by another, or potential problems where the order in which software is elaborated will determine the outcome. Both of these latter anomalies may in fact be intentional and will not cause an error.

Run-time Error Analyser

Run-time errors are those errors that cannot be easily checked by a compiler; they usually only manifest themselves when the program runs. Some analysis tools enable the potential for some of these types of errors to be detected. These include array out of range, division by zero, numerical overflow and the violation of the range of a particular variable type. This particular analysis is performed by formal proof. Certain conditions (known *as proof obligations*) have to be proved to be met by the program so as to ensure the errors do not occur. If this can be achieved, then the defensive programming techniques covering these errors should not be required. However, in a safety system it would seem prudent to incorporate the required checks and the associated code to deal with the exception.

Execution Time Analyser

Program execution time should be analysed to demonstrate the worst case timings for activities. In some cases this can be done through the use of a suitable tool. This will enable potential timing problems to be found without relying on the random interactions of programs during dynamic testing.

Q12.7 Has formal verification of the software been used?

A12.7.1 In order to formally verify the complete design, there needs to be a formal mathematical model of the requirements' specification that has been validated for consistency, completeness and correctness, using animation as appropriate. In addition, the safety properties of the model will have been proven, i.e. that the model does not have unsafe states. This formal mathematical model of the safety system's requirements will then have been developed further to produce the system and software architecture. This design process should use a combination of formal mathematics and semi-formal structured design. The demonstration of the correctness of the mapping of this structured model onto the requirements specification will, more than likely, require the use of rigorous argument rather than formal proof. The end product of the design process will be a set of communicating software modules with associated software specifications and interface definitions. The code of the software modules can be formally verified as shown in A12.7.2. The intercommunication will require formal mathematical languages that are appropriate to the temporal issues. A language such as LOTOS can be used to check the protocols between processors; and a language such as CSP can be used to check the concurrency issues involved in any real-time system. There still then has to be the demonstration that the source code correctly implements these LOTOS or CSP models of the system. This will usually require the use of suitable tools or, again, may involve rigorous argument rather than formal proof.

A12.7.2 Formal verification of the software requires that the software requirements be formally specified. Ideally, this will have been done during the design process, but as the safety case for Sizewell B's PPS demonstrates, this can be done retrospectively (see "The rigorous retrospective static analysis of the Sizewell B Primary Protection System software", N J Ward, SafeComp '93, Poznan, Poland, 27 - 29 October 1993). This formal definition of the specification usually takes the form of a set of *pre-conditions* and *post conditions* which formally describe the restrictions on the inputs to a piece of software (pre-conditions) and the range of the outputs of that software (post-conditions). In addition, there can be other intermediate conditions that must be met (usually known as *assertions*), and conditions that must always be met during a program loop (known as *loop invariats*). The quality of any formal proof depends to some extent on the strength or weakness of these conditions. For example, a program with a weak pre-condition of x=10 and a post-condition of y = 'any value' for a program 'y = f(x)' will always prove correct. For any verification to be useful, the pre- & post-conditions, the assertions and the loop invariats must be carefully selected to be representative of the specification. An additional feature regarding loops is that the loop must be demonstrated to terminate if the software is to be proven to be fully correct.

A12.7.3 The final stage is to demonstrate that the actual machine code that resides in the computer's memory is a correct translation of the source code. This verification stage can be done by means of a technique known as reverse engineering. This is not always feasible for high level languages but was performed successfully for the Sizewell B PPS (see "Demonstrating equivalence of source code and PROM contents", D J Pavey, L A Winsborrow, Fourth European Workshop on Dependable Computing, April 8-10 1992, Prague). Tool assistance is usually required to demonstrate that the source code and machine code are equivalent. The approach is to automatically produce formal models of the two sets of code and then to use a substitution technique that simplifies each model until it is possible to formally prove the equivalence, or not, as the case may be.

A12.7.4 The *Safety Review* should describe the formal verification process that has been undertaken to demonstrate the correctness of the code against the system requirements specification, the software design and the software requirements specifications. The appropriate tools should be used for this process, and the tools should be described in the *Safety Review*. The limitations of the tools and the particular method should also be documented, and a justification for the choice given.

Q12.8 Have tools for symbolic execution of the program been applied?

A12.8.1 The process of 'symbolic execution' is used to generate the mathematical relationships between inputs and outputs for the chosen item of software (this is different from the information-flow analyser where all possible paths are included - the symbolic execution process is only applied to logically feasible paths). The *Safety Review* should describe the tool that has been used to perform the symbolic execution, and should then describe the process used to determine that the mathematical relationships generated satisfy the safety system's requirements.

Q12.9 Have the programs been measured by some software metrics?

A12.9.1 The *Safety Review* should justify any software metrics used by the licensee in support of the fitness for purpose of the safety system's software. This will necessarily be in the form of research demonstrating the correlation between the metric and the software's attribute of interest. Various measures of well-structuredness are required in A10.8.5 and A11.3.2; these should be included as part of the *Safety Review*. Numbers of modules and numbers of lines of code should be included to give an indication of the size of the software problem.

A12.9.2 The *Safety Review* should contain statistics on the numbers of errors found during the independent verification and validation phases. These should be categorised in terms of their importance to safety, and grouped in terms of a specification error, a design error, a coding error, a documentation error. The means of detection of the error should be included for each. Where any safety significant errors are found, i.e. those errors that would cause the safety system to fail to perform a safety function on demand, then the *Safety Review* should describe, and justify, the measures taken to restore confidence in the final safety system.

Questions about Experience based assessment:

Comments:

A particular validation activity is required the pre-existing software, since there is often limited information about the program structure of such software. This validation is based on experience data collect about these software modules, which can be gained from user reports and documentation of failures from the development and later usage. Such data may also be used to perform some kind of probabilistic assessment about the modules

Q12.10 Are user reports available which can give an indication of the quality of the pre-existing software?

For COTS

A12.10.1 Section 10.2.7, "Operational feedback", in IAEA Technical Reports Series No. 367, "Software Important to Safety in Nuclear Power Plants" (reference 4) states that the "benefits available from operational feedback are relatively small ……. and arises from the long period of operation (beyond 10 years) that are required before the results are significant." This view is also supported by the UK Health and Safety Commission's Study Group on the Safety of Operational Computer Systems in their report "The use of computers in safety-critical applications", ISBN 0 7176 1620 7, 1998, section 10.4.1, "COTS software".

Hence, for safety systems of the highest integrity (SIL 3 or SIL 4), the only acceptable user reports would be those that demonstrated that the COTS software meets the requirements of these guidelines. It is not considered that operational experience feedback, on its own, provides a sufficiently robust demonstration for such systems, even though the documented evidence meets the requirements of A11.12.3. However, for lower integrity systems such evidence may be used.

A12.10.2 The *Safety Review* should justify the use of any pre-existing software on similar lines to the requirements expressed in this guide. There should be a summary description of the user reports that provide sufficient evidence to support the justification.

For Transformation Tools

A12.10.3 The *Safety Review* should contain a summary report, and a justification of the acceptability of any validation report for the transformation tools. The validation of the tool should have been performed by an accredited organisation. Where a validated tool is not used, the tool's output should be subjected to a rigorous demonstration of correctness. In addition, its maturity should be demonstrated by reporting, in the *Safety Review*, on the appropriate elements of A11.12.3. The tool might be considered sufficiently mature if there are, say, ten copies of the particular version out in the field, and there have been no errors reported for one year.

Q12.11 If yes, give a quality assessment?

A12.11.1 The *Safety Review* should provide a justification for the quality attributed to the pre-existing software. The quality should be based on how well the software meets the requirements expressed in these guidelines.

Q12.12 Are failure statistics from the development and later usage of the pre-existing software available?

A12.12.1 The *Safety Review* should provide a summary report of any failure statistics from the development and later usage of the pre-existing software.

Q12.13 If yes, can the reliability of the pre-existing software be assessed on the basis of these reports?

Q12.13.1 The only reliability figure that should be accepted for pre-existing software is that which has been demonstrated through rigorous statistical testing. The *Safety Review* should provide the necessary documentary evidence to support any claimed reliability figure. Reliability claims based on in-service use (operational experience feedback) should be viewed with some scepticism since reporting of the operational profile to which the software has been subjected may not be sufficiently accurate. The fault reporting regime and the operational archives will need to be reviewed in depth in the *Safety Review*, and a clear demonstration provided which shows that the two aspects have been properly controlled.

13 Maintenance and modification procedures.

Before the system can be set into operation, the maintenance and modification procedures must also be approved. The objective of the assessment at this milestone is to show that appropriate procedures for maintenance and modification during operation are made before the system is set into operation, and that safety is preserved in all actions. However, hardware maintenance will not be considered in this Guideline, only aspects which are related to software.

Figure 10 is the Influence net for *Maintenance and modification procedures.*

Questions about Preservation of safety in normal, foreseen modifications

Comments:

One can distinguish between different types of maintenance and modifications of system software during operation. One type is normal, foreseen modifications, as e.g. changing of set points or changing of calibration constants. The program system should be made so that such changes are simple to make, but as they may be safety critical, they must be verified in some way. There are alternative complementary methods to achieve this, as e.g. independent inspection, automatic plausibility checks, redundant changes, i.e. that different persons perform the changes independently in redundant channels.

Q13.1 Do the procedures require independent inspection of foreseen modifications?

A13.1.1 For safety systems, changing calibration data is the only modification that can be classed as a foreseen modification since trip settings should be held in non-volatile memory (see A11.5.5), and only be modified through the software modification procedure. The *Safety Review* should therefore describe, and justify, how it is ensured that the calibration data is of a sufficiently high accuracy so as not to degrade the software-based safety system's reliability. This might be by means of automatic generation by a system that has been developed to the same standards as the safety system. Or, where the data generation system has not been developed to this standard, the calibration data produced might be checked by a diverse method performed by an identified, independent group. This diverse check may be either manual or automatic but must be controlled by suitable procedures that ensure maximum diversity. It should be demonstrated in the S*afety Review* that the combination of the principal and the diverse means of generating the calibration data form a sufficiently high integrity process so as not to degrade the safety system. Verification checkpoints should be introduced at convenient points in the calculation process whether it is automatic or manual. All calculations should be documented and retained for future audit.

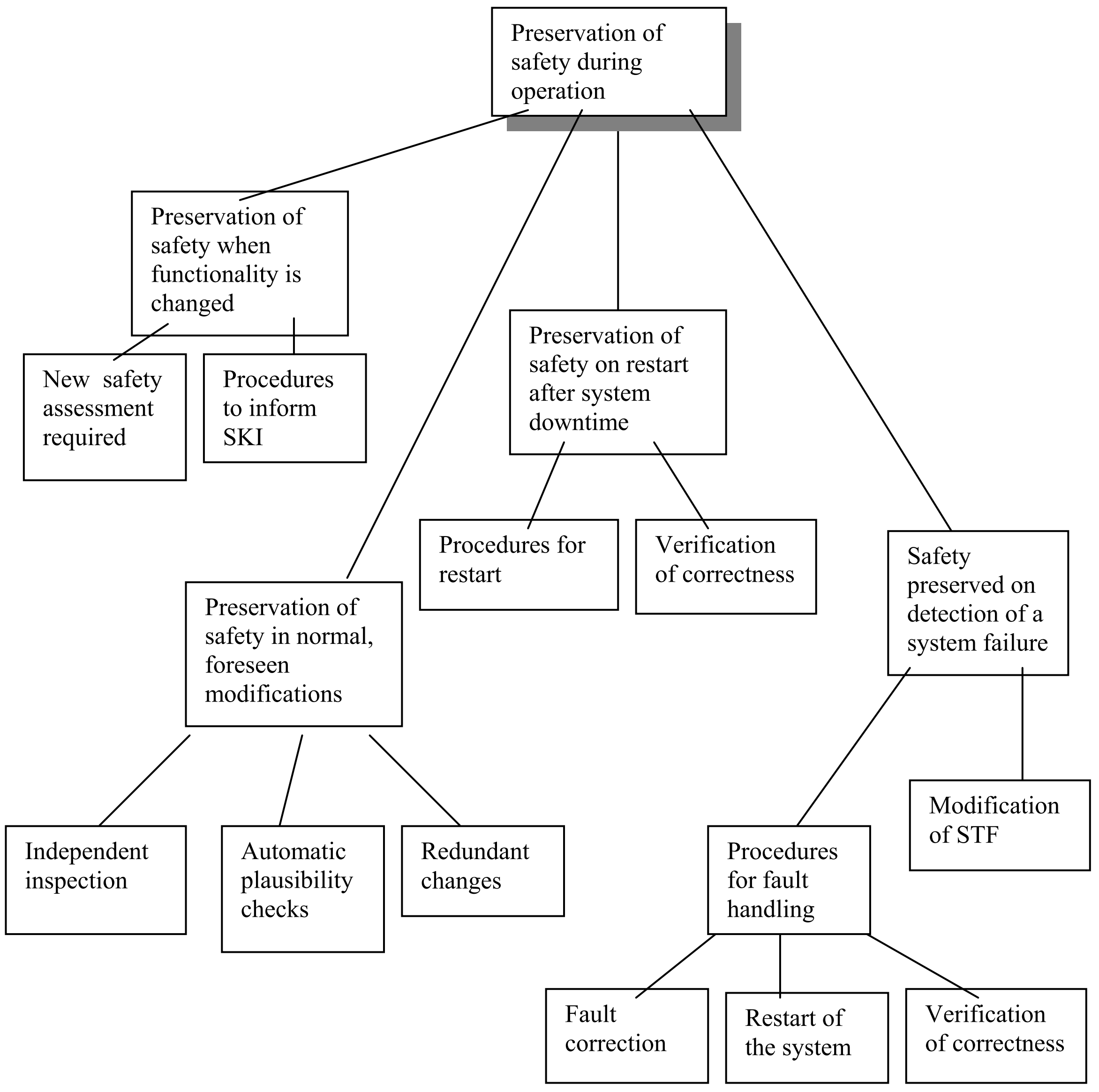

Preservation of safety during operation
Preservation of safety when functionality is changed
New safety assessment required
Procedures to inform SKI
Preservation of safety on restart after system downtime
Procedures for restart
Verification of correctness
Safety preserved on detection of a system failure
Preservation of safety in normal, foreseen modifications
Independent inspection
Automatic plausibility checks
Redundant changes
Modification of STF
Procedures for fault handling
Fault correction
Restart of the system
Verification of correctness

Figure 10 Influence net for milestone *Maintenance and modification procedures.*

A13.1.2 Because of the potential for operator error whilst manually entering data, each data item entered should be confirmed to be correct, using an engineered read-back facility, before proceeding to the next item. On completion of the data entry, the entered data should be archived and checked by an independent party before the safety system is re-instated. The *Safety Review* should describe the process for manually entering data, and justify any non-compliance with this requirement.

A13.1.3 The *Safety Review* should demonstrate that the engineered facilities provided for making foreseen modification to the calibration data are either of the same integrity as the safety system, or that the safety system software and data cannot be corrupted by the loading/display software. This latter might be demonstrated by means of a software hazard analysis.

Q13.2 Are automatic plausibility checks of foreseen modifications implemented?

A13.2.1 Automatic plausibility checks reduce the probability of the integrity of the safety system being reduced through human error. Therefore, the *Safety Review* should demonstrate how the following requirements (taken from references 3 & 8) are met, or provide a justification for any alternative solution.

i) The degree of variability of the calibration data should be limited, through the software, to the range justified in the plant safety analysis. (reference 3, section 15.6)

ii) Where the calibration data is entered electronically, that set of data should be covered by a checksum that will enable the data to be verified. In addition, the data should be read back and verified automatically against the original. (reference 8, section 2.8.3.3.3)

Q13.3 Is some kind of diversity required when the same changes are made in redundant channels?

A 13.3.1. Since the safety system contains identical redundant channels, in say a 2-out-of-4 voting configuration, there will not normally be a diverse means of entering the required data. However, different maintenance teams entering the changes in each of the redundant channels would go some way to providing a form of diversity, but this may not always be reasonably practicable. The *Safety Review* should describe how it is ensured that the data receives the appropriate checking to maintain the safety system's integrity level. As a minimum, the changes should be installed one redundant channel at a time so as to avoid common mode failure - this approach obviously provides a level of diversity since, only on completion of the changes, are all channels identical.

<u>Questions about Preservation of safety when functionality is changed</u>

Comments:

Another type of modification is when the functionality of the system is changed, and a reprogramming of the software is required. This may occur when changes which influence the system are made in the plant, or when modifications are identified through operating experience, e.g. improvement of the man-machine interface. In general, a new safety assessment is required and SKI should be contacted when such changes are made. The procedures should show how this shall be proceeded.

Q13.4 Do the procedures contain detailed requirements on a new safety assessment after reprogramming?

A13.4.1. Before a modification can be introduced, SKIFS 1998: 1 requires in Chapter 4, paragraph 6, "Modifications" that any engineering or organisational modifications to an NPP which can affect the conditions specified in the Safety Report, shall be reviewed in accordance with paragraph 3 of SKIFS 1998:1. The same review process applies to any essential modifications to the report. Any software modification to a safety system is a potential safety hazard since the modification might be either incorrectly conceived or implemented. The licensee's software modification procedures for a software-based safety system must contain a requirement to conduct the two *Safety Reviews* set out in paragraph 3 prior to introducing the modification. From this it follows that the NPP's procedures must contain detailed requirements for a new safety assessment after reprogramming.

Q13.5 Do the procedures contain requirements to inform SKI after reprogramming?

A13.5.1. As already stated in the previous response, before a modification can be introduced, SKIFS 1998: 1 requires in Chapter 4, paragraph 6, "Modifications" that SKI be notified of any engineering or organisational modifications to an NPP which can affect the conditions specified in the *Safety Report* as well as essential modifications to the report. Any software modification to a safety system is a potential safety hazard since the modification might be either incorrectly conceived or implemented. The licensee's software modification procedures for a software-based safety system must contain a requirement to inform SKI prior to introducing the modification. For extensive, safety significant modifications to software-based, safety systems, it is recommended that an early, preliminary change proposal be submitted to SKI to ensure that the opportunity is available for them to consider any "additional or other requirements or conditions" that might apply (failure to do so would, of course, be at the licensee's own commercial risk). Chapter 4 of these guidelines gives further guidance on the approach to modifications.

Questions about Preservation of safety on restart after system downtime

Comments:

A type of normal action, which is not actually a modification, but which may have serious consequences if it is not performed correct, is the restart of the software if the system for some reason has been shut down. This includes procedures for how the restart shall be made, e.g. a reload of the software if that is necessary, as well of methods, automatic and/or manual, to verify the correctness of the restart.

Q13.6 Are there clear procedures for how to restart after system downtime?

A13.6.1. The *Safety Review* should contain a summary description of the procedures for restarting the safety system after system downtime (this will usually apply to only one channel of a redundant system). Reference should be made to the specific procedures. Details of a review of these procedures by the NPP users should be included in the *Safety Review*. This review should provide evidence that they were able to follow and implement the procedures without the aid of the authors.

A13.6.2. One of the requirements for software-based safety systems is that the software, the trip settings and calibration data be held in non-volatile memory (see A11.5.5) so that the system can be re-started after downtime. Hence, the software will not need to be reloaded. The *Safety Review* should demonstrate that the initialisation of the system has been designed to require the minimum of manual intervention. For example, the outputs of the safety system should not latch (the latching being provided by the plant item). Hence, there will be no need to reset the safety system's outputs following the return to a safe state after an initiating event. It should be pointed out, however, that the individual safety actuation system items should latch and not automatically reset. A positive decision must be made to reinstate these items once it has been established that it is safe to do so.

Q13.7 Do the procedures require verification that the restart is correctly made?

A13.7.1 These restart procedures described in the *Safety Review* should require that a full periodic test be performed on the safety system before it is placed into operation again. The operator should confirm, and record that the safety system is responding and that the information produced on displays and screens is properly refreshed and updated.

<u>*Questions about Preservation of safety on detection of a system failure*</u>

Comments:

Particular emphasis must be put on procedures concerning detection of a system failure. This may be a hardware failure or caused by a software fault. In the latter case, there should be a procedure for the process of correcting the fault, restart the system and verify the correctness of the repaired system.

Another aspect to consider in this respect is the Safety Technical Specifications (Säkerhets-Tekniska Föreskrifter STF). In general, the STFs specify limitations in the plant operation when one or two redundant channels in a safety system are out of operation. However, these limitations are based on the assumption that the redundant channels fail independently, and if the failure is caused by a fault in a software module which is identical in more than one channel, this assumption is not valid. A safety system with redundant channels all containing identical software has the potential to fail totally, i.e. all channels fail almost simultaneously. Where this software failure is detected by the automatic self-diagnostic provisions, the reactor will be tripped and a series of alarms will alert the operators to the event. The more problematical situation is where the software fault is not detected - the more common of the two, but of low probability for a SIL 3 or 4 system. The first manifestation of this type of fault would be a failure to act in the event of a fault sequence. The STFs should be modified to cope with this situation. SKIFS 1998:1, Chapter 5, paragraph 1 requires that these modifications be subjected to a safety review in accordance with Chapter 4, paragraph 3 of that document, and SKI must be notified before they may be applied in the NPP. The Inspectorate may then chose to further modify these Technical Specifications or procedures.

Q13.8 Are there procedures for how software faults shall be corrected?

A13.8.1. SKIFS 1998:1, Chapter 2, paragraph 3 requires that the nuclear facility shall have

> *"establish documented guidelines for how safety shall be maintained at the facility as well as ensure that the personnel performing duties which are important to safety are well acquainted with the guidelines."*

To fulfil the requirement of *maintaining safety,* the NPP must have adequate procedures for planning, and controlling, changes to the installed software-based safety system. These should cover:

- normal foreseen modifications such as calibration data;
- modifications to the actual software and fixed data (trip settings, array sizes, input & output addresses etc).

The personnel involved in making these changes must be thoroughly familiar with the content of these procedures so that the risk of error is reduced to a level commensurate with the integrity of the safety system. In particular, once there is significant nuclear material in the reactor, only one software change at a time should be made on the safety system. Also, the software should be installed one redundant channel at a time so as to avoid common mode failure. The reactor should be tripped during the installation of the software unless a written justification to make the change with the reactor at power has been produced. (see reference 8, section 2.7.4.3)

A13.8.2. The *Safety Review* should provide a summary description of the software procedures provided for changing the software to correct software faults, or to make improvements or meet environmental changes - these should meet the requirements of SKIFS 1998:1, Chapter 4, para 6. These should apply to all element of the system, including the documentation. There should also be a suitable configuration management system in place at the NPP so that there is proper control of the software versions during a change.

A13.8.3 The procedures should not allow modification of the software, and fixed data, of the safety system during on-line operation; and no equipment should be provided for this purpose.

A13.8.4. Part of these procedures should be an adequate problem reporting and tracking procedure. This should include a review process by staff with sufficient seniority and expertise to ensure that problems are properly reviewed and the reason for their lack of earlier detection documented. Any generic concerns should be rectified and a report produced for suitable dissemination. Proposed modifications should then be assigned a sponsor.

A13.8.5. For each of the proposed changes the procedures should ensure that at least the following information is provided (reference 3, section 15.9 and reference 8, Chapter 2.7, "Change control and configuration management"):

- an identifier for the modification;
- the reason for the modification;
- a functional description of the modification and its technical feasibility. Its impact on the *Safety Report* should be described (this should include a review of the

Safety Analysis demonstrating the impact of the change on all plant items and the software itself);

- a detailed description of the software modification which includes an impact analysis which covers the full extent of the proposed change[3]. The impact analysis should contain for each and every software change:
- a short description of the change
- a list of software parts (modules, subroutines and procedures) affected by the change, including the interacting parts that are not subject to the actual change
- the effect on non-functional properties
- data items (with their locations and scope) affected by the change
- any new data items introduced again with location and scope;
- evidence that the change has been implemented to the same standards as the original software and that the verification and validation has also been to these standards;

- a justification of the proposed installation method;
- all test specifications and reports, including the site test report.

A13.8.6 The procedures should ensure that all modification documents are dated, numbered and filed in the software modification control history. The modification history should provide details of all modifications and changes including references to the modification/change request, impact analysis, verification and validation of data and results, and to all documents affected by modifications and change activities.

A13.8.6 Only software approved for release should be installed at site. Before the software is installed, and brought into service, *Safety Reviews* (as per SKIFS 1998: 1 chapter 4, para 3) of the modification should be undertaken to demonstrate that plant safety is not adversely affected by the changes. These reviews should be by suitably qualified and experienced staff, e.g. suppliers, system designers, system analysis, plant and operational staff. SKI should also be notified (see SKIFS 1998:1, Chapter 4).

Q13.9 Are there procedures for how the system shall be restarted after software faults has been corrected?

A13.9.1 The *Safety Review* should provide a summary description, with a justification for their adequacy, of the restart procedures for restarting the safety system after a software fault has been corrected.

[3] *The configuration management system should contain list of items, and their changes, that were included in previous baselines. These lists should include names of individuals and organisations to whom configuration items have been distributed.*

A13.9.2 These procedures should meet, but not be limited to, the following requirements.

i) Prior to loading the safety system software, the presence of the correct items and version should be confirmed by means of a configuration audit. The loaded software should be automatically verified as uncorrupted in the memory. (see reference 8, section 2.7.3.4.5)

ii) Where the fault was in a non-safety function, a justification might be made for loading the new software whilst the reactor is at power. This would be done one channel at a time. In the initialisation phase, and before the safety system was in operation, any parameters that depended on specific reactor operating conditions should be loaded. The safety system channel being corrected would then be allowed to go into service. There would then follow a probationary period when there was heightened vigilance in case there was a problem with the correction. The other channels would then be updated progressively in the same manner. (See also reference 8, section 2.8.2.5 for other maintenance activities that might be done at power.)

iii) Where a safety function has been modified, the change should be made with the reactor safely shutdown. The modification should be made on all channels. The procedure to be followed will be that of a normal reactor start-up. Again there should be a probationary period with extra vigilance during this time.

Q13.10 Do the procedures require verification of the correction?

A13.10.1 As can be seen in A13.8.5 the modification should be subjected to the same level of verification and validation as the original system. The procedures must, therefore, require verification of the correction.

A13.10.2 In order to demonstrate the safety system's correct operation before its re-instatement, the procedures should require that a full test, using the safety system's automatic test facility, be performed on all the redundant channels. The unmodified test facility should be run to demonstrate either that the fault has been removed, if there has been no change in functionality; or, where a change in functionality has been implemented, the unmodified test facility will highlight the change in functionality. This change should be capable of explanation, if the correction has been properly implemented. The automatic test facility should then be upgraded to reflect the change. A further test will confirm that this has been implemented correctly.

Q13.11 Are the STF modified to cope with the aspect that identical software modules in redundant channels may contain identical faults?

A13.11.1 The *Safety Review* should provide an analysis of the diverse provisions (usually hardwired) for dealing with a total loss of the safety system due to software common cause failure. These diverse provisions should trip the reactor, either automatically or manually, in the event of a potential accident, and there should be adequate hard-wired controls to bring the reactor to a safe shutdown state. The STF should provide the necessary information required to determine that the software-based, safety system had failed to act, and there should be identified procedures for dealing with such an event. These procedures should cover manual tripping and all operator actions required to bring the reactor to a safe shutdown, and to hold it in that state.

A13.11.2 The procedures should indicate that the event must be reported to SKI without delay (see SKIFS 1998:1, Chapter 7, paragraph 1) since it is a failure of protection occurring the same time as an incident, i.e. a Category 1 event (see SKIFS 1998:1, Appendix 1) - unless it can be clearly demonstrated that there was not a failure in the software. For example, the failure might be due to a total loss of supplies to the safety system, for which there are demonstrable procedures for handling. Where there is not a clear case for blaming a hardware failure which has been covered by the STF and associated procedures, then software failure must be assumed. For software-based safety systems, an incorrectly conceived or implemented modification might have a significant impact on safety; therefore, the reactor should not be brought back into operation until the software fault has been corrected, unless a satisfactory justification has been approved by SKI, as required by SKIFS 1998:1.

www.ski.se

STATENS KÄRNKRAFTINSPEKTION
Swedish Nuclear Power Inspectorate

POST/POSTAL ADDRESS SE-106 58 Stockholm
BESÖK/OFFICE Klarabergsviadukten 90
TELEFON/TELEPHONE +46 (0)8 698 84 00
TELEFAX +46 (0)8 661 90 86
E-POST/E-MAIL ski@ski.se
WEBBPLATS/WEB SITE www.ski.se