

Safety activities during early software project phases

Jon Arvid Børretzen, Tor Stålhane, Torggrim Lauritsen, and Per Trygve Myhrer
*Department of Computer and Information Science,
Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
Email: borretze@idi.ntnu.no*

Abstract

This paper describes how methods taken from safety-critical practises can be used in development of business-critical software. The emphasis is on the early phases of product development, and on use together with the Rational Unified Process. One important part of the early project phases is to define safety requirements for the system. This means that in addition to satisfying the need for functional system requirements, non-functional requirements about system safety must also be included. By using information that already is required or produced in the first phases of RUP together with some suitable “safety methods”, we are able to produce a complete set of safety requirements for a business-critical system before the system design process is started.

1. Introduction

Software systems play an increasingly important role in our daily lives. The technological development has led to the introduction of software systems into an increasing number of areas. In many of these areas we become dependent on these systems, and their weaknesses could have grave consequences. There are areas where correctly functioning software is important for the health and well-being of humans, like air-traffic control and in health systems. There are, however, other systems that we also expect and hope will run correctly because of the negative effects of failure, even if the consequences are mainly of an economic nature. This is what we call **business-critical** systems, and business-critical software. The number of areas where functioning software is at the core of operation is steadily increasing. Both financial systems and e-business systems are relying on increasingly larger and more complex software systems. In order to increase the quality and efficiency of such products we need methods, techniques and processes specifically aimed at improving the development, use and maintenance of this type of software.

In this paper, we will discuss methods that can be used together with Rational Unified Process in the early parts of a development project. These methods are Safety Case, Preliminary Hazard Analysis and Hazard and Operability Analysis. Our contribution is to combine these methods into a comprehensive method for use early in the development of business-critical systems.

1.1 BUCS

The BUCS project is a research project funded by the Norwegian Research Council (NFR). The goal of the BUCS project is to help developers, users and customers to develop software that is safe to use. In a business environment this means that the system seldom or never behaves in such a way that it causes the customer or the customer’s users to lose money or important information. We will use the term “business-safe” for this characteristic.

The goal of the BUCS project is not to help developers to finish their development on schedule and to the agreed price. We are not particularly interested in delivered functionality

or how to identify or avoid process and project risk. This is not because we think that these things are not important – it is just that we have defined them out of the BUCS project.

The BUCS project is seeking to develop a set of integrated methods to improve support for analysis, development, operation, and maintenance of business-critical systems. Some methods will be taken from safety-critical software engineering practices, while others will be taken from general software engineering. Together they are tuned and refined to fit into this particular context and to be practical to use in a software development environment. The research will be based on empirical studies, where interviews, surveys and case studies will help us understand the needs and problems of the business critical software developers.

Early in the BUCS project, we conducted a series of short interviews with eight software developing companies as a pre-study to find some important issues we should focus on [Stålhane03]. These interviews showed us that many companies used or wanted to use RUP or similar processes, and that a common concern in the industry was lack of communication, both internally and with the customers. With this basis, the BUCS project has decided to use RUP as the environment for our enhanced methods, and the methods used will be helpful in improving communication on requirements gathering, implementation and documentation in a software development project. Adaptation of methods from safety-critical development has to be done so that the methods introduced fit into RUP and are less complicated and time consuming than when used in regular safety-critical development.

That a system is business-safe does not mean that the system is error free. What it means is that the system will have a low probability of causing losses for the users. In this respect, the system characteristic is close to the term “safe”. This term is, however, wider, since it is concerned with all activities that can cause damage to people, equipment or the environment or severe economic losses. Just as with general safety, business-safety is not a characteristic of the system alone – it is a characteristic of the system’s interactions with its environment.

BUCS is considering two groups of stakeholders and wants to help them both.

- The customers and their users. They need methods that enables them to:
 - Understand the dangers that can occur when they start to use the system as part of their business.
 - Write or state requirements to the developers so that they can take care of the risks incurred when operating the system – product risk.
- The developers. They need help to implement the system so that:
 - It is business-safe.
 - They can create confidence by supporting their claims with analysis and documentation.
 - It is possible to change the systems so that when the environment changes, the systems are still business-safe.

This will **not** make it cheaper to develop the system. It will, however, help the developers to build a business-safe system without large increases in the development costs.

Why should developing companies do something that costs extra – is this a smart business proposition? We definitively mean that the answer is “Yes”, and for the following reasons:

- The only solution most companies have to offer to customers with business-safety concerns today is that the developers will be more careful and test more – which is not a good enough solution.

- By building a business-safe system the developers will help the customer achieve efficient operation of their business and thus build an image of a company that have their customers' interest in focus. Applying new methods to increase the products' business-safety must thus be viewed as an investment. The return on the investment will come as more business from large, important customers.

2. The Rational Unified Process

The Rational Unified Process (RUP) is a software engineering process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget.

RUP is developed and supported by Rational Software [Rational]. The framework is based on popular development methods used by leading actors in the software industry. RUP consists of four phases; inception, elaboration, construction and transition. The BUCS project has identified the three first phases as most relevant to our work, and will make proposals for introduction of safety methods for these phases. In this paper, we will concentrate on the **inception** phase.

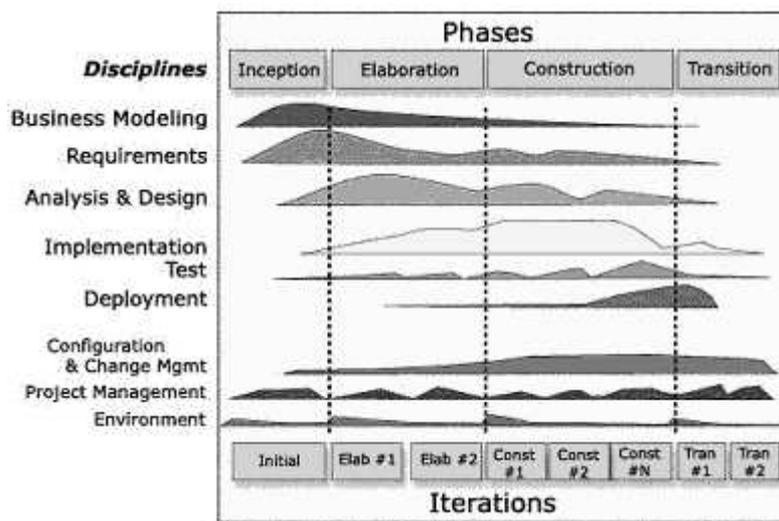


Figure 1 - Rational Unified Process; © IBM [Rational]

Figure 1 shows the overall architecture of the RUP, and its two dimensions:

- The horizontal axis which represents time and shows the lifecycle aspects of the process as it unfolds
- The vertical axis which represents disciplines and group activities to be performed in each phase.

The first dimension represents the dynamic aspect of the process as it is enacted, and is expressed in terms of phases, iterations, and milestones. The second dimension represents the static aspect of the process: how it is described in terms of process components, disciplines, activities, workflows, artefacts, and roles [Kroll03] [Krutchen00]. The graph shows how the emphasis varies over time. For example, in early iterations, we spend more time on requirements, and in later iterations we spend more time on implementation.

The ideas presented in this paper are valid even if the RUP process is not used. An iterative software development process will in most cases be quite similar to a RUP process in broad

terms, with phases and where certain events, artefacts and actions exist. Some companies also use other process frameworks that in principle differ from RUP mostly in name. Therefore, it is possible and beneficial to include and integrate the safety methods we propose into any iterative development process.

2.1 Inception

Early in a software development project, system requirements will always be on top of the agenda. In the same way as well thought-out plans are important for a system in general, well thought-out plans for system safety are important when trying to build a correctly functioning, safe system. Our goal is to introduce methods that are helpful for producing a safety requirements specification, which can largely be seen as one type of non-functional requirements. However, safety requirements also force us to include the system's environment. In RUP, with its use-case driven approach, this process can be seen as analogous to the process of defining general non-functional requirements, since use-case driven processes are not well suited for non-functional requirements specification. Because the RUP process itself does not explicitly command safety requirements in the same way it does not command non-functional requirements, other methods have to be introduced for this purpose. On the other hand, the architecture-centric approach in RUP is helpful for producing non-functional requirements, as these requirements are strongly linked to a system's architecture. Considerations about system architecture will therefore influence non-functional and safety requirements.

Although designing safety into the system from the beginning (upstream protection) may incur some design trade-offs, eliminating or controlling hazards may result in lower costs during both development and overall system lifetime, due to fewer delays and less need for redesign [Leveson95]. Working in the opposite direction, adding protection features to a completed design (downstream protection) may cut costs early in the design process, but will increase system costs, delays and risk to a much greater extent than the costs owing to early safety design.

The main goal of the inception phase is to achieve a common understanding among the stakeholders on the lifecycle objectives for the development project [Krutchen00]. You should decide exactly what to build, and from a financial perspective, whether you should start building it at all. Key functionality should be identified early. The inception phase is important, primarily for new development efforts, in which there are significant project risks which must be addressed before the project can proceed. The primary objectives of the inception phase include (from [Kroll03] [Krutchen00]):

- Establishing the project's software scope and boundary conditions, including an operational vision, acceptance criteria and what is intended to be included in the product and what is not.
- Identifying the critical use cases of the system, the primary scenarios of operation that will drive the major design trade-offs. This also includes deciding which use cases that are the most critical ones.
- Exhibiting, and maybe demonstrating, at least one candidate architecture against some of the primary scenarios.
- Estimating the overall cost and schedule for the entire project (and more detailed estimates for the elaboration phase that will immediately follow).
- Assessing risks and the sources of unpredictability.
- Preparing the supporting environment for the project.

3. Safety methods introduced by BUCS

Early in a project's life-cycle, many decisions have not yet been made, and we have to deal with a conceptual view or even just ideas for the forthcoming system. Therefore, much of the information we have to base our safety-related work on is at a conceptual level. The methods we can use will therefore be those that can use this kind of high-level information, and the ones that are suited to the early phases of software development projects.

We have identified five safety methods that are suitable for the inception phase of a development project. Two of them, Safety Case and Intent Specification, are methods that are well suited for use throughout the development project [Adelard98] [Leveson00], as they focus on storing and combining information relevant to safety through the product's life-cycle. The other three, Preliminary Hazard Analysis, Hazards and Operability Analysis and Event Tree Analysis are focused methods [Rausand91] [Leveson95], well suited for use in the inception phase, as they can be used on a project where many details are yet to be defined. In this paper, the Safety Case, Preliminary Hazard Analysis and Hazard and Operability Analysis methods are used as examples of how such methods can be used in a RUP context.

When introducing safety related development methods into an environment where the aim is to build a business-safe system, but not necessarily error-free and completely safe, we have to accept that usage of these methods will not be as stringent and effort demanding as in a safety-critical system. This entails that the safety methods used in business-critical system development will be adapted and simplified versions, in order to save time and resources.

3.1 Safety Case

A safety case is a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment [Adelard98] [Bishop98]. The safety case method is a tool for **managing safety claims**, containing a reasoned argument that a system is or will be safe. It is manifested as a collection of data, metadata and logical arguments. The main elements of a safety case are shown in Figure 2:

- Claims about a property of the system or a subsystem (Usually about safety requirements for the system)
- Evidence which is used as basis for the safety argument (Facts, assumptions, sub-claims)
- Arguments linking the evidence to the claim
- Inference rules for the argument

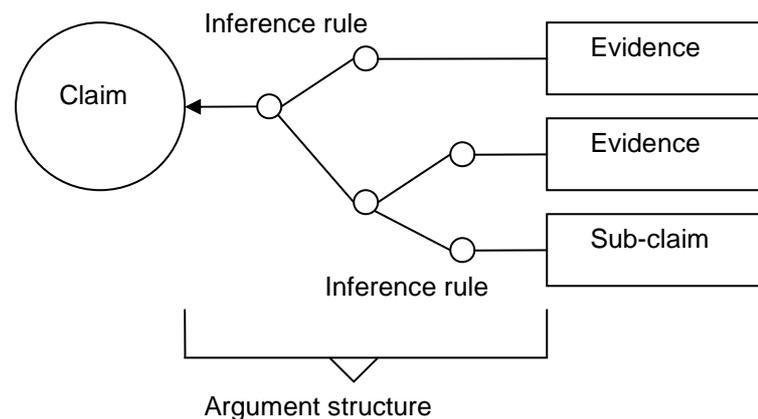


Figure 2 – How a safety case is built up

The arguments can be:

- *Deterministic*: Application of predetermined rules to derive a true/false claim, or demonstration of a safety requirement.
- *Probabilistic*: Quantitative statistical reasoning, to establish a numerical level.
- *Qualitative*: Compliance with rules that have an indirect link to the desired attributes.

The safety case method can be used throughout a system's life-cycle, and divides a project into four phases: Preliminary, Architectural, Implementation, and Operation and Installation. This is similar to the phases of RUP, and makes it reasonable to tie a preliminary safety case to the inception phase of a development project. The development of a safety case does not follow a simple step by step process. The main activities interact with each other and iterate as the design proceeds and as the level of detail in the system design increases. This also fits well with the RUP process.

The question the safety case documents will answer is in our case "*How will we argue that this system can be trusted?*" The safety case shows how safety requirements are decomposed and addressed, and will provide an appropriate answer to the question. The characteristics of the safety case elements in the inception phase are:

1. **Establish the system context**, whether the safety case is for a complete system or a component within a system.
2. **Establish safety requirements** and attributes for the current level of the design, and how these requirements and attributes are related to the system's safety analysis.
3. **Define important operational requirements** and constraints such as maintenance levels, time to repair and issues related to the operating environment.

3.2 Preliminary Hazard Analysis and Hazard and Operability Analysis

Preliminary Hazard Analysis (PHA) is used in the early life cycle stages to identify critical system functions and broad system hazards. The identified hazards are assessed and prioritized, and safety design criteria and requirements are identified. A PHA is started early in the concept exploration phase so that safety considerations are included in tradeoff studies and design alternatives. This process is iterative, with the PHA being updated as more information about the design is obtained and as changes are being made. The results serve as a baseline for later analysis and are used in developing system safety requirements and in the preparation of performance and design specifications. Since PHA starts at the concept formation stage of a project, little detail is available, and the assessments of hazard and risk levels are therefore qualitative. A PHA should be performed by a small group with good knowledge about the system specifications.

Both Preliminary Hazard Analysis and Hazard and Operability Analysis (HazOp) are performed to identify hazards and potential problems that the stakeholders see at the conceptual stage, and that could be created by the system after being put into operation. A HazOp study is a more systematic analysis of how deviations from the design specifications in a system can arise, and whether these deviations can result in hazards. Both analysis methods build on information that is available at an early stage of the project. This information can be used to reduce the severity or build safeguards against the effects of the identified hazards.

HazOp is a creative team method, using a set of guidewords to trigger creative thinking among the stakeholders and the cross-functional team in RUP. The guidewords are applied to all parts and aspects of the system concept plan and early design documents, to find possible deviations from design intentions that have to be handled. Examples of guidewords are MORE and LESS. This will mean an increase or decrease of some quantity. For example, by using the "MORE" guideword on "a customer client application", you would have "MORE customer client applications", which could spark ideas like "How will the system react if the servers get swamped with customer client requests?" and "How will we deal with many different client application versions making requests to the servers?" A HazOp study is

conducted by a team consisting of four to eight persons with a detailed knowledge of the system to be analysed.

The main difference between a HazOp and a PHA is that PHA is a lighter method that needs less effort and available information than the HazOp method. Since HazOp is a more thorough and systematic analysis method, the results will be more specific. If there is enough information available for a HazOp study, and the development team can spare the effort, a HazOp study will most likely produce more precise and more suitable results for the safety requirement specification definition.

4. Integration: Using safety methods in the RUP Inception phase

In the inception phase we will focus on understanding the overall requirements and scoping the development effort. When a project goes through its inception phase, the following artifacts will be established/produced:

- **Requirements, leading to a System Test Plan**
- **Identification of key functionality**
- **Proposals for possible solutions**
- Vision documents
- Internal business case
- Proof of concept

The artifacts in bold are the ones that are interesting from a system-safe point of view, and the fact that the RUP inception phase requires development teams to produce such information eases the introduction of safety methods into the process. Because of RUP's demands on information collection, using these methods do not lead to extensive extra work for the development team.

By using the safety methods we have proposed, we can produce safety requirements for the system. These are high-level requirements, and must be specified before the project goes from the inception to the elaboration phase. When the project moves on from the inception to the elaboration phase, identification of the business-critical aspects should be mostly complete; and we should have high confidence in having identified the requirements for those aspects.

The safety work in the project continues into the elaboration phase, and some of the methods, like Safety Case and Intent Specification will also be used when the project moves on to this phase.

4.1 Software Safety Case in a RUP context

According to [Bishop98], we need the following information when producing a safety case:

- Information used to construct the safety argument
- Safety evidence

As indicated in 3.1, to implement a safety case we need to:

- make an explicit set of claims about the system
- produce the supporting evidence
- supply a set of safety arguments linking the claims to the evidence, shown in Figure 2
- make clear the assumptions and judgements underlying the arguments

The safety case is broken down into claims about non-functional attributes for sub-systems, such as reliability, availability, fail-safety, response time, robustness to overload, functional correctness, accuracy, usability, security, maintainability, modifiability, and so on.

The evidence used to support a safety case argument comes from:

- The design itself
- The development processes
- Simulation of problem solution proposals
- Prior experience from similar projects or problems

Much of the work done early in conjunction with safety cases tries to identify possible hazards and risks, for instance by using methods like Preliminary Hazard Analysis (PHA) and Hazard and Operability Analysis (HazOp). These are especially useful in combination with Safety Case for identifying the risks and safety concerns that the safety case is going to handle. Also, methods like Failure Mode and Effects Analysis, Event Tree Analysis, Fault Tree Analysis and Cause Consequence Analysis can be used as tools to generate evidence for the safety case [Rausand91].

The need for concrete project artefacts as input in the safety case varies over the project phases, and is not strictly defined. Early on in a project, only a general system description is needed for making the safety requirements specification. When used in the inception phase, the Safety Case method will support the definition of a safety requirements specification document by forcing the developers to “prove” that their intended system can be trusted. When doing that, they will have to produce a set of safety requirements that will follow the project through its phases, and which will be updated along with the safety case documents.

The Safety Case method, when used to its full potential, will be too elaborate when not dealing with safety-critical projects. The main concept and structure will, however, help trace the connection between hazards and solutions through the design from top level down to detailed level implementation.

Much of the work that has to be performed when constructing a software safety case is to collect information and arrange this information in a way that shows the reasoning behind the safety case. Thus, the safety case does not in itself bring much new information into the project; it is mainly a way of structuring the information.

4.2 Preliminary Hazard Analysis and Hazard and Operability Analysis in a RUP context

By performing a PHA or HazOp we can identify threats attached to both malicious actions and unintended design deviations, for instance as a result of unexpected use of the system or as a result of operators or users without necessary skills executing an unwanted activity.

To perform a PHA or HazOp, we only need a conceptual system description, and a description of the system’s environment. RUP encourages such information to be produced in the inception phase of a project. When a hazard is identified, either by PHA or HazOp, it is categorized and we have to decide if it is acceptable or if it needs further investigation. When trustworthiness is an issue, the hazard should be tracked in a hazard log and subjected to review along the development process. This makes a basis for further analysis, and produces elements to be considered for the safety requirement specification.

The result of a PHA or HazOp investigation is the identification of possible deviations from the intent of the system. For every deviation, the causes and consequences are examined and documented in a table. The results are used to focus work effort and to solve the problems identified. The results of PHA and HazOp are also incorporated into the safety case documents either as problems to be solved, or as evidence used in existing safety claim arguments.

4.3 Combining the methods

By introducing the use of Safety Case and PHA/HazOp into the RUP inception phase, we have a process where the system safety requirements are maintained in the safety case documents. PHA and HazOp studies on the system specification, together with its customer requirements and environment description, produces hazard identification logs that are incorporated into the safety case as issues to be handled. This also leads to revision of the safety requirements. Thus, the deviations found with PHA/HazOp will be covered by these requirements as shown in Figure 3.

From the inception phase of the development process, the safety requirements and safety case documents are used in the remaining phases where the information is used in the implementation of the system.

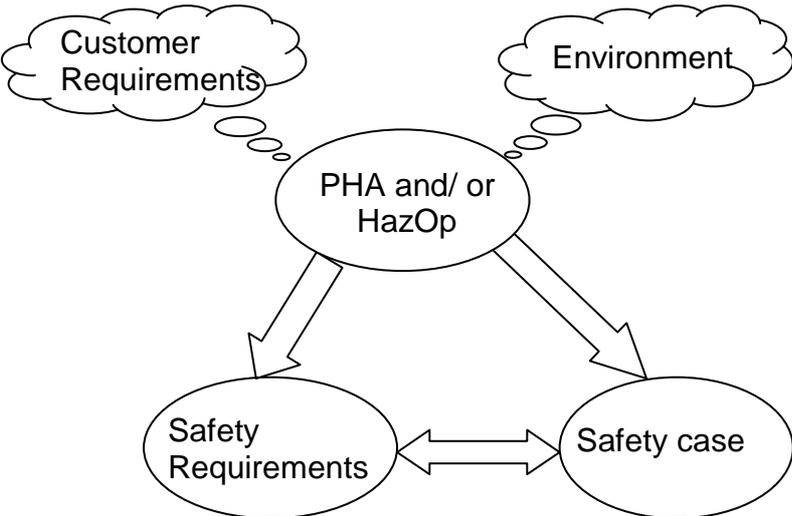


Figure 3 – Combining PHA/HazOp and Safety Case

5. A small example

Let us assume a business needing a database containing information about their customers and the customers’ credit information. When developing a computer system for this business, not only should we ask the business representatives which functions they need and what operating system they would like to run their system on, but we should also use proper methods to improve the development process with regard to business-critical issues. An example of an important requirement for such a system would be ensuring the correctness and validity of customers’ credit information. Any problems concerning this information in a system would seriously impact a company’s ability to operate satisfactorily.

The **preliminary hazard analysis** method will be helpful here, by making stakeholders think about each part of the planned system and any unwanted events that could occur. By doing

this, we will get a list of possible hazards that have to be eliminated, reduced or controlled. This adds directly to the safety requirements specification. An example is the potential event that the customer information database becomes erroneous, corrupt or deleted. By using a preliminary hazard analysis, we can identify the possible causes that can lead to this unwanted event, and add the necessary safety requirements.

We can use the system’s database as an example. In order to identify possible database problems – Dangers – we can consider each database item in turn and ask: “What will happen if this information is wrong or is missing?” If the identified effect could be dangerous for the system’s users or owner – Effects – we will have to consider how it could happen – Causes - and what possible barriers we could insert into the system. The PHA is documented in a table. The table, partly filled out for our example, is shown below in Table 1.

Customer info management			
Danger	Causes	Effects	Barriers
Wrong address	Wrong address inserted	Correspondence sent to wrong address	Check against name and public info, e.g. “Yellow pages”
	Update error		Testing
	Database error		
Wrong credit info	Wrong credit info inserted	Wrong billing. Can have serious consequences	Manual check required
	Update error		Consistency check
	Database error		Testing

Table 1 – PHA example

When we have finished the PHA, we must show that each identified danger that can have a serious effect will be catered to during the development process. In BUCS we have chosen to use safety cases for this.

When using the **safety case** method, the developers will have to show that the way they want to implement a function or some part of the system is trustworthy. This is done by producing evidence and a reasoned argument that this way of doing things will be safe. From Table 1, we see that for the customer’s credit information, the safety case should be able to document what the developers are going to do to make sure that the credit information used in billing situations is correct. Figure 4 shows a high level example of how this might look in a safety case diagram. The evidence may come from earlier experience with implementing such a solution, or the belief that their testing methods are sufficient to ensure safety.

The lowest level in the safety case in Figure 4 contains the evidences. In our case, these evidences give rise to three types of requirements:

- **Manual procedures.** These are not realised in software but the need to perform manual checks will put extra functional requirements onto the system.
- **The software.** An example in Figure 4 is the need to implement a credit information consistency check.
- **The process.** The safety case requires us to put an extra effort into testing the database implementation. Most likely this will be realised either by allocating more effort to testing or to allocate a disproportional part of the testing effort to testing the database.

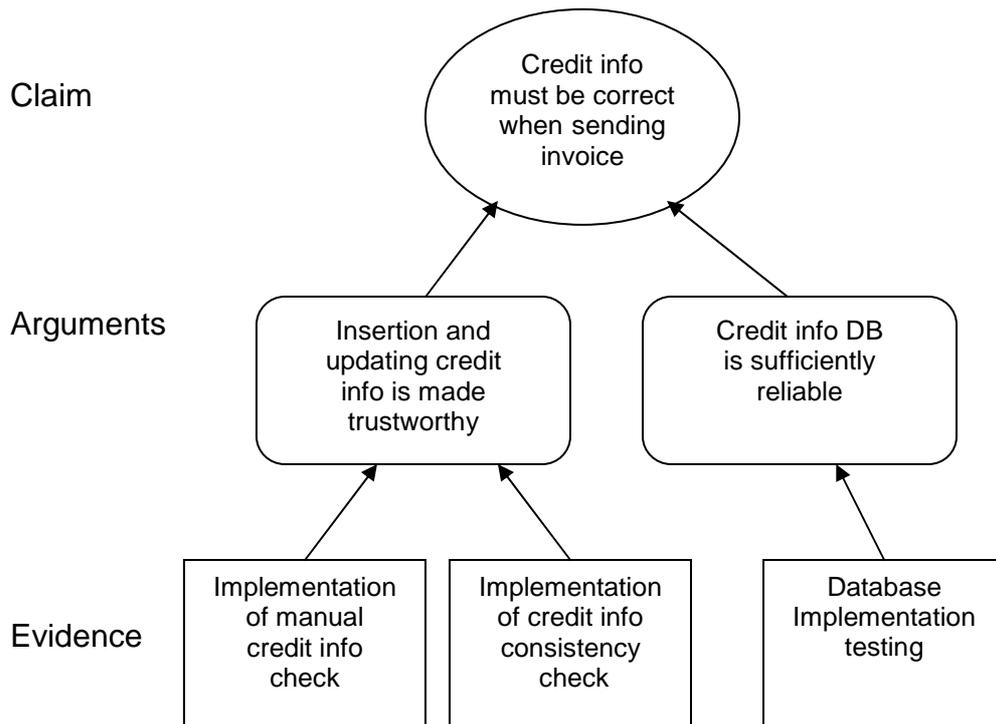


Figure 4 – Safety Case example

After using these methods for eliciting and documenting safety requirements, in the next development stages the developers will have to produce the evidence suggested in the diagram, show how the evidence supports the claims by making suitable arguments and finally document that the claims are supported by the evidence and arguments. Some examples of evidence are trusted components from a component repository, statistical evidence from simulation, or claims about sub-systems that are supported by evidence and arguments in their own right. Examples of relevant arguments are formal proof that two pieces of evidence together supports a claim, quantitative reasoning to establish a required numerical level, or compliance with some rules that have a link to the relevant attributes.

Further on in the development process, in the elaboration and construction phases, the evidence and arguments in the safety case will be updated with information as we get more knowledge about the system. Each piece of evidence and argumentation should be directly linked to some part of the system implementation. The responsibility of the safety case is to show that the selected barriers and their implementation are sufficient to prevent the dangerous event from taking place. When the evidence and arguments in the safety case diagram are implemented and later tested in the development process, the safety case documentation is updated to show that the safety case claim has been validated.

By using PHA to find potential hazards and deviations from intended operation, and Safety Case to document how we intend to solve these problems, we produce elements to the safety requirements specification, which without these methods may have been missed.

6. Conclusion and further work

We have shown how the Preliminary Hazard Analysis, Hazard and Operability Analysis and Safety Case methods can be used together in the RUP inception phase, to help produce a safety requirements specification. The shown example is simple, but demonstrates how the combination of these methods will work in this context. By building on information made

available in an iterative development process like RUP, we can use the presented methods to improve the process for producing a safety requirements specification.

As a development project moves into the proceeding phases, the need for safety effort will still remain to ensure the development of a trustworthy system. The other RUP phases contain different development activities and therefore different safety activities. The BUCS project will make similar descriptions of the other RUP phases and show how safety related methods can be used beneficially also in these phases.

BUCS will also continue the effort in working with methods for improving safety requirements collection, and will make contributions in the following areas:

- Proposals on adaptation of methods from safety development for business-critical system development.
- Guides and advice on business-critical system development.
- Tools supporting development of business-critical systems.
- Investigations on use of component-based development in the development of business-critical systems.

References

[Adelard98] "ASCAD, Adelard Safety Case Development Manual", Published 1998 by Adelard.

[Bishop98] P.G. Bishop, R.E. Bloomfield, "A Methodology for Safety Case Development", Safety-critical Systems Symposium (SSS 98), Birmingham, UK, Feb, 1998.

[Kroll03] P. Kroll, P. Krutchen, *The Rational Unified Process Made Easy: A Practitioner's Guide to Rational Unified Process*, Addison Wesley, Boston, 2003, ISBN: 0-321-16609-4.

[Krutchen00] P. Krutchen, *The Rational Unified Process: An Introduction (2nd Edition)*, Addison Wesley, Boston, 2000, ISBN: 0-201-70710-1.

[Leveson95] N.G. Leveson, *Safeware: System safety and computers*, Addison Wesley, USA, 1995, ISBN: 0-201-11972-2.

[Leveson00] N.G Leveson, "Intent specifications: an approach to building human-centered specifications", IEEE Transactions on Software Engineering, Volume: 26, Issue: 1, Jan. 2000, Pages:15 – 35.

[Rational] Rational Software, <http://www.rational.com>

[Rausand91] M. Rausand, *Risikoanalyse*, Tapir Forlag, Trondheim, 1991, ISBN: 82-519-0970-8.

[Stålhane03] T. Stålhane, T. Lauritsen, P.T. Myhrer, J.A. Børretzen, *BUCS rapport - Intervju med utvalgte norske bedrifter omkring utvikling av forretningskritiske systemer*, October 2003, <http://www.idi.ntnu.no/grupper/su/bucs/files/BUCS-rapport-h03.doc>