Deriving Safety Requirements Using Scenarios

Karen Allenby
University of York
Department of Computer Science
Heslington, York, YO10 5DD
karen.allenby@cs.york.ac.uk

Tim Kelly
tim.kelly@cs.york.ac.uk

Abstract

Elicitation of requirements for safety critical aero-engine control systems is dependent on the capture of core design intent and the systematic derivation of requirements addressing hazardous deviations from that intent. Derivation of these requirements is inextricably linked to the safety assessment process. Conventional civil aerospace practice (as advocated by guidelines such as ARP4754 and ARP4671) promotes the application of Functional Hazard Assessment (FHA) to sets of statements of functional intent. Systematic hazard analysis of scenario-based requirements representations is less well understood. This paper discusses the principles and problems of hazard analysis and proposes an approach to conducting hazard analysis on use case requirements representations. Using the approach, it is possible to justifiably derive hazard-mitigation use cases as first class requirements from systematic hazard analysis of core design intent scenarios. An industrial example is used to illustrate the technique.

1. Introduction

This section provides an overview of safety requirements and their relationship to core requirements. The concept of hazard analysis is introduced and some of the problems encountered with its application in practice are outlined.

1.1. Safety requirements

Safety critical systems within the civil aerospace sector are developed subject to the recommendations outlined in ARP4754 [1] and ARP4761 [3]. These standards give guidance on the 'determination' of requirements, including requirements capture, requirements types and derived requirements.

Requirements capture is based primarily on the identification of core aircraft functions. As design progresses, multiple levels of system emerge and the aircraft requirements are discharged by decomposition and allocation across sub-systems. In addition to core requirements, the design process incorporates many decisions that lead to additional requirements being placed on sub-systems. These requirements, which may not arise directly from the core aircraft requirements, are referred to as derived requirements.

Safety requirements derived through safety analysis will place integrity constraints on existing core functions. In addition, new functional requirements may be needed to prevent or mitigate the effects of failures identified in the analysis.

ARP4754 recommends that derived requirements "be captured and treated in a manner consistent with other requirements applicable at that development phase". It is good practice to treat safety related functional requirements in a similar way, since they are subject to the same obligations as other requirements with respect to traceability.

1.2. Hazard analysis

Hazard analysis is "those activities within safety analysis which pertain to identifying hazards, determining their causes, and planning their elimination or mitigation" [4]. Hazard analysis, therefore, provides the mechanism for identification of the safety related requirements discussed above, as "countermeasures" implemented to improve the safety of the system.

Functional hazard assessment (FHA) is a technique advocated in ARP4754 and ARP4761 as a way of systematically identifying hazards (" a physical situation, often following some initiating event, that can lead to an accident" [2]). The FHA process is described as follows in ARP4761.

1. Identification of all functions associated with the level under study.

2. Identification and description of failure conditions associated with these functions.

3. Determination of the effects of the failure condition.
4. Classification of failure effects on the aircraft.
5. Assignment of requirements to the failure conditions to be considered at the lower level.

Experience in application of FHA to engine controller development has led to the identification of common problems associated with the technique.

Definition of functional requirements An essential precursor to efficient FHA are concise, unambiguous requirements, at consistent levels of abstraction. Extraction of a requirements set of this quality from large, natural language documents is an undesirable overhead of the safety process that needs to be alleviated, perhaps through adoption of an improved functional representation.

Completeness Completeness is a primary objective of failure identification [14]. Although FHA improves completeness over the use of traditional hazard checklists (records of accumulated experience), there is little detailed guidance in ARP4754 regarding assurance of completeness through consideration of appropriate failure classes.

Behaviour There is no explicit mechanism in the technique for identification of hazards associated with the current system state. The consideration of functions alone is insufficient to identify hazardous conditions, since some functions are intended only to be executed during certain aircraft modes, or flight phases.

Integration of safety related functional requirements Following hazard identification and assessment, new functional requirements are likely to be derived for inclusion in the system specification. The safety and requirements processes may not adequately support the integration of these requirements into the system specification in a satisfactorily traceable manner. Often, safety related functional requirements appear in documentation without due acknowledgement of their origin in hazard analysis. This is a problem associated with hazard analysis generally, rather than a single technology for hazard identification and a solution is likely to be found in adequate definition of the relationship between the safety and requirements processes, supported by a suitable requirements representation.

1.3. Objectives of the proposed approach

While the safety process remains entirely separate from the requirements definition process, the problems of functional definition, completeness, behaviour and integration highlighted in 1.2 are unlikely to be resolved. In order to address the problems posed by the derivation of safety requirements and in particular hazard analysis, an integrated approach to requirements expression and failure identification is proposed. Defining such a technique will alleviate some of the current discontinuities that exist between the requirements and safety processes and improve confidence in the systematic identification of failures. The following sections provide more details on the objectives of the work.

2. Requirements representation

The first step in the FHA process is the identification (and representation) of all functions associated with the level under study, involving the examination of a number of data sources [3]. This section highlights the desirable qualities for a representation that will be amenable to hazard analysis. The section goes on to describe the proposed approach to this step of the process, including details of graphical representation and scenario documentation.

2.1. General principles

A method of function specification amenable to hazard analysis also supports good requirements practice by addressing requirements quality attributes (understandability, redundancy, completeness, ambiguity, consistency, organisation, conformance to standards and traceability [9]).

ARP4754 and ARP4761 do not specify the type of representation required for FHA. There are, however, some common properties that will apply to any chosen representation technique for requirements:

- shared understanding (across multiple stakeholders);
- expressiveness;
- clarity and conciseness;
- implementation independence.

At worst, functional requirements are specified in monolithic requirements documents, in some form of natural language. This form of expression makes the direct application of functional failure identification techniques difficult, if not impossible. However, representation of functions in the safety process is often reduced to a simple function tree, giving very limited information regarding intended behaviour (such as state dependency and timing).

2.2. Use cases

The use case concept [8] has been adopted as part of the Unified Modelling Language (UML) [13]. According to the

UML standard, use cases are represented in use case diagrams (see figure 1), showing a set of use cases enclosed by a system boundary, associations between actors and use cases, relationships among use cases and generalisation between actors. The standard, however, offers limited guidance on the documentation of individual use cases beyond brief, informal text.

Although the adoption of use cases has largely come from within the object-oriented community [8], object-orientation is not a pre-requisite for their use in system development [7]. Use cases represent an means of gathering, recording and communicating requirements that is not dependent on implementation technology.

There are a number of reasons why use cases are a suitable medium for the representation of functions for safety-critical systems, including:

- explicit representation of interactions with actors (perhaps other sub-systems) in the environment of the system;
- the ability to represent multiple levels of system using use cases at multiple levels of abstraction;
- a black-box approach to specification helps to define system boundaries and provides a framework for enforcing consistency of abstraction levels in description;
- avoiding redundancy through use of generalisation, extend and include relationships.

2.3. Scenarios

Scenarios are sequences of actions used to illustrate system behaviour. While a use case is intended to represent system functions for the general case, scenarios represent operational instances of system use. In practice, however, the distinction between the two is less clear and the terms are often used interchangeably.

Scenarios are a common concept in requirements engineering and they appear in numerous techniques [8, 10]. The level of interest in the approach is evident from the number of recent contributions to literature on the subject [5], covering various aspects of development. Although the literature contains many variations in style, there are some simple guidelines on minimum content [9]:

- a description of the state of the system before entering the scenario;
- the normal flow of events in the scenario;
- exceptions to the normal flow of events;
- information about other activities which might be going on at the same time;
- a description of the state of the system after completion of the scenario.

Scenarios each represent intentional uses of the system, but perhaps under different circumstances or with different pre-conditions. However, the goal of the scenario (or its post-condition) is always the same. Exceptions to the flow of events are errors that arise in the execution of the scenario, either through actor interactions or through system malfunctions. It is usual to specify how these exceptions will be handled by the system. Identification of these exceptional events and specifying their mitigation is similar to the identification of failures in hazard analysis.

It is worth noting that there is little guidance available within the UML standard [13] or accompanying guidance material [6] for the systematic identification of either ‘alternative paths’ or ‘exceptional courses’ of events in scenario or use case descriptions. Under these circumstances, the practitioner is left with little assurance of sufficient coverage. An impediment to adoption of a use case approach in safety critical systems is the current lack of systematic method for identification of these alternative paths in association with failure conditions.

2.4. Proposed approach to representation

The essential steps of this part of the proposed approach are summarised below.

1. Identification of core aircraft use cases.
2. Identification and documentation of the scenarios for each core use case.
3. Decomposition and allocation of functionality across communicating sub-systems.

Functional requirements at each development level under consideration will be represented in use case diagrams. A typical use case diagram for a two layer system will look like that shown in figure 1. The use case scenarios are documented in tabular format using structured text, described below.

Use case The name of the use case (or function) to which the scenario applies.

Scenario The name of the scenario that captures the operational use of the function. Possible scenarios are identified with reference to the intended operational states of the system, typically described as an annotated state chart in the requirements or design model. The scenarios encapsulate the goals to be achieved using the functions described. The expansion of the state chart to include hazardous or emergency states will progress with the hazard analysis.

Pre-conditions The operational state that the system must be in before the function is executed. In all other states (unless specified) the function is not available. These conditions may also apply to the state of the external systems being monitored.

Guard-condition A conditional expression used to determine the triggering of system responses.

System response The (often continuous) response of the system based on the inputs and current state. The system is treated as a black box and only externally visible behaviour is recorded in the scenario specification.

Post-condition The state of the system (or external systems) following execution of the use case (the end result, or goal).

3. Failure identification

Failure identification, determination of failure effects and classification of failure effects on the aircraft are steps 2, 3 and 4 of the FHA process (see section 1.2). In this section we outline two established approaches to hazard identification (FHA and HAZOP) and also the proposed technique. Unlike safety analysis techniques such as Failure Modes and Effects Analysis (FMEA) that assess the effects of known behaviour, both FHA and HAZOP systematically consider hypothetical deviations from declared intent. This makes them appropriate for use early in the development lifecycle.

3.1. FHA

In FHA, each of the represented core functions is typically considered for possible failures in three common categories:

- loss of function;
- function provided when not required;
- incorrect operation of function.

These failure categories are guidewords that ensure some level of systematic failure identification. However, the categories represented do not necessarily provide a complete framework for failure identification, relying on 'incorrect operation' as a catch-all term for possible failure modes.

In addition, FHA provides no mechanism for the systematic identification of failure causes. While the intention of FHA is to remain at the level of abstract functional specification, some useful insights can be gained when failure causes are included in the analysis. It may be possible to identify, at an early stage, reliability, accuracy and timing requirements for system level functions or cross-boundary data exchanges, that are critical to safety.

3.2. HAZOP

A hazard and operability study (HAZOP) is "the application of a formal systematic technique to the identification of hazards" [12]. Although HAZOP originated as a hazard identification technique for process plants, it has broad applicability to any type of system. Guidance on application of the technique to systems containing programmable electronics is encapsulated in U.K. Ministry of Defence Standards [2, 4].

DEF STAN 00-58 [4] gives some more specific guidance on the form of representation used in the analysis, specifying that any chosen technique should be able to represent:

- design intent and the attributes which enable the identification of system hazards;
- where the subject of the study is embedded in a larger system, interactions with other parts of the system;
- the user and interactions between the user and the system;
- interactions with the environment.

HAZOP shares its objectives with FHA but uses a design view of the system, rather than a functional requirements one, as the basis for failure identification. HAZOP also uses the application of guidewords (No, More, Part of, Other than, Early, Late, Before, After) to individual flows as a tool for failure identification. However, the guideword set is more extensive than for FHA and has the potential to ensure a more complete investigation of possible deviations from intent than is achieved using FHA.

Within the process industries, HAZOP is established as one of the foremost hazard identification techniques. However, the consistency of application and its effectiveness in practice have proved problematic [12]. Clearly, there are similarities between the FHA and HAZOP processes. Although HAZOP is based on flows and FHA on function, the two approaches are complementary. A combined approach to hazard identification based on requirements statements should be possible, using a suitably expressive requirements representation.

3.3. Proposed technique for failure identification

The proposed technique makes use of a subset of refined software HAZOP guidewords [11]:

Omission The service is never delivered: there is no communication.

Commission The service is delivered when not required: there is an unexpected communication.

Early The service occurs earlier than intended: this may be absolute or relative.

Late The service occurs later than intended: this may be absolute or relative.

Value The information delivered has the wrong value.

The essential steps of the technique are summaried below.

1. For every scenario describing the chosen functional requirement, identify each pre-condition, guard-condition, system response and post-condition and record them as 'elements'in the analysis documentation table (see figure 2).
2. Apply each of the guidewords in turn to the identified element.
3. Interpret the application of the guideword into identifiable deviation from core intent.
4. Identify possible failure causes.
5. Interpret the failure in terms of the use case.
6. Interpret the effect of deviation on the system and aircraft.
7. Assign a failure classification based on the aircraft level effect.
8. Identify necessary integrity constraints on the core function.
9. Identify where the failure classification merits the incorporation of new, safety-related use cases.

3.4. Initial experience of applying use cases to aero-engine control

Experience in the application of use cases for the specification of requirements for engine control systems at Rolls-Royce has shown that it is possible to represent functions within the safety critical domain using use case diagrams and scenario descriptions. There are benefits to be gained from applying the concepts directly to existing specifications, such as increased visibility of system structure, functional dependencies and consistency of abstraction in expression. However, there are some critical factors that limit the potential of simply applying notational and structural concepts.

Completeness There is no explicit mechanism for separation of core and safety related requirements in the specification. The degree of completeness of the requirements set is, therefore, difficult to determine.

Justification The origin or justification of safety related requirements remains implicit, even though experience suggests that they exist as a result of previous safety analyses.

Until these factors are addressed, there are limitations on potential improvements in aspects of the specification such as requirements traceability and amenability to change.

4. Identifying safety requirements

The identification of new, safety related requirements from the analysis is the final step in the FHA process. This section discusses the potential for identification of cases where the results of analysis call for the identification of new, safety related functional requirements and also their subsequent specification.

4.1. Criticality of failure

Failures, once identified, must be assessed for their potential effect at the aircraft level. The failure classifications (based on accident/incident data, regulatory guidance material, previous design experience and consultation with flight crews) are [3]: Catastrophic, Severe-Major/Hazardous, Major, Minor and No safety effect. The failure classification assigned to each of the failure effects provides some indication of the need for safety related requirements. In the case of a catastrophic failure (all failure conditions which prevent continued safe flight and landing) the potential for damage, injury or loss of life may preclude the assignment of integrity requirements in isolation as a satisfactory means of failure prevention or mitigation. In this case, the implementation of new functions, specifically targeted at prevention and mitigation of failure may be required.

4.2. Proposed approach to specifying safety-related functional requirements

ARP4754 recommends that derived requirements should be captured and treated in a manner consistent with other requirements applicable at the same development phase. Similarly, it is important that safety requirements, if they are to be manifest in system design, are treated as part of the 'mainstream' requirements set. The specification of safety requirements will, therefore, be the same as that adopted for core requirements, outlined in section 2.4. The use of a consistent representation demonstrates a positive approach to the specification of safety related functions by acknowledging their status as independent functions, augmenting the requirements set. In addition, they are also subject to hazard analysis and their representation needs to be consistent with the hazard analysis technique used.

5. Application of the proposed technique

The following example is aimed at demonstrating the utility of the concepts outlined in sections 2.4 and 3.3. The example is taken from the engine control domain, targeted specifically at the identification of system level safety requirements relating to the control of reverse thrust. The example shows the selection of a core aircraft function (deceleration), its decomposition and allocation to sub-systems of the aircraft and the subsequent identification of failures and safety requirements associated with a single function allocated to the engine controller (reverse thrust direction). A certain level of prior design knowledge is assumed, in order to carry out the sensible allocation of core aircraft functionality.

5.1. Representation of core functionality

Operational context for core system functions is provided by reference to the behvaiour of the aircraft. Its statechart was used to derive a structured operational context for 'decelerate', selected for analysis here (see table 1). This context is applied consistently at the system level when considering 'reverse thrust direction' (table 2).

Use Case	Decelerate
Scenario	Decelerate on landing
Pre-conditions	Landing
System Response	While [the Pilot commands deceleration] the Aircraft shall: • decelerate
Post-conditions	Aircraft speed = Pilot commanded speed

Table 1. 'Decelerate on landing' scenario

Subsequent decomposition of the use case and allocation of refined functions to communicating aircraft systems is shown in the system level use case diagram in figure 1. Of the two core system level functions allocated to the engine controller, 'reverse thrust direction' is targeted for further analysis.

The use case diagram shows that, during execution of this use case, the controller communicates with the engine, thrust reverser and airframe systems in order to gather data and effect control. The relevant specification for this function in the context of the 'decelerate on landing' scenario is shown in table 2.

5.2. Identification of failures

Each element (pre-conditions, guard-conditions, system responses and post-conditions) of the scenario documented in table 2 was identified and recorded in an analysis table. Each element was subject to the process steps identified in section 3.3. The results of the failure identification and subsequent interpretation of failures was documented in accordance with the information listed in figure 2.

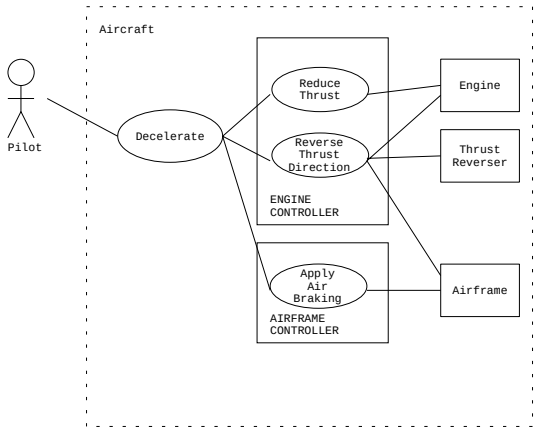

Figure 1. System level use case diagram.

Use Case	Reverse thrust direction
Scenario	Decelerate on landing
Pre-conditions	Airframe on ground AND Engine running
System Response	While [the Pilot commands reverse thrust] the System shall: • command the Thrust Reverser to deploy • if [the Thrust Reverser state = in transit], command Engine thrust demand to thrust limit
Post-conditions	Thrust reverser state = deployed

Table 2. System level scenario

The analysis yielded failures with nine different real world consequences. The classification of these failures was: 3 catastrophic (for example, loss of controlled flight, the result of applying 'omission', 'commission' and 'value' to pre- and guard- conditions); 5 hazardous (for example, possible runway overshoot identified by applying each of the guidewords to pre-, guard- and post-conditions and system responses); 2 no safety effect ('late' detection of post-condition, resulting in slower than anticipated speed). Catastrophic failures at the system level are described in figure 2.

Element	Guideword	Deviation	Possible Causes	Use Case Effect	Real World Effect	Severity	Integrity Constraints	New Safety Requirements
Airframe status = on ground *(pre-condition)*	Commission	On ground detected when not true	System failure; invalid airframe data; data transmission failure	Reverse thrust implmented when pre-condition not satisfied	Thrust reverser deployed when not on ground; loss of controlled flight	Catastrophic	Assign on ground detection realiability; validate airframe data; specify data sampling rate	Disallow thrust reverser when airframe not on ground; detect inadvertent deploy; provide auto restow
Thrust reverser state = in transit *(guard condition)*	Omission	Thrust reverser state = in transit not detected when true	System failure; invalid thrust reverser data; data transmission failure	Engine thrust demand not commanded to thrust limit when guard condition satisfied	Engine thrust exceeds thrust limit; structural damage to thrust reverser; loss of controlled deceleration on landing	Catastrophic	Assign thrust reverser state detection reliability; validate thrust reverser data; specify data sampling rate	
Thrust reverser state = in transit *[guard condition]*	Value	Thrust reverser state = in transit detected as thrust reverser = deployed	System failure; invalid thrust reverser data; data transmission failure	Engine thrust demand not commanded to thrust limit when guard condition satisfied	Engine thrust exceeds thrust limit; structural damage to thrust reverser; loss of controlled deceleration on landing	Catastrophic	Assign thrust reverser state detection reliability; validate thrust reverser data; specify data sampling rate	

Figure 2. Catastrophic failures (extract from system level analysis).

5.3. Identification of safety related use cases

New safety requirements were identified for a number of failure conditions. Most of these requirements were performance constraints, such as the assignment of timing requirements. However, consideration of the catastrophic failures associated with the thrust reverser led to some additional functional requirements. The following requirements were identified to prevent and mitigate the first of the catastrophic failures identified in figure 2.

- 'Disallow thrust reverser deployment' – prevention of catastrophic failure (loss of controlled flight) due to unintentional thrust reverser deployment;
- 'Detect inadvertent deploy' – mitigation of catastrophic failure by detection of unintentional thrust reverser deployment;
- 'Provide automatic restow' – mitigation of failure by automatic stow following detection of an unintentional thrust reverser deployment.

The first of these requirements is likely to be implemented as an explicit set of pre-conditions (or system state) for which the reverse thrust direction use case must not be allowed. The remaining two requirements are new functions and can be defined as use cases. Specification of these use cases, which augment the system level use case diagram, will also be in the form of scenarios and the functions will be subject to hazard analysis.

6. Discussion

6.1. Observations

Representation Referring back to the guidance on representation from DEF STAN 00-58 (see section 3.2) we can see that the scenarios clearly convey the intent of the function being specified (via the scenario name). Elements identified from the structured scenario description facilitate a systematic approach to failure identification. Interactions with other parts of the system are recorded in the use case diagram and scenario description, elements of which translate as system inputs (pre-, guard- and post-conditions) and outputs (system responses).

Failure identification When coupled with the structured scenario descriptions, the use of guidewords provides a systematic method of failure identification, assuring a certain level of completeness. The approach also includes consideration of failure causes, for which there is no equivalent in FHA. Identification of potential failure causes provides useful information to designers when determining optimal ways of preventing or mitigating the effects of failure.

Taken out of context, the application of the guideword 'early' to conditional statements could seem meaningless (either the condition is true or it is not). However, given that we have knowledge of the intent of the function, through the scenario name, the concept becomes meaningful. Consider the early detection of a pilot reverse command. At first sight, this failure can be considered the same as commission of the pre-condition. However, in the context of the aircraft landing, early detection of the command and the subsequent failure effect may be less critical if the pilot is about to land, rather than take off. The explicit referencing of context in this sense adds value to the analysis by making such distinctions.

New requirements This analysis, like others, primarily bases the justification for new requirements on the criticality of identified failure effects. However, the approach taken offers the ability to integrate new requirements directly into the requirements model with explicit reference to analysis, providing improved traceability. We have demonstrated how failure identifica-

tion is used to refine and augment the core requirements set by identifying additional requirements at the same development level.

6.2. Limitations of the approach

The example used here is restricted in scope and it has not been possible to develop some aspects of the technique that are inevitably required in order to demonstrate completeness. In order to fully capture failures, consideration needs to be given to the operation of functions in emergency or degraded operational states (such as aborted take-off, inclement weather, etc.). This completeness is related to the reliable identification of scenarios, which has not been fully explored here and which is the subject of ongoing work.

So far, the proposed analysis technique does not take account of multiple failure combinations (such as combinations of pre-condition failure) or consistent failure across multiple similar systems (such as asymmetrical thrust reverser deployment). The technique needs to be extended to include consideration of these failure modes.

A further extension to the technique would be the inclusion of combinations of use cases, or ordering, in the analysis. This concept is a standard part of HAZOP (Before and After guidewords) and would enhance the capabilities of the technique. Scenarios give an ideal mechanism for investigating this order by using sets of pre- and post-conditions to determine allowable or obligated sequences of actions for requirements specification.

7. Conclusions

It is attractive to apply a scenario-based requirements approach in the development of safety critical systems. However, it must be possible to integrate the expression of requirements of core design intent with the activities of, and resultant derived safety requirements from, systematic hazard identification techniques such as HAZOP and FHA. In this paper we have demonstrated how it is possible to apply an adaptation of conventional hazard identification techniques to the structured documentation of functional requirements using scenarios. Based on the results of the hazard identification we expose the rationale for deriving safety requirements that can then form part of the specification of desired system behaviour. Through such an approach we believe there can be increased confidence in the completeness of the specification of safety critical systems.

8. Acknowledgements

The authors would like to acknowledge the financial support given by Rolls-Royce plc for the work reported in this paper.

References

[1] Certification considerations for highly-integrated or complex aircraft systems. Society of Automotive Engineers, December 1994.

[2] Draft defence standard 00-56: Safety management requirements for defence systems containing programmable electronics. Ministry of Defence, UK, 1995.

[3] Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. Society of Automotive Engineers, August 1995.

[4] Interim defence standard 00-58: Hazop studies on systems containing programmable electronics. Ministry of Defence, UK, 1996.

[5] *Requirements Engineering Journal*, 3(3/4), 1998.

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

[7] A. Jaaksi. Our cases with use cases. *Journal of Object Oriented Programming*, pages 58 – 65, February 1998.

[8] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, Harlow, England, 1992.

[9] G. Kotonya and I. Sommerville. *Requirements Engineering: Processes and Techniques*. Wiley, Chichester, England, 1998.

[10] C. Potts, K. Takahashi, and A. I. Anton. Inquiry-based requirements analysis. *IEEE Software*, 11(2):21 – 32, March 1994.

[11] D. J. Pumfrey. The principled design of computer system safety analyses. D.Phil Thesis, University of York, UK, 1999.

[12] F. Redmill, M. Chudleigh, and J. Catmur. *System Safety: HAZOP and Software HAZOP*. Wiley, Chichester, England, 1999.

[13] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Reading, Massachusetts, 1999.

[14] P. J. Wilkinson and T. P. Kelly. Functional hazard analysis for highly integrated aerospace systems. Presented at IEE Seminar on the Certification of Ground/Air Systems, IEE Savoy Place, London, 1998.