Towards a safety case for runtime risk and uncertainty management in safety-critical systems

R Eastwood*, T P Kelly*, R D Alexander*, E Landre†

*University of York, United Kingdom <ralph@cs.york.ac.uk>, <rob.alexander | tim.kelly@york.ac.uk>
†Statoil KTJ/IT, Forus, Stavanger / Rotvoll, Trondheim, Norway <einla@statoilhydro.com>

Abstract

Many safety-critical systems have a human-in-the-loop for some part of their operation, and rely on the higher cognitive abilities of the human operator for fault diagnosis and risk-management decision-making. Although these operators are often experts on the processes being controlled, they still sometimes misjudge situations or make poor decisions. There is thus potential for Safety Decision Support Systems (SDSS) to help operators, building on past successes with Clinical Decision Support Systems in the health care industry. Such SDSS could help operators more accurately assess the system's state along with any associated risk and uncertainty. However, such a system supporting a safety critical operation inevitably attracts its own safety assurance obligations. This paper will outline those challenges and suggest an initial safety case architecture for SDSS.

1 Introduction

An operator sits in a stuffy control room, watching a screen with monitoring data scrolling down. The operator is drowsy; it is near the end of his shift and he has had nothing significant to do for hours. Suddenly, alarm lights flash and a series of audible alarms are tripped. The operator looks around and is startled to see that these are not the normal "nuisance" alarms that he normally waves away. He diagnoses it as a genuine problem - he saw something similar about three years ago which turned into a serious incident. He calls up his supervisor, and talks for a few minutes about what he believes the alarms to indicate. He strongly advises that they divert the normal operation of the plant to perform some tests as a precautionary measure. The supervisor, eventually convinced, reluctantly agrees to stop the plant so they can do some tests to see whether it is safe to continue.

This example could happen in many areas of process industry: nuclear, chemical and oil; the operators performing the monitoring and process control may have years of experience, but it is often described as a task that involves "99% boredom and 1% sheer terror" [1]. The time scales for responding to alarms vary from industry to industry and from problem to problem (and hence procedures can differ significantly from the example) but it captures the influence of risk and uncertainty pertaining to the decision making process in such situations. With small permutations to the example, the situation could be made very different and potentially a lot worse.

Conceivably, a Decision Support System could be developed to take sensor information to determine probabilistically the state of the system, and then provide advice to the operator. This would be somewhat similar to clinical Decision Support Systems (which are showing increasing use [2]). Ideally, the system would list the various options available along with their associated risk and uncertainty; even better, the reasoning process behind that information would be available to the operator on request. Such a system would be able to truly work alongside the operator — we call this a Safety Decision Support System (SDSS) here.

However, "Who Guards the Guardians?" — Schumann et al. [3] write that there is an unanswered concern of how to assure the safety of the software that is tasked to monitor a system.

2 Related Work

There is much current interest in the use of Decision Support Systems for safety-critical applications. In particular there is work in Prognostics and Health Management Systems [4] and Condition-based Maintenance Systems [5] which involve activities such as fault detection, fault diagnostics and failure prognostics. Much of the work, however, is focussed on the techniques of modelling in application-specific domains, rather than on generally-applicable techniques.

Work in Prognostics and Health Management systems [6] has paid some attention to reliability, but little to safety (to ensuring that the system does not have dangerous side effects). Frost et al. [7] express the concern that there can be negative impact of decision support on the decision-making ability of a human pilot. Schumann et al. do consider the issues of verifying, validating and certifying "all" software on a system [8] (this necessarily includes any Decision Support System), but this is only a very preliminary investigation.

The most closely related work to our current concerns is that of Pace and Seward [9] in the domain of autonomous robotics. They present a method to make autonomous robots that manage safety in themselves, and provide justification for that use in a safety context.

3 Abductive Reasoning

Normal safety analyses performed on systems can be described as asking "what if" questions throughout the safety lifecycle during the development of a system to discover hazards, their causes and their likelihood and likely consequences. This begins with predictive analyses that use inductive techniques (that work forward from known causes to potential (unknown)

effects) and ends with confirmatory analyses that use both inductive and deductive techniques (the latter that work backwards from a known unwanted event) [10]. The SDSS would supplement the above by supplying abductive reasoning (reasoning that fits the available knowledge to the most likely of several rival theories) in the act of transforming the information from sensor inputs into the likely explanations of the system state (e.g normal operation, valve X is blocked, natural gas is pushing up through the wellbore. . .).

4 General function of a SDSS

The general function of a SDSS would initially begin with the abductive reasoning stage which influences later reasoning. This abductive reasoning will give information about the probable state that the system is in, with some uncertainty. The risk analysis stage would then perform inductive analyses which will reveal the risks posed by being in that state (the models used for this would be described from standard safety analyses of the system). Anomaly detection could also be performed to detect abnormal patterns — patterns that the SDSS has not seen before. These may be innocuous or they may correspond to an unanticipated failure mode or a flaw in the SDSS's model of the plant. The decision support stage will receive the results of the risk analysis and will consider the trade-offs between reducing risk, reducing uncertainty, and maintaining performance. Based on this, it will then present the operator with a number of alternative plans; ideally, it will be able to explain the rationale behind each plan to the operator, on demand.

5 An Architecture Model of an SDSS

A general model of a SDSS is presented in Figure 1.

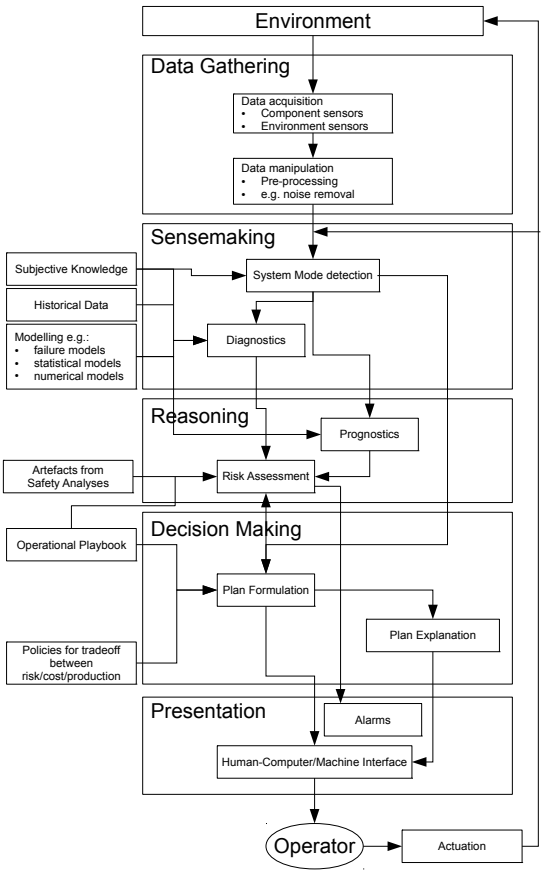

Figure 1. SDSS Architecture Model

5.1 Data Gathering

The Decision Support System needs to be able to sense the world: the sensors on the physical system and the environment with which the system interacts. Some preprocessing of the raw data could be involved to remove noise.

5.2 Sensemaking

Sensemaking is the continuous effort to understand and interpret relevant information for the purpose of anticipating events and making decisions in response [11]. This is a crucial part of a Decision Support System to "orientate" itself in the mass of data being received from the sensors. For example, sensemaking would involve the process of determining the mode or state of the system (e.g. is the chemical process active or inactive) using historical data and models of the advised system and the process it controls.

5.3 Reasoning

Reasoning in a Decision Support System is the intermediate step between the Sensemaking and the Decision Making (in practice, reasoning would be closely tied to the Decision Making) where prognostics is performed to predict future states of the system. Risk assessment provides a utility to help determine the best decision to make with regards to safety analyses, the operational playbook and the various plans that can be formulated. Here, we have used the term "Playbook" to refer to the collection of standard procedures, including the conditions under which each is appropriate to apply.

5.4 Decision Making

The *Decision Making* is where the plans of action are actually made. This involves taking all the information that has been processed and deciding the best course of action based on that with regard to standard procedures. One constraint on the decision-making approach used here is that the SDSS needs to be able to describe to the operator *why* it has made the decision it has. The approach used must, therefore, be one that produces such an intelligible rationale.

5.5 Presentation

The interface between the Decision Support System and the operator is extremely important. This has to be done in such a way that it doesn't overload the operator nor skip important details relevant to the current situation. Although there is relevant work and experience in conventional alarm and monitoring systems, it is not clear exactly how to do this for more

sophisticated SDSS.

The alarm management system may work autonomously; if there is a high risk risk system state, it may be able to trip alarms (audible and visible to not just the operator but to all in the area of concern) without operator intervention. In this, it is slightly more autonomous than the rest of the SDSS, which is ultimately subordinate to the operator.

6 Architectural Failure Modes

The standard Functional Failure Analysis technique, as described in the ARP 4761 (as "System Functional Hazard Assessment") [12], provides three failure classifications to help guide a systematic exploration of failures during design time:

(1) Function not provided
(2) Function provided when not required
(3) Function provided incorrectly.

Although these are applicable to software failures as well, many types of failures would be simply classified as *function provided incorrectly* [10]. The failure modes in this paper utilise a set of failure classifications (in Table 1) from the SHARD HAZOP technique [10] to provide guidance in systematically analysing the architecture. Note that these tables are not exhaustive due to the abstract nature of the SDSS we consider and to space constraints.

The extracts highlighting the distinctive challenges of the SDSS from our FFA are presented here (avoiding the obvious "software bug prevents execution").

Group	**Failure classes**
Service provision	(a) Omission, (b) Commission
Service timing	(c) Early, (d) Late
Service value	(e) Coarse (detectably) incorrect, (f) Subtle (undetectably) incorrect

Table 1. SHARD Failure Classifications [10].

Modelling the system plays an important role in the SDSS; however, it is a difficult task to model all phenomena completely – more so when considering environmental effects that occur in an oil drilling scenario, or autonomous robots working in an outdoor environment. We call this *modelling uncertainty*.

6.1 Data Gathering

The input sensors that provide data to the SDSS play a large role as it involves the entire process, up to the Plan Formulation and its explanation, is dependent on them. As sensors are often quite exposed to the environment, they are quite likely to fail. Hence, failures here must be mitigated through redundancy and cross-checking for contradictory sensor readings. Table 2 shows the FFA for some of the more subtle problems that may occur.

Data Manipulation refers to pre-processing of the sensor data, and possible failures here are shown in Table 3. It is possible for biases to be introduced and for subtle artifacts to be produced through e.g. noise smoothing in this pre-processing. The effect of these on the Risk Assessment and Plan Formulation may be difficult to predict (or even detect).

Failure Mode	Indicative Cause(s)	Effects
(a) Sensor readings are lost.	(1) Sensor readings come in a lower granularity than expected (misconfiguration). (2) Faulty connections causing lost data.	Reduction in the confidence of risk assessment and ultimately plan formulation.
(e) Sensor data values are out of range.	(1) Implementation does not follow sensor specification. (2) Sensor is malfunctioning and providing data outside of its specification.	Invalid data leading to readings lost due to filtering.
(d) Sensor updates are too late.	(1) Sensor updates are sent through a congested network. (2) Polling of the sensor is too slow.	The sensor data is too late to be useful – the SDSS will either be unable to use the sensor information or will treat it as current when it is not.
(f) Fluctuating sensor readings.	Conditions exceed specification of the sensor causing undefined behaviour.	Incoming sensor data has spurious values as the conditions have caused the sensor to go out of range.

Table 2. Sensor FFA.

Failure Mode	Indicative Cause(s)	Effects
(e) Data values are corrupted.	(1) Uncertainty in modelling causes incorrect preprocessing to be performed. (2) Software bug.	Invalid data fed to system mode detection, diagnostics, prognostics and ultimately plan formulation. Alarms may be triggered due to misinterpretation of such values.
(f) Introduction of a bias in the data values.	(1) Uncertainty in modelling introduces hidden bias. (2) Software bug.	Invalid data fed to system mode detection, diagnostics, prognostics and ultimately plan formulation.
(f) Removal of detail from the data values.	Noise removal smooths over detail.	Invalid data fed to system mode detection, diagnostics, prognostics and ultimately plan formulation.

Table 3. Data Manipulation FFA.

6.2 Sensemaking

The main concern of system mode detection is that the wrong mode is detected; the failure modes that relate to this are shown in Section 6.2. If it is obvious that a system mode is incorrect; the operator can likely determine what the mode of the system should be.

Failure Mode	Indicative Cause(s)	Effects
(e) The wrong mode is detected.	(1) Uncertainty in system mode modelling. (2) Input sensor readings are too noisy (uncertainty in the actual sensor readings)	The SDSS provides incorrect *Diagnostics* and *Prognostics* as well as *Plan Formulation*. Furthermore, alarms may be triggered as sensors will appear abnormal with regards to the mode.
(f) The wrong mode is detected (between two similar modes).	(1) Uncertainty in system mode modelling. (2) Input sensor readings are too noisy (uncertainty in the actual sensor readings)	The SDSS provides incorrect *Diagnostics* and *Prognostics* as well as *Plan Formulation*.
(e) Algorithm detects uncertainty in system mode	(1) Insufficient system mode modelling due to uncertainty.	*Diagnostics*, *Prognostics*, *Risk Assessment* and *Plan Formulation* all have less confidence in their results.

Table 4. System Mode Detection.

Diagnostics can be used to both detect malfunctions in the system hardware (failure modes shown in Table 5) and problems in the target application. The target application, for example in oil drilling, can involve tasks such as measuring the pressure of the mud flowing through the drill pipe. Such diagnostics give a good indication whether there is a risk of the well 'kicking' (oil and gas flowing up the well uncontrollably).

Failure Mode	Indicative Cause(s)	Effects
(a) No diagnostics available.	(1) Modelling uncertainty causing a complete mismatch between input and the model. (2) Software Bug.	Other *Diagnostics* dependent components cannot execute. The system cannot provide the operator with risk assessment and formulated plans.
(e) Contradictory evidence for hardware component malfunction.	(1) Modelling uncertainty. (2) Sensor uncertainty. (3) Sensor failure.	Unable to determine whether a component is faulty or not. There is less confidence in the risk assessment.
(e) False positive failure.	(1) Modelling uncertainty. (2) Sensor uncertainty. (3) Sensor failure.	The *Risk Assessment* is incorrect impacting on the validity of Plan Formulation.
(f) Source of failure is incorrect.	(1) Modelling uncertainty. (2) Sensor uncertainty. (3) Sensor failure.	The *Risk Assessment* attributes higher risk to the wrong parts of the system and this is reflected in plan formulation.

Table 5. Diagnostics for hardware components.

6.3 Reasoning

Prognostics attempts to predict the future based on past trends and the current state. This makes it valuable in making preemptive decisions, such as scheduling maintenance or replacement of failing components before they completely fail. Prognostics may take too long to be useful, especially in a real-time context. If simply adding faster computers does not mitigate the risk that the prognostics do not complete in time (for instance in heavy simulations), then faster, less accurate, approximations may have to be used. Some of the failure modes are illustrated in Table 6.

Failure Mode	Indicative Cause(s)	Effects
(d) *Prognostic* calculations take too long.	(1) Memory contention. (2) Inadequate CPU speed. (3) Network speed is too slow.	*Risk assessment* and *Plan Formulation* will stall until prognostics calculations are complete.
(e) Incorrectly predict component failure, despite component not having problems.	(1) Modelling uncertainty. (2) Sensor uncertainty. (3) Sensor failure.	Maintenance downtime to inspect component.
(e) Failure to predict a component's failure.	(1) Modelling uncertainty. (2) Sensor uncertainty. (3) Sensor failure.	Component fails without warning.

Table 6. Prognostics.

Failure Mode	Indicative Cause(s)	Effects
(a) Algorithm produces a *Risk Assessment* with high uncertainty.	(1) Modelling uncertainty (2) Sensor uncertainty (3) Sensor failure	Confidence in the plan formulation is severely reduced.
(f) *Risk Assessment* is incorrect; the risk for the current operation is too low.	(1) Modelling uncertainty. (2) Sensor uncertainty. (3) Sensor failure.	The *Plan Formulation* suggests no action to handle the current situation as the risk is not deemed high enough.
(f) *Risk Assessment* is incorrect; the risk for the current operation is too high.	(1) Modelling uncertainty. (2) Sensor uncertainty. (3) Sensor failure.	The *Plan Formulation* suggests an action to handle the current situation although none is needed - furthermore, the actions could increase risk as a result.

Table 7. Risk Assessment.

Risk Assessment involves evaluating the risk associated with the current state of the system. This takes into consideration the mode of the system, diagnostics and prognostics that have been determined. The failure modes, as shown in Table 7, rely crucially on the completeness of the modelling of the system.

6.4 Decision Making

We assume here that *Plan Generation* generates plans using an "operational playbook" — a database of the common procedures to deal with the system state — and presents them to the operator in order of relevance. There are three main ways (expanded in Table 8) this can fail; when the operator does not get a plan, gets an incorrect plan but notices it is wrong, or gets

Failure Mode	Indicative Cause(s)	Effects
(a) No plan generated.	(1) Software bug. (2) System that executes the algorithm loses communication. (3) No viable plan can be generated (e.g. nothing is found in the Operational Playbook). (4) Execution never terminates.	The operator must make decisions without advice from the SDSS.
(b) Partial execution.	(1) Executes incompletely due to sensor failures. (2) Memory exhausted. (3) Execution timed out due to long run time. (4) Network outage midway through execution. (5) Unforeseen code bug causes algorithm to terminate prematurely.	Limited number of plans formulated - full set of combinations is not provided. Plans provided may not cover whole tradeoff space.
(e) Plan generated is not viable.	(1) Modelling uncertainty. (2) System state detection failure. (3) Operational Playbook contains mistakes.	Actions that are suggested cannot be performed on the system. The worst case is that the actions are performed until the first unviable action, which leaves the system in a irrecoverable state.
(e) Plan generated is implausible.	(1) Modelling uncertainty. (2) System state detection failure. (3) Operational Playbook does not contain a suitable set of procedures.	Actions in plan are incompatible with system state.
(f) Plan generated has incorrect sequence of actions.	(1) Modelling uncertainty. (2) System state detection failure. (3) Operational Playbook contains mistakes.	Actions performed in the wrong order leads to hazards.
(d) Plan formulation does not return a value in time for decision making.	(1) Memory contention. (2) Inadequate CPU speed. (3) Network speed is too slow.	The operator must make decisions without advice from the SDSS when there is a time-critical situation.

Table 8. Plan Formulation.

an incorrect plan and accepts it. In the first instance, the operator is left to manually define a plan. In the second instance, the operator explores the other, safer, alternatives and performs some actions for this purpose. In the third, this is the most dangerous – the SDSS must optimise plans such that they prefer safer actions that gather more information about the system to reduce the risk that an incorrect plan cannot be aborted, as well as, continuously monitor the execution of the plan to see if the plan was the correct plan to execute.

Plan Explanation produces evidence and explanation for the choice of plans. The aim of the plan explanation would be to clarify and give confidence to the operator that the suggested plans are correct. However, plan explanation could fail is numerous ways, as shown in Table 9.

Failure Mode	Indicative Cause(s)	Effects
(a) Partial explanation.	(1) Executes incompletely due to sensor failures. (2) Memory exhausted. (3) Execution timed out due to long run time. (4) Network outage midway through execution. (5) Unforeseen code bug causes algorithm to terminate prematurely.	The operator may lack confidence in a correctly suggested plan and prefer a better explained (but not necessarily good) one
(a) No explanation.	(1) Modelling uncertainty. (2) Software bug. (3) Cannot form analysis due to incomplete Operational Playbook information.	Explanation for the plan is not exposed to the operator. Operator would not be clear as to why each plan has been suggested.
(b) Information is too dense.	(1) Modelling error. (2) Software design inadequacy.	Explanation given has too much information that makes it hard to understand.
(d) Plan formulation does not return a value in time for decision making.	(1) Memory contention. (2) Inadequate CPU speed. (3) Network speed is too slow.	The operator must make decisions without advice from the SDSS when there is a time-critical situation.
(e) Explanation is invalid for the plan and is unconvincing.	(1) Modelling uncertainty. (2) System state detection failure.	The operator may lose its confidence in a correctly suggested plan and prefer a better explained plan (but not necessarily good plan)
(f) Explanation is invalid for the plan but is convincing.	(1) Modelling uncertainty. (2) System state detection failure.	The operator may gain confidence in an incorrectly suggested plan. Contradictions may only be hidden in "Detailed Explanation".

Table 9. Plan Explanation.

6.5 Presentation

We have not used FFA on the Presentation stage because Human factors specific techniques would be more appropriate.

7 Establishing A Safety Case

Establishing a valid safety case for a system that has an advanced SDSS requires an understanding of the underlying nature of the SDSS and its role in an operator's decision-making. We call *advice* anything that the SDSS suggests, from its diagnostic and prognostic information to formulated plans. Following Hawkins et al. [13], a safety case can be divided

into three complementary parts — the *safety, confidence* and *compliance* arguments.

The first need is to establish the *safety* argument — to show that the potential chains of accident causation within the system have been understood and managed appropriately. The decision-making in the system is still ultimately the operator, and the risks involved are if the operator is influenced in such a way that he/she makes the wrong decision. The intent of adding a SDSS is to add a significant positive influence on the operator's decision-making.

The SDSS has to be reliable in its suggestions and advice about plans of action. The link between decision-making and SDSS is not so simple as being simply 'reliable' because the operator may be:

(1) convinced by correct advice,
(2) convinced by incorrect advice,
(3) unconvinced by correct advice, and
(4) unconvinced by incorrect advice.

The operator may also simply ignore advice, especially if the SDSS has a reputation for being unreliable. If strenuous efforts are made to ensure that operators are 'convinced by correct advice', there is a danger that the risk of the the second, third and fourth items are increased.

As will be clear from the FFA tables given previously, uncertainty plays a very important role here. Uncertainty manifests itself in many parts of the system: inexact modelling, propagates of uncertainties in input, and the inputs to the system. But, as a concept, uncertainty (or rather certainty) is a key part of decision-making i.e. clarification of certainty leads to confidence and confidence helps with convincing.

Secondly, the safety case will need to establish *confidence* at deployment time that the safety argument is valid, in particular that the evidence from analysis and testing is adequate. This will be difficult as SDSS will need to rely on technologies that have not traditionally been used for safety-related systems.

Thirdly, a safety case will need to establish *compliance* with safety standards. Currently, most standards do not provide specific guidance on the use of Decision Support Systems in safety-critical systems. An exception is the standard IEC 61508, where the use of a Decision Support System in an advisory role is discussed [14].

Acknowledgements

We would like to thank AOS and Statoil for their support in this work.

References

[1] R. J. Mumaw, E. M. Roth, K. J. Vicente, C. M. Burns, "There is more to monitoring a nuclear power plant than meets the eye", *Human Factors: The Journal of the Human Factors and Ergonomics Society*, **42(1)**, pp. 36–55 (2000).

[2] T. Bright, A. Wong, R. Dhurjati, E. Bristow, L. Bastian, R. Coeytaux, G. Samsa, V. Hasselblad, J. Williams, M. Musty, L. Wing, A. Kendrick, G. Sanders, D. Lobach, "Effect of Clinical Decision-Support Systems: A Systematic Review", *Annals of Internal Medicine* (2012).

[3] J. Schumann, A. Srivastava, O. Mengshoel, "Who Guards the Guardians? – Toward V&V of Health Management Software", *Runtime Verification*, pp. 399–404 (2010).

[4] M. Schwabacher, K. Goebel, "A Survey of Artificial Intelligence for Prognostics", *AAAI Fall Symposium*, pp. 107–114 (2007).

[5] Y. Peng, M. Dong, M. J. Zuo, "Current status of machine prognostics in condition-based maintenance: a review", *The International Journal of Advanced Manufacturing Technology*, **50(1-4)**, pp. 297–313 (2010).

[6] N. Iyer, K. Goebel, P. Bonissone, "Framework for Post-Prognostic Decision Support", *in IEEE Aerospace Conference*, pp. 1–10, IEEE (2006).

[7] S. Frost, K. Goebel, J. Celaya, "A Briefing on Metrics and Risks for Autonomous Decision-making in Aerospace Applications", *in Infotech@Aerospace 2012*, Infotech@Aerospace Conferences, American Institute of Aeronautics and Astronautics (June 2012).

[8] J. Schumann, O. J. Mengshoel, A. N. Srivastava, A. Darwiche, "Towards Software Health Management with Bayesian Networks", *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pp. 331–336 (2010).

[9] C. Pace, D. Seward, "A Model for Autonomous Safety Management in a Mobile Robot", *in International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce 2006*, volume 1, pp. 1128–1133, IEEE (2005).

[10] D. J. Pumfrey, "The Principled Design of Computer System Safety Analyses", Ph.D. thesis, University of York (1999).

[11] G. Klein, B. Moon, R. R. Hoffman, "Making sense of sensemaking 1: Alternative perspectives", *Intelligent Systems, IEEE*, **21(4)**, pp. 70–73 (2006).

[12] Society of Automotive Engineers, "Guidelines and Methods for Conducting the Safety Assessment Process on Airborne Systems and Equipment", Technical report, Society of Automotive Engineers (1996).

[13] R. Hawkins, T. Kelly, J. Knight, P. Graydon, "A new approach to creating clear safety arguments", *in Advances in Systems Safety*, pp. 3–23, Springer (2011).

[14] European Committee for Electrotechnical Standardization, *BS EN 61508-3 : 2010 Functional safety of electrical / electronic / programmable electronic safety-related systems*, BSI Standards Publication, Brussels (2010).