Deriving Hierarchical Safety Contracts

Omar Jaradat[1]
[1]School of Innovation, Design, and Engineering
Mälardalen University
Box 883, 721 23 Västerås, Sweden
Email: omar.jaradat@mdh.se
Telephone: +46 (21) 10-1369, Fax: +46 (21) 10-1460

Iain Bate[1,2]
[2]Department of Computer Science
University of York
Deramore Lane, Heslington, York, YO10 5GH, UK
Email: iain.bate@york.ac.uk
Telephone: +44 (1904) 325572, Fax: +44 (1904) 325599

***Abstract*—Safety cases need significant amount of time and effort to produce. The required amount of time and effort can be dramatically increased due to system changes as safety cases should be maintained before they can be submitted for certification or re-certification. Sensitivity analysis is useful to measure the flexibility of the different system properties to changes. Furthermore, contracts have been proposed as a means for facilitating the change management process due to their ability to record the dependencies among system's components. In this paper, we extend a technique that uses a sensitivity analysis to derive safety contracts from Fault Tree Analyses (FTA) and uses these contracts to trace changes in the safety argument. The extension aims to enabling the derivation of hierarchical and correlated safety contracts. We motivate the extension through an illustrative example within which we identify limitations of the technique and discuss potential solutions to these limitations.**

Keywords–Sensitivity Analysis, Safety Case, Safety Argument, Safety Contracts, Change Management.

I. Introduction

The concept of a safety case originated in 1989 when the UK Health and Safety Executive (HSE) requested from the British nuclear sites to generate a written report — according to the Control of the Industrial Major Accident Hazards (CIMAH) regulations — that should contain: (1) facts about the site, and (2) reasoned arguments about the hazards and risks from the site [1]. This report is known as a safety case, which should systematically demonstrate a reasoned argument that a nuclear site is acceptably safe to operate. More specifically, a safety case should identify the major hazards and risks in a nuclear site and demonstrate that the site is satisfactory since all of these hazards and risks are adequately mitigated. The increasing size and complexity of safety critical systems motivate the application of the safety case concept in different domains (e.g., oil, avionics, railway, automotive, etc.) to demonstrate a reasoned argument that those systems are acceptably safe to operate.

Safety certification is typically imposed by authorities as a censorship procedure to control the development of safety critical systems. The certification processes are based on an evaluation of whether the hazards associated with a system are mitigated to an acceptable level. Developers must provide evidence of safety to their regulators. Safety cases can explain how this evidence shows that the system is acceptably safe to operate. Hence, the development of safety cases has become common practice.

The certification process is amongst the most expensive and time-consuming tasks in the development of safety critical systems. A key reason behind that is the increasing complexity and size of these systems combined with their growing market demands. Moreover, safety critical systems are expected to operate for a long period of time and frequently subject to changes during both development and operational phases. Any change that might compromise system safety involves repeating the certification process (i.e., re-certification) and thus, ultimately, necessitates maintaining the corresponding safety case. For example, the UK Ministry of Defence *Ship Safety Management System Handbook JSP 430* requires that "The safety case will be updated ... to reflect changes in the design and/or operational usage which impact on safety, or to address newly identified hazards. The safety case will be a management tool for controlling safety through life including design and operation role changes" [2], [3]. Similarly, the UK HSE *Railway safety case regulations 1994* states in regulation *6(1)* that "A safety case is to be revised whenever, appropriate that is whenever any of its contents would otherwise become inaccurate or incomplete" [4], [3].

One of the biggest challenges that affects safety case revision and maintenance is that a safety case comprises a complex web of interdependent elements. That is, safety goals, evidence, argument, and assumptions about operating context are highly interdependent and thus, seemingly minor changes may have a major impact on the contents and structure of the safety argument. As such, a single change to a safety case may necessitate many other consequential changes — creating a ripple effect [5]. For example, if a new system component was integrated into a system, old items of evidence might no longer support the developers' claims about components' consistency because these claims reflect old assumptions in the development artefacts that do not take into consideration the new component integration.

In order to maintain a safety case after implementing a system change, system developers need to assess the impact of changes on the original safety argument. The assessment shall include reviewing the relevant assumptions made in the argument, and examining the adequacy of the collected body of evidence. For example, the UK Defence *Standard DS 00-56* states that: "Any amendments to the deployment of the system should be examined against the assumptions and objectives contained in the safety case" [6], [3]. Hence, a step to assess the impact of this change on the safety argument is crucial and highly needed prior to updating a safety argument after a

system change. Despite clear recommendations to adequately maintain and review safety cases by safety standards, existing standards offer little advice on how such operations can be carried out [5]. If developers do not understand the impact of change then they have to be conservative and do wider verification (i.e., check more elements than strictly necessary). This increases the maintenance cost.

Modularity has been proposed as the key element of the 'way forward' in developing systems [7]. For modular systems, the required maintenance efforts to accommodate predicted changes can be less than the required efforts to accommodate arbitrary changes. This is because having a list of predicted changes during the system design phase allows system engineers to contain the impact of each of those changes in a minimal number of system's modules. Hence, predicted changes can have traceable consequences as engineers will be aware of how a change in one module can result in a change in another module. In practice, it is hard to align the safety case structure with the system's modules [8]. However, a well-established traceability between system's modules and its safety case can provide the same traceable consequences of changes in the safety case. The problem though is that system changes and their details cannot be fully predicted and made available up front. In particular, software aspects of the safety case is hard to be predicted as software is highly changeable and harder to contain. In this paper, we use sensitivity analysis based approach to assist system's engineers to predict changes.

In our previous work [8], we introduced a technique that contains Sensitivity ANalysis for Enabling Safety Argument Maintenance (SANESAM) phase that supports system engineers to anticipate potential changes. The key principle of SANESAM phase is to determine the flexibility (or compliance) of a system to changes using sensitivity analysis. The output is a ranked list of FTA events that system engineers can refine. The result after the refinement is a list of contracts that can be used as part of later change impact analysis. We also use safety contracts to record the information of changes that will ultimately advise the engineers what to consider and check when changes actually happen. The main contribution of this paper comprises (1) identifying possible limitations for SANESAM, and (2) suggest an a SANESAM extension to resolve the identified limitations. The paper uses the hypothetical aircraft Wheel Braking System (WBS) to illustrate SANESAM extension.

This paper is composed of four further sections. In Section II we present background information about sensitivity analysis, safety contracts, Goal Structuring Notations (GSN), incremental certification and WBS description. In Section III, we give an overview of a technique to facilitate the maintenance of safety cases and identify limitations. In Section IV, we suggest extending the technique to resolve the identified limitations, we also use the WBS system to illustrate the extensions. Finally, we conclude and propose future works in Section V.

II. Background

A. Sensitivity Analysis

Sensitivity analysis can be defined as: "The study of how uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input" [9]. The analysis helps to establish reasonably acceptable confidence in the model by studying the uncertainties that are often associated with variables in models [10]. There are different purposes for using sensitivity analysis, such as, providing insight into the robustness of model results when making decisions [11]. For instance, sensitivity analysis can be used to determine what level of accuracy is necessary for a parameter (variable) to make the model sufficiently useful and valid [12].The analysis can be also used to enhance communication from modelers to decision makers, for example, by making recommendations more credible, understandable, compelling or persuasive [13]. The analysis can be performed by different methods, such as, mathematical, graphical, statistical, etc.

Emberson et al. [14] use sensitivity analysis to improve the flexibility of task allocation in real-time system design. More specifically, the analysis is used to evaluate the impact on task allocation solution after applying possible change scenarios to task allocation framework.

In this paper, we apply the sensitivity analysis on FTAs to measure the sensitivity of outcome A (e.g., a safety requirement being true) to a change in a parameter B (e.g., the failure probability in a component). The sensitivity is defined as $\Delta B/B$, where ΔB is the smallest change in B that changes A (e.g., the smallest increase in failure probability that makes safety requirement A false). The failure probability values that are attached to FTA's events are considered input parameters to the sensitivity analysis. A sensitive part of a FTA is defined as one or multiple FTA events whose minimum changes (i.e., the smallest increase in its failure probability due to a system change) have the maximal effect on the FTA, where effect means exceeding failure probabilities (reliability targets) to inadmissible levels. A sensitive event is an event whose failure probability value can significantly influence the validity of the FTA once it increases. A sensitive part of a FTA is assigned to a system design component that is referred to as a sensitive component in this paper. Changes to a sensitive component cause a great impact to system design. [8]

B. Safety Contracts

The concept of contract is familiar in software development. For instance, For instance, Design by Contract (DbC) was introduced in 1986 [15], [16] to constrain the interactions that occur between objects. Contract-based design is an approach where the design process is seen as a successive assembly of components where a component behaviour is represented in terms of assumptions about its environment and guarantees about its behavior [17]. In the context of contract-based design, a contract is conceived as an extension to the specification of software component interfaces that specifies preconditions and postconditions to describe what properties a component can offer once the surrounding environment satisfies one or more related assumption(s). A contract is said to be a *safety contract* if it guarantees a property that is traceable to a hazard. There have been significant works that discuss how to represent and to use contracts [18], [19], [20]. In the safety critical systems domain, researchers have used, for example, assume-guarantee contracts to propose techniques to lower the cost of developing software for safety critical

systems. Moreover, contracts have been exploited as a means for helping to manage system changes in a system domain or in its corresponding safety case [21], [22], [23].

The following is an example that depicts the most common used form of contracts:

> **Guarantee**: The WCET of task X is $\leq$ 10 milliseconds
> **Assumptions**:
> X is:
> 1) compiled using compiler $[C]$,
> 2) executed on microcontroller $[M]$ at 1000 MHz with caches disabled, and
> 3) not interrupted

C. Safety Argumentation and Goal Structuring Notations (GSN)

GSN was introduced to provide a graphical means of communicating (1) safety argument elements, claims (goals), argument logic (strategies), assumptions, context, evidence (solutions), and (2) the relationships between these elements [24]. The principal symbols of the notations (with example instances of each concept) are shown in Figure 1. A goal structure shows how goals are successively broken down into (solved by) sub-goals until a point is reached where claims can be supported by direct reference to evidence. Using GSN, the writer can clarify the adopted argument strategies (i.e., how the premises imply the conclusion), the rationale for the approach (assumptions, justifications) and the context in which goals are stated [8]. GSN has been extended to enable modularity in safety cases (i.e., module-based development of the safety case) so that it enables the partitioning of a safety case into an interconnected set of modules.

Well-structured argument may help the developers to mechanically propagate the change through the goal structure. However, it does not tell if the suspect elements of the argument in question are still valid. For example, having made a change to a model we must ask whether goals articulated over that model are still valid. Expert judgment is still required in order to answer such questions [25]. Hence, merely having well-structured arguments does not directly help to preserve the soundness of the argument after a change, but it can more easily determine the questions to be asked to do so.

D. Incremental Certification

The Industrial Avionics Working Group (IAWG) — a consortium of researchers and practitioners — has proposed Modular Safety Cases as a means of containing the cost of change by dividing the safety case into a set of argument modules. IAWG's Modular Software Safety Case (MSSC) process [26] facilitates handling system changes as a series of relatively small increments rather than occasional major updates. The key principle of the state-of-the-art process is to modularise a safety case so as to contain changes within a minimal area of the safety case [26]. More specifically, the process starts by anticipating potential changes over the lifetime of a system. System developers modularise the argument so as to contain the impact of the anticipated changes. Hence, MSSC process ensures that the maximum amount of safety case material that was previously certified is not impacted, and thus it is available for re-use in the re-certification process without a need to be revisited [26].

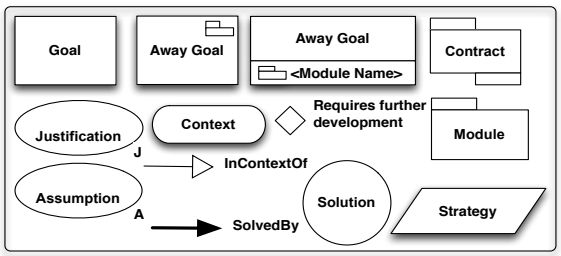

Figure 1. Overview of GSN Notations

E. Wheel Braking System (WBS): System Description

The WBS is a hypothetical aircraft braking system described in Appendix L of a popular standard for safety assessment processes, ARP4761 [27]. Figure 2 shows a high-level architecture view of the WBS. The system is installed on the two main landing gears of a civil air transport. The main function of the system is to provide wheel braking as commanded by the pilot when the aircraft is on the ground. The system is composed of three main parts: 1) Computer-based part which is called the Brake System Control Unit *(BSCU)*, 2) *Hydraulic* part, and 3) *Mechanical* part. The *BSCU* is internally redundant and consists of two channels, *BSCU System 1 and 2* (*BSCU* is the box in the gray background in Figure 2). Each channel consists of two components: *Monitor* and *Command*. *BSCU System 1 and 2* receive the same pedal position inputs, and both calculate the command value. The two command values are individually monitored by the *Monitor 1 and 2*. Subsequently, values are compared and if they do not agree, a failure is reported. The results of both *Monitors* and the compared values are provided to a the *Validity Monitor*. A failure reported by either system in the *BSCU* will cause that system to disable its outputs and set the *Validity Monitor* to invalid with no effect on the mode of operation of the whole system. However, if both monitors report failure, the *BSCU* is deemed inoperable and is shut down [8], [27], [28]. Figure 2 shows high-level view of the *BSCU* implementation and it omits many details.

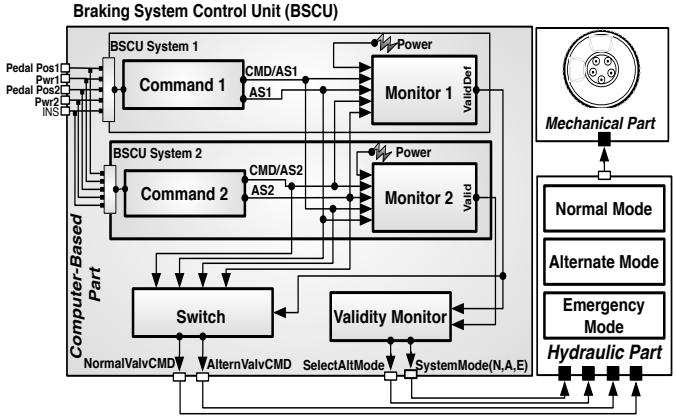

Figure 2. A high-level view of the WBS [8]

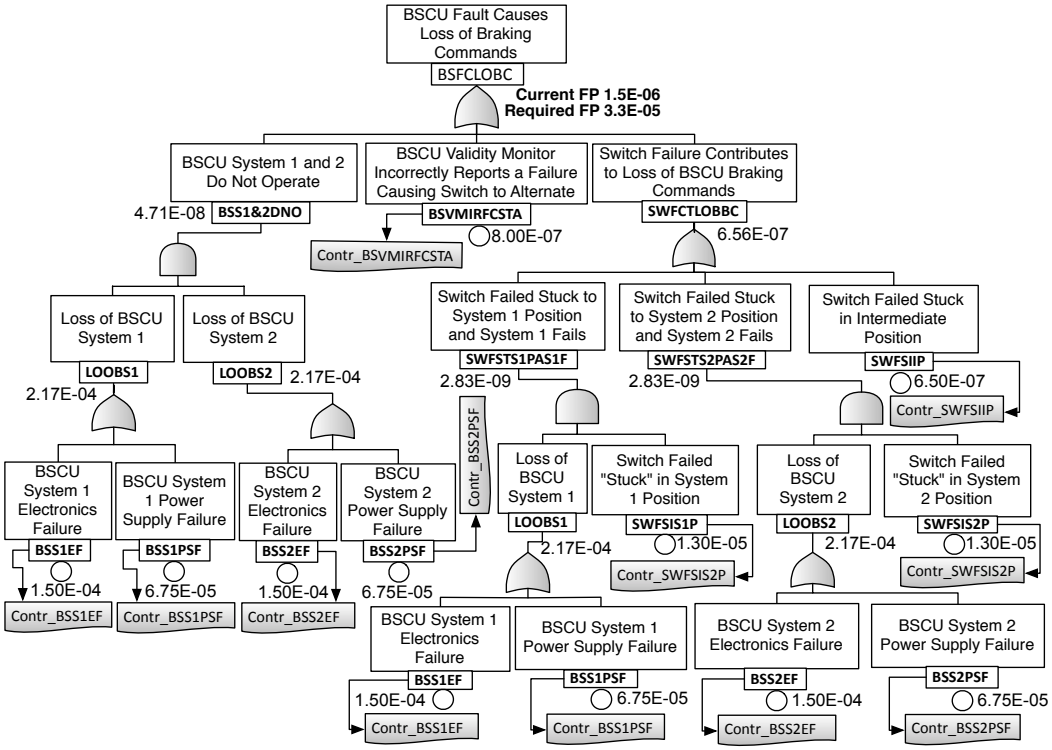

Figure 3. Loss of Braking Commands FTA [8]

However, the figure is still sufficient to illustrate key elements of our technique. More details about the *BSCU* implementation can be found in ARP-4761 [27]. Figure 3 shows the "Loss of Braking Commands" probabilistic FTA.

III. A Technique to Facilitate the Maintenance of Safety Cases

In this section we give an overview of a technique that aims to facilitate the maintenance of a safety case. The technique comprises 7 steps that are distributed between the Sensitivity ANalysis for Enabling Safety Argument Maintenance (SANESAM) phase and the safety argument maintenance phases as shown in Figure 4. The steps of the SANESAM phase are represented along the upper path, whilst the lower path represents the steps of the safety argument maintenance phase. Considering a complete list of anticipated changes is difficult. This technique uses sensitivity analysis to measure the flexibility of system components to changes. That is, regardless of the type of changes it will be seen as factors to increase or decrease a certain parameter value. Thus system developers can focus more on predicting those changes that might make the parameter value inadmissible [8]. Furthermore, the technique utilises the concept of contracts to record the information of changes that will ultimately advise the engineers what to consider and check when changes actually happen.

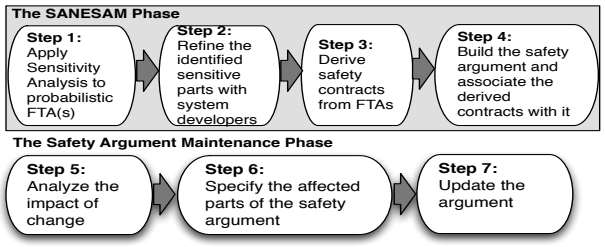

Figure 4. Process diagram of safety cases maintenance technique

A. SANESAM Phase

The rationale of this phase is to determine, for each component, the allowed range for a certain parameter within which a component may change before it compromises a certain system property (e.g., safety, reliability, etc.). Sensitivity analysis is used in this phase as a method to determine the range of failure probability parameter for each component. The technique assumes the existence of a probabilistic FTA where each event in the tree is specified by a current estimate of failure probability $FP_{Current|event(x)}$. In addition, the technique assumes the existence of the required failure probability for the top event $FP_{Required(Topevent)}$, where the FTA is considered unreliable if: $FP_{Currentl(Topevent)} > FP_{Required(Topevent)}$. [8]

The steps of SANESAM phase are as follows: [8]

- ***Step 1. Apply the sensitivity analysis to a probabilistic FTA***: In this step the sensitivity analysis is applied to a FTA to identify the sensitive events whose minimum changes have the maximal effect on the $FP_{Topevent}$. Identifying those sensitive events requires the following steps to be performed:
  1) Find the Minimal Cut Set (MC) in the FTA. The minimal cut set definition is: *"A cut set in a fault tree is a set of basic events whose (simultaneous) occurrence*

ensures that the top event occurs. A cut set is said to be minimal if the set cannot be reduced without losing its status as a cut set" [29].

2) Calculate the maximum possible increment in the failure probability parameter of event x before the top event $FP_{Required(Topevent)}$ is no longer met, where $x \in MC$ and $(FP_{Increased|event(x)} - FP_{Current|event(x)}) \not\Rightarrow FP_{Increased(Topevent)} > FP_{Required(Topevent)}$.
3) Rank the sensitive events from the most sensitive to the less sensitive. The most sensitive event is the event for which the following formula is the minimum:

$$\frac{FP_{Increased|event(x)} - FP_{Current|event(x)}}{FP_{Current|event(x)}.}$$

- ***Step 2. Refine the identified sensitive parts with system developers***: In this step, the generated list of sensitive events from Step 1 should be discussed by system developers (e.g., safety engineers) as they should choose the sensitive events that are most likely to change. The list can be extended to add any additional events by the developers. Moreover, it is envisaged that some events might be removed from the list or the rank of some of them might change.
- ***Step 3. Derive safety contracts from FTAs***: In this step, a safety contract or contracts should be derived for each event in the list from Step 2. The main objectives of the contracts are to 1) highlight the sensitive events to make them visible up front for developers attention, and 2) to record the dependencies between the sensitive events and the other events in the FTA. Hence, if the system is later changed in a way that increases the failure probability of a contracted event where the increased failure probability is still within the defined threshold in the contract, then it can be said that the contract(s) in question still hold (intact) and the change is containable with no further maintenance. The contract(s), however, should be updated to the latest failure probability value. On the other hand, if the change causes a bigger increment in the failure probability value than the contract can hold, then the contract is said to be broken and the guaranteed event will no longer meet its reliability target. It is worth noting that the role of safety contracts in SANESAM is to highlight sensitive events, and not to enter new event failure probabilities. We introduce a new notation to FTAs to annotate the contracted events, where every created contract should have a unique identifier, see Figure 5-a. Figure 3 shows the derived safety contracts as a result of SANESAM application to the Loss of Braking Command FTA. We also create a template to document the derived safety contracts. Figure 5-b shows an instantiation of the contents of one of the derived safety contracts for WBS.
- ***Step 4. Build the safety argument and associate the derived contracts with it***: In this step, a safety argument should be built and the derived safety contracts should be associated with the argument elements. Associating the contracts with GSN goals is done by using the introduced notation in Figure 5-a.

B. SANESAM Limitations

The essence of SANESAM is to calculate the maximum possible increment in the failure probability parameter of only one event at a time before the top event $FP_{Required(Topevent)}$ is no longer met. In this section, we identify three limitations of the current version of SANESAM and we give an example for each limitation.

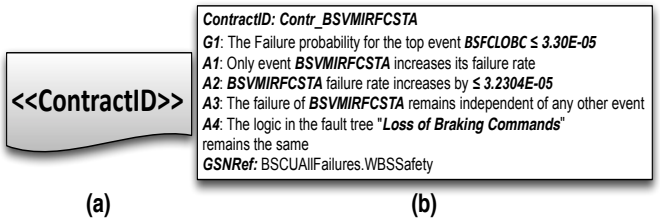

Figure 5. (a) FTA Safety contract notation, (b) Derived safety contract

1) No Support for Intermediate Events: The followed method for applying sensitivity analysis relies on the calculated cut set for the full FTA. Hence, only basic events are considered during the application of sensitivity analysis and no contracts are derived for the intermediate events. Figure 6-a shows an example of this limitation, where LOOBS1 is an intermediate event and based on SANESAM it cannot be provided as a sensitive event thus no contract can be derived for it. Having said that, system developers may need to contain the impact of changes in intermediate levels to prevent them from rippling through the top-level event. Additionally, some events might look trivial for system engineers but if those events were packed in events from higher levels, then they could look nontrivial. For example, providing system engineers with "Jam of speedbrake lever" as a sensitive event to a particular change might look less serious than the parent event "Mechanical failures of speedbrake lever". Another motivation for deriving contracts on intermediate levels comes from the fact that some intermediate events may represent top goals in the safety case modules which will be more supportive for incremental certification as introduced in Section II-D. In other words, pinpointing the entire safety case module as affected is easier than starting from intermediate goals in that module. From the forgoing reasons, SANESAM should be able to provide system engineers with sensitive events from different levels of the FTA's hierarchy.

2) No Support for Multiple Events Impact: SANESAM calculates the highest possible boundary of failure probabilities for certain events. SANESAM also assumes independence of events and does not address the problem of event interdependencies that is typical for any realistic system. This means that only one event failure probability is allowed to increase per change. However, a change might impact multiple events in the same time. For instance, adding distinct functional redundancy of a critical software component might decrease the failure probability of multiple events in the FTA. Likewise, removing a redundant component to make the system simpler and cheaper might increase the failure probability of multiple events. Since the failure probabilities of multiple components often change at once, a SAMESAM extension to handle such changes is highly desirable. A clear example can be given by assuming a change to BSCU System 1 power supply in Figure 6-a. A change to BSCU System 1 power supply will necessitate a correlated change to BSCU System 2 power supply. Hence,

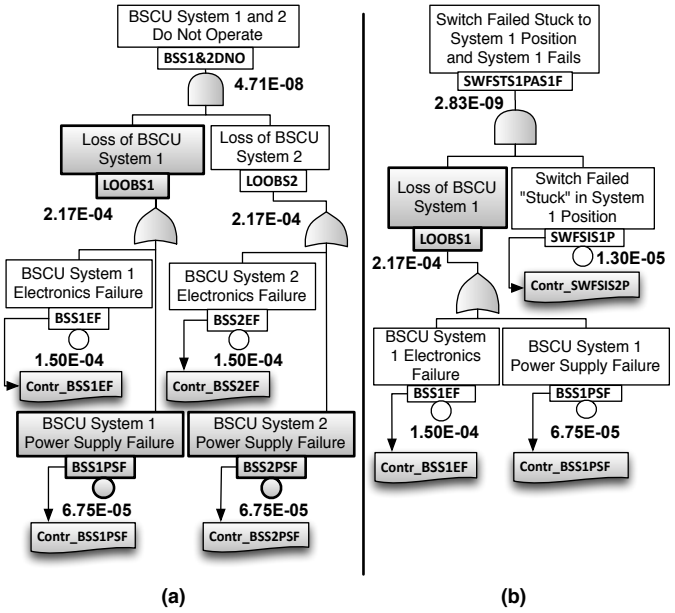

Figure 6. Limitation examples

when we need to calculate the possible increment to the failure probability of BSS1PSF (for this specific change), we must take into account the correlation between BSS1PSF and BSS2PSF.

3) Neglecting Duplication: It is possible for an event to be represented in more than one location in the same FTA. For example, LOOBS1 event is represented twice in this FTA. In the first representation (Figure 6-a) it is combined with LOOBS2 by AND gate, where both events have the same failure probability. In its second representation (Figure 6-b), LOOBS1 is combined with SWFSIS1P by AND gate, both events have different failure probabilities. Hence, the possible increment on LOOBS1 failure probability will vary in the two locations. SANESAM neglects events duplication, and this is considered a limitation because the calculated possible failure probability increment of the same event may vary in the same FTA if the event is duplicated. Calculating the possible increment on the failure probability of a duplicated event is based on the failure probability(s) of the combined events. More clearly, in each duplication of an event, the event may be combined with different event(s), different failure probability values, and by different gates (e.g., AND, OR, XOR, etc.).

IV. SANESAM Extension

In this section, we suggest extending SANESAM to resolve the limitations that were identified in Section III-B. The extended SANESAM is referred to as SANESAM+ in this paper. SANESAM and SANESAM+ are mutually exclusive and selecting among them is very dependent on the refined list by system engineers in Step 2 of the technique (described in Section III-A). More clearly, if at least one of the events in the refined list is duplicated, or if its change necessitates a correlated event to change, then SANESAM+ is the one to go. Otherwise, developers are free to choose SANESAM or SANESAM+. However, choosing SANESAM means that the developers accept the assumption that only one event is allowed to change at a time.

SANESAM relies on the calculated MC (minimum cut set) for the full FTA which means that only the basic events are considered for sensitivity analysis. However, SANESAM+ requires measuring the sensitivity of all events in the FTA. This means that we need to calculate the Maximum Allowed Failure Probability (MAFP) for each event in the FTA taking into account that all events may change at a time. That is, $\Delta FP_{(Topevent)}$ = ($FP_{Required(Topevent)} - FP_{Current(Topevent).}$) will be distributed over all FTA's events, where $\Delta FP_{(Topevent)} > 0$.
In order to apply SANESAM+ and calculate the MAFP for FTA events, we replace the procedure of Step 1 in Section III-A with the following procedure:

1) Find the difference between new and current FPs of the ancestor events, as follows:

$$\Delta FP_{(Ancestor)} = FP_{New} - FP_{Current}$$

The first run of this step should start with $\Delta FP_{(Topevent)}$, where the new FP in this specific case is the required FP. The second run should be for each event in the very next level and so on and so forth until the basic events are reached.
2) This sub-step is very dependent on the type of the gate between the ancestor and descendant events. In case of OR gate, sub-steps 2-A and 2-B should be followed. In case of AND gate, sub-step 2-C should be followed.
   a) Find the ratio of the descendant events to the ancestor event. The first run of this step should start with the top event and the events beneath it. The second run of this step should consider one more level down. In other words, descendant events in the first run will become ancestors in the second one. The ratio of a descendant event to its ancestor is calculated by Equation 1, as follows:

$$Ratio_{Desc(x)} = \frac{FP_{Current(Desc(x))}}{FP_{Current(Ancestor)}} \quad (1)$$

   b) Increase the FP for each of the descendant events by $\Delta FP_{(Ancestor)}$ which is calculated in step 1. Increasing events' FP is done by Equation 2, as follows:

$$FP_{Increased|Desc(x)} = FP_{Current(Desc(x))} + (Ratio_{(Desc(x))} * \Delta FP_{(Ancestor)}) \quad (2)$$

   c) In this sub-step, we need to distribute the increment to the FP of an ancestor event over its descendent events in the presence of an AND gate. The increment to each descendant event is calculated in two different ways based on the number of descendent events and if their FPs vary.

   Case 1. if the events share the same FP value, we can use: $\sqrt[n]{FP_{(Increased|Ancestor)}}$, where n is the number of the descendent events.
   LOOBS1 and LOOBS2 in Figure 6-a represents an example of this case.
   Case 2. if the descendent events do not share the same FP, then $FP_{(Topevent)}$ is distributed over them unevenly, but rather based on the FP ratio of every

descendent event to $\Delta FP_{(Ancestor)}$ as described by Equation 3:

$$FP_{Current(Desc(x))} + (\frac{FP_{Current(Desc(x))}}{\sum FP_{Current(AllDesc)}} * I) \quad (3)$$

In order to determine I we need to consider all sibling events as described in Equation 4:

$$\begin{aligned}(FP_{Current(Desc(x_1))} + (\frac{FP_{Current(Desc(x_1))}}{\sum FP_{Current(AllDesc)}} * I)) \\ *(FP_{Current(Desc(x_2))} + (\frac{FP_{Current(Desc(x_2))}}{\sum FP_{Current(AllDesc)}} * I)) \\ *(FP_{Current(Desc(x_n)} + (\frac{FP_{Current(Desc(x_n))}}{\sum FP_{Current(AllDesc)}} * I)) \\ = FP_{Increased(Ancestor)}\end{aligned} \quad (4)$$

LOOBS1 and SWFSIS1P in Figure 6-b represent an example of this case.

3) Repeat steps 1 and 2 until FP of the basic events get increased. Unlike SANESAM, SANESAM+ distinguish between duplicated events. That is, if an event shows up in multiple locations in the FTA, we still need to calculate its FP wherever we encounter it. Later on when finish calculating the FP for all duplicates of an event we unify the its FP by considering the minimum calculated FP of them.
4) Finally, rank the sensitivity of events from the most sensitive to the less sensitive. The most sensitive event is the event for which Equation 5 is the minimum, as follows:

$$Sensitivity_{(x)} = \frac{FP_{Increased(x)} - FP_{Current(x)}}{FP_{Current(x)}} \quad (5)$$

It is worth noting that the difference between the steps of SANESAM and SANESAM+ is observed only in Step 1, all other later steps are identical.

A. SANESAM+ Application: An Example

In this section, we use the Loss of Braking Commands FTA (in Figure 3) to show an application example of SANESAM+.

1) Find $\Delta FP_{(Ancestor)}$ (which is the top event in the FTA for the first of this sub-step): $\Delta\ FP_{(BSFCLOBC)}$ = 3.30E-05 - 1.5031E-06
   $\Delta\ FP_{(BSFCLOBC)}$ = 3.14969E-05
2) Since BSFCLOBC is correlated with its descendants (i.e., BSS1&2DNO, BSVMIRFCSTA and SWFCTLOBBC) via OR gate, then sub-steps 2-A and 2-B should be followed.
   a) We need to find FP ratio for each of BSS1&2DNO, BSVMIRFCSTA and SWFCTLOBBC to BSFCLOBC using Equation 1.

   Example: BSS1&2DNO
   $Ratio_{BSS1\&2DNO}$ = $\frac{4.71E\text{–}08}{1.5031E\text{–}06}$

   $Ratio_{BSS1\&2DNO}$ = 3.133524050E-02.
   b) In this sub-step, $\Delta FP_{(BSFCLOBC)}$ should be distributed over each of BSS1&2DNO, BSVMIRFCSTA and SWFCTLOBBC based on their ratios to $FP_{Actaul(Ancestor)}$ using Equation 2.

   Example: BSS1&2DNO
   $FP_{Increased(BSS1\&2DNO)}$ =
   4.71E-08 + (3.133524050E-02 * 3.14969E-05)
   $FP_{Increased(BSS1\&2DNO)}$ = 1.034062937E-06

   c) Now, let us take other examples where AND gate correlates an ancestor event with its descendent events. The example covers Case 1 and 2 as described in sub-step 2-C in Section IV.

   Example of Case 1: LOOBS1 and LOOB2.
   $FP_{(Increased(x)}$ = $\sqrt[2]{FP_{(Increased|BSS1\&2DNO)}}$
   $FP_{(Increased|LOOBS1)}$ = 1.016889E-03
   $FP_{(Increased|LOOBS2)}$ = 1.016889E-03

   Example of Case 2: LOOBS1 and SWFSIS1P
   In this example, the FP of LOOBS1 and SWFSIS1P are increased using Equations 3 and 4.

$$\begin{aligned}= (2.17E\text{–}04 + (\frac{2.17E\text{–}04}{2.17E\text{–}04 + 1.30E\text{–}05} * I)) * \\ (1.30E\text{–}05 + (\frac{1.30E\text{–}05}{1.30E\text{–}05 + 2.17E\text{–}04} * I)) \\ = 6.216381375E\text{–}08\end{aligned}$$

   $FP_{(Increased|LOOBS1)}$ = 1.02E-03
   $FP_{(Increased|SWFSIS1P)}$ = 6.10E-05
3) In this step, we repeat the 1 and 2 steps until FP of the basic events get increased. Figure 7 shows the calculated FPs (in boxes) using SANESAM+. Looking at the figure, It should be observed that the events, BSS1EF, BSS2EF, BSS1PSF and BSS2PSF are duplicated. These events were assumed independent from each other while calculating their FPs. However, this assumption was vanished after the calculation and the minimum FP (values between brackets in Figure 7) was considered the maximum possible FP for each duplicate events. For example, two FP values were obtained for BSS1EF in different locations (i.e., 4.56E-04 and 3.167E-04) but since 3.167E-04 is the minimum FP value of the two duplicates, it is, therefore, the maximum possible FP for all BSS1EF's duplicates.
4) In this step, we use Equation 5 in Section IV. Table I shows the results of the sensitivity analysis and the ranking of the events' sensitivity where 1 is the most sensitive event.

B. SANESAM+ For Predicted Changes

SANESAM+ can be useful even for arbitrary changes. That is, even if the system engineers are not sure of the potential future changes, SANESAM+ enable the derivation of safety contracts for all events in different levels in the FTA. Hence, when a change request shows up, system engineers, and by returning to the sensitivity results, can decide whether the effect of the change is tolerable or not. However, SANESAM+ can be more useful in the presence of a predicted change as it can increase the effect tolerance of that change. More clearly, distributing $\Delta FP_{(Topevent)}$ over all FTA's events might increase the change impact tolerance of some events that are

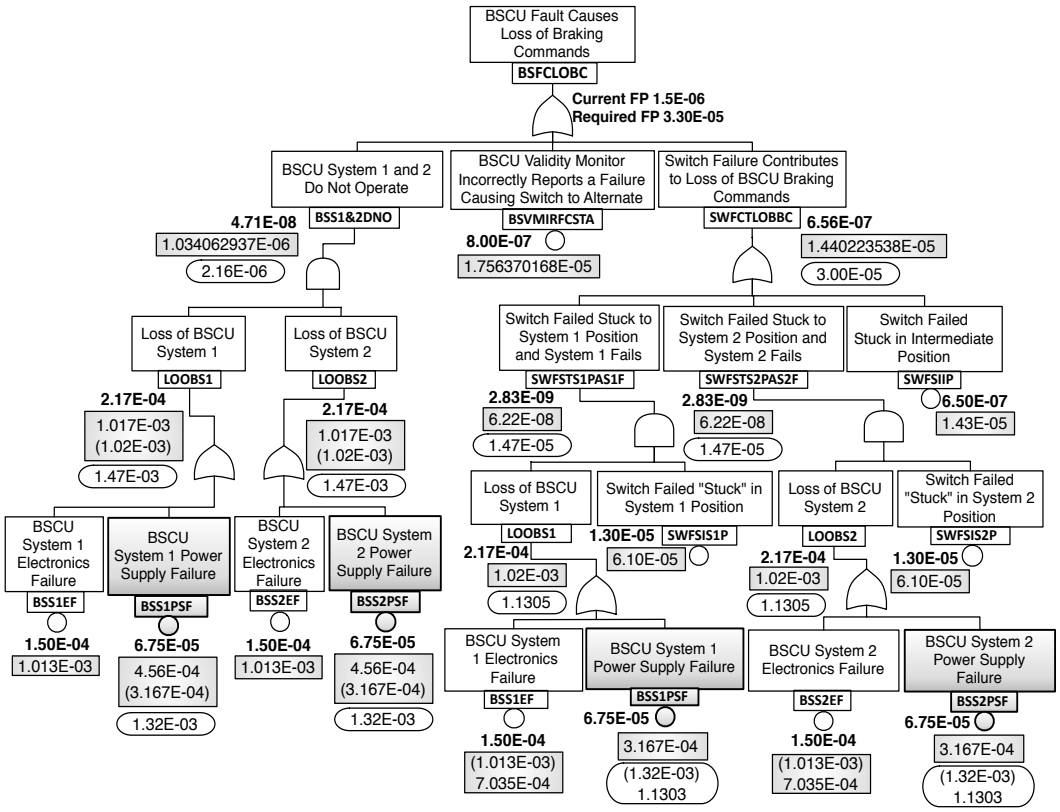

Figure 7. Loss of Braking Commands FTA [8]

unlikely to change. On the other hand, the change impact tolerance might be slightly increased for events that are more likely to change. Consequently, having a change scenario in advance will motivate increasing the change impact tolerance for only the events that fall in the scope of that change. Since, however, SANESAM+ (for predicted changes) will exclude the events that are unlikely to change, we will slightly modify the steps by which we calculate the FP of events. The following steps give guidance on how to calculate the FP SANESAM+ for predicted change scenarios:

TABLE I. The results of SANESAM+ sensitivity analysis

Event Name	Current FP	Increased FP	Sensitivity	Rank
SWFSIS1P SWFSIS2P	1.30E-05	6.10E-05	3.692307692	2
LOOBS1 LOOBS2	2.17E-04	1.02E-03	3.700460829	3
BSS1EF BSS2EF	1.50E-04	1.013E-03	5.753333333	4
BSS1PSF BSS2PSF	6.75E-05	3.167E-04	3.69185185	1
SWFCTLOBBC	6.56E-07	1.44E-05	20.95121951	5
SWFSTS1PAS1F SWFSTS2PAS2F	2.83E-09	6.22E-08	20.97879859	6
BSVMIRFCSTA	8.00E-07	1.76E-05	21	7
SWFSIIP	6.50E-07	1.43E-05	21	7
BSS1&2DNO	4.71E-08	1.04E-06	21.08067941	8

1) Find the difference between the current and required FP of the top event $\Delta FP_{(Topevent)}$.
2) Find the highest event that contains the effect. If the highest event does not fall directly under the top event, the effect should be traced up the fault tree further until we reach the affected event that falls directly under the top event.
3) Distribute the calculated $\Delta FP_{(Topevent)}$ in sub-step 1 to the identified events in sub-step 2 based on the determined ratio in sub-step 3. The first run of this sub-step should start with the top event and the events beneath it, and the second runshould consider one more level down.This sub-step is very dependent on the type of the gate between the ancestor and descendant events. In the case of an OR gate sub-step 4-A should be followed. In the case of an AND gate sub-step 4-B should be followed.
   a) In this sub-step, we need to distribute the increment to the FP of an ancestor event over its descendent affected events in the presence of an OR gate. We first need to find the ratio of the affected event to its ancestor event. Afterwards, we need to use the calculated ratio to determine the amount of the increment to the affected event. The first run of this step should start with the affected events that fall directly under the top event. The second run of this step should consider one more level down. In other words, descendant events in the first run will become ancestors in the second one. The simplest FP calculation is when to have two descendent events and only one of them is affected. This is because all what we need to do is to subtract

the unaffected FP from the increased ancestor event to get the the increased FP of the affected event as presented in Equation 6:

$$FP_{Increased(Ancestor)} - FP_{Current(Unaffect|Desc(x))} = FP_{Increased(Desc(x))} \quad (6)$$

Otherwise, the ratio of a descendant event to its ancestor and the granted increment to an affected event is calculated by Equation 7 as follows:

$$FP_{Increased(Desc(x))} = (\frac{FP_{Current(Desc(x))}}{FP_{Current(Ancestor)} - \sum FP_{Current(Unaffect)}} * FP_{Increased(Ancestor)}) + FP_{Current(x)} \quad (7)$$

b) In this sub-step, we distribute the increment to the FP of an ancestor event over its affected descendent events in the presence of an AND gate. The increment calculation is dependent on five cases, as follows:
Case 1. Two descendent events and only one of them is affected. This is the simplest case because all what we need to do is to divide the increased FP of the ancestor event on the current FP of the unaffected descendent event as presented in Equation 8:

$$FP_{Increased(Desc(x))} = \frac{FP_{Increased(Ancestor)}}{FP_{Current(Unaffect|Desc(x))}} \quad (8)$$

Case 2. All descendent events are affected and share the same FP value. In this case, we apply: $\sqrt[n]{FP_{(Increased|Ancestor)}}$, where n is the number of the descendent events.
Case 3. All descendent events are affected and do **NOT** share the same FP value. In this case, we apply equations (3) and (4) as described in Section IV.
Case 4. NOT all descendent events are affected where the affected ones share the same FP value. In this case, we apply Equation 9 as follows:

$$FP_{Increased(Desc(x))} = \sqrt[n]{(\frac{FP_{Increased(Ancestor(x))}}{\sum FP_{Current|Unaffect(x_1)*(x_2)*...*(x_n))}} \quad (9)$$

where n is the number of the affected events.
Case 5. NOT all descendent events are affected where the affected ones do **NOT** share the same FP value. In this case, we use Equation 10, as follows:

$$\frac{FP_{Increased(Ancestor(x))}}{\sum FP_{Current|Unaffected(x_1)*(x_2)*...*(x_n))}} \quad (10)$$

4) Repeat step 3 until the FP of all affected events get increased.

C. SANESAM+ For Predicted Changes: An Example

In this example we again use the WBS FTA. We consider a predicted change that will be applied to the power supplies within both $BSCU1$ and 2. However, it is still unknown how this change will increase the FPs of the two power supplies. We apply "SANESAM+ For Predicted Changes" to dedicate the maximum allowed FP to the affected events by the change, as follows:

1) $\Delta\ FP_{(BSFCLOBC)}$ = 3.30E-05 - 1.5031E-06
   $\Delta\ FP_{(BSFCLOBC)}$ = 3.14969E-05
2) Find the highest event that contains the effect. Changes to System 1 and 2 power supplies will directly affect the events BSS1EF and BSS2EF as highlighted in Figure 7 . These two events, however, are duplicated elsewhere in the FTA and thus there are multiple high events that contain the change.
   a) **BSS1EF on the left-hand side of the FTA** falls under LOOBS1 but the latter does not fall directly under the top event thus BSS1&2DNO is the highest event that contains the effect on BSS1EF.
   b) **BSS2EF on the left-hand side of the FTA** falls under LOOBS2 but the latter does not fall directly under the top event thus BSS1&2DNO is the highest event that contains the effect on BSS2EF.
   c) **BSS1EF on the right-hand side of the FTA** falls under LOOBS1 but the latter does not fall directly under the top event thus it is not the required highest event and we need to take one more level up to find the highest event. Having done that will lead us to SWFSTS1PAS1F which is also not the highest event that falls directly under the top event thus we need to go up again which will result SWFCTLOBBC as the required highest event.
   d) **BSS2EF on the right-hand side of the FTA** falls under LOOBS2 but the latter does not fall directly under the top event thus it is not the required highest event and we need to take one more level up to find the highest event. Having done that will lead us to SWFSTS1PAS2F which is also not the highest event that falls directly under the top event thus we need to go up again which will result SWFCTLOBBC as the required highest event.
3) Distribute the increment to the FP of an ancestor event over its descendent affected events. Since BSS1&2DNO and SWFCTLOBBC are the events that contain the change, no further calculations will be applied to BSVMIRFCSTA.

$$(\frac{4.71E\text{–}08}{1.5031E\text{–}06 - 8.00E\text{–}07} * 3.30E\text{–}05) + 4.71E\text{–}08$$

$$= 2.16E\text{–}06$$

$$(\frac{6.56E\text{–}07}{1.5031E\text{–}06 - 8.00E\text{–}07} * 3.30E\text{–}05) + 6.56E\text{–}07$$

$$= 3.00E\text{–}05$$

4) In this step, we repeat the previous step until all FPs of the affected events get increased. Figure 7 shows the calculated FPs (in squashed boxes).
5) In this step, we use 5 to calculate events' sensitivity.

It is worth noting that the sensitivity of BSS1PSF and BSS2PSF using SANESAM+ is 3.69185185 as shown in Table I. However, the sensitivity of these events is 18.55555556 when SANESAM+ For Predicted Changes is used.

V. Conclusions and Future Work

Sensitivity analysis is useful to measure the flexibility of different system properties to changes. In our previous work, we proposed a technique comprises of two phases to facilitate the maintenance of safety cases. SANESAM is the first phase of the technique in which we (1) measure the sensitivity of FTA events to system changes using the events' failure probabilities, and (2) derive safety contracts based on the results of the analysis. In the second phase, we map the derived safety contracts to a safety argument to improve the change impact analysis on the safety argument. In this paper, we identified some limitations to SANESAM and we suggested two options as extensions to resolve these limitations. The first option is SANESAM+, which is useful in the case of arbitrary changes because it calculates the FP for all events in the FTA regardless of any change scenario. The second option is SANESAM+ For Predicted Changes, this option increases the FP for only the events that are associated to a predicted change. A derived safety contract by SANESAM+ For Predicted Changes can guarantee higher FP than the guaranteed FP (for the same event and using the same set of assumptions) in a derived safety contract by SANESAM+. Hence, the derived safety contracts by SANESAM+ For Predicted Changes are more tolerant and robust than those derived by SANESAM+. Future work will focus on describing the second phase of the technique. Creating a case study to validate both the feasibility and efficacy of the technique is also part of our future work.

Acknowledgment

We acknowledge the Swedish Foundation for Strategic Research (SSF) SYNOPSIS Project for supporting this work. We thank Patrick Graydon for his help and fruitful discussions of this paper.

References

[1] C. K, "CIMAH safety cases - the HSE approach," IChemE Symposium series no. 110, 1988.

[2] U.K. Ministry of Defence, "JSP 430 - Ship Safety Management System Handbook," Ministry of Defence January 1996.

[3] T. P. Kelly, "Arguing safety – a systematic approach to managing safety cases," 1998.

[4] HSE, "Railway Safety Cases - Railway (Safety Case) Regulations 1994 - Guidance on Regulations," Health and Safety Executive, HSE Books 1994.

[5] T. Kelly and J. McDermid, "A systematic approach to safety case maintenance," in Proceedings of the Computer Safety, Reliability and Security, ser. Lecture Notes in Computer Science, 1999, vol. 1698, pp. 13–26.

[6] U.K. Ministry of Defence, 00-56 Safety Management Requirements for Defence Systems, Ministry of Defence, Defence Standard December 1996.

[7] S. Bates, I. Bate, R. Hawkins, T. Kelly, J. McDermid, and R. Fletcher, "Safety case architectures to complement a contract-based approach to designing safe systems," in Proceedings of the 21st International System Safety Conference (ISSC), 2003.

[8] O. Jaradat, I. Bate, and S. Punnekkat, "Using sensitivity analysis to facilitate the maintenance of safety cases," in Proceedings of the 20th International Conference on Reliable Software Technologies, June 2015. [Online]. Available: http://www.es.mdh.se/publications/3860-

[9] A. Saltelli, Global sensitivity analysis: the primer. John Wiley, 2008. [Online]. Available: http://books.google.at/books?id=wAssmt2vumgC

[10] O. Jaradat, I. Bate, and S. Punnekkat, "Facilitating the maintenance of safety cases," in Proceedings of the 3rd International Conference on Reliability, Safety and Hazard - Advances in Reliability, Maintenance and Safety (ICRESH-ARMS), June 2015. [Online]. Available: http://www.es.mdh.se/publications/3885-

[11] A. Cullen and H. Frey, Probabilistic techniques in Exposure assessment. New York: Plenum Press, 1999.

[12] M. C. Lucia Breierova, "An introduction to sensitivity analysis," Massachusetts Institute of Technology, Tech. Rep., September 1996.

[13] D. J. Pannell, "Sensitivity analysis of normative economic models: theoretical framework and practical strategies," Agricultural Economics, vol. 16, no. 2, 1997, pp. 139 – 152. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169515096012170

[14] P. Emberson and I. Bate, "Stressing search with scenarios for flexible solutions to real-time task allocation problems," Software Engineering, IEEE Transactions on, vol. 36, no. 5, Sept 2010, pp. 704–718.

[15] B. Meyer, "Design by contract," Interactive Software Engineering Inc., Tech. Rep. TR-EI-12/CO, 1986.

[16] ——, Object-Oriented Software Construction, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[17] L. Benvenuti, A. Ferrari, E. Mazzi, and A. L. Vincentelli, "Contract-based design for computation and verification of a closed-loop hybrid system," in Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 58–71. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78929-1_5

[18] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis, "Multiple viewpoint contract-based specification and design," in Proceedings of the 6th International Symposium, October 2007, pp. 200–225.

[19] W. Damm, H. Hungar, B. Josko, T. Peikenkamp, and I. Stierand, "Using contract-based component specifications for virtual integration testing and architecture design," in Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011, pp. 1–6.

[20] S. S. Bauer, A. David, R. Hennicker, K. Guldstrand Larsen, A. Legay, U. Nyman, and A. Wasowski, "Moving from specifications to contracts in component-based design," in Proceedings of the 15th International Conference on Fundamental Approaches to Software Engineering, ser. FASE'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 43–58.

[21] J. L. Fenn, R. D. Hawkins, P. Williams, T. P. Kelly, M. G. Banner, and Y. Oakshott, "The who, where, how, why and when of modular and incremental certification," in Proceedings of the 2nd IET International Conference on System Safety. IET, 2007, pp. 135–140.

[22] "Safecer," (2013, June) Safety certification of software-intensive systems with reusable components. [Online]. Available: http://www.safecer.eu.

[23] P. Graydon and I. Bate, "The nature and content of safety contracts: Challenges and suggestions for a way forward," in Proceedings of the 20th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC), November 2014.

[24] GSN Community Standard: (http://www.goalstructuringnotation.info/) Version 1; (c) 2011 Origin Consulting (York) Limited.

[25] S. P. Wilson, T. P. Kelly, and J. A. McDermid, "Safety case development: Current practice, future prospects," in Proceedings of the 12th Annual CSR Workshop - Software Based Systems. Springer-Verlag, 1997.

[26] Modular Software Safety Case (MSSC) — Process Description. [online]. available: https://www.amsderisc.com/related-programmes, Industrial Avionics Working Group (IAWG), Nov 2012.

[27] "SAE ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment," Warrendale, USA, Dec. 1996.

[28] O. Lisagor, M. Pretzer, C. Seguin, D. J. Pumfrey, F. Iwu, and T. Peikenkamp, "Towards safety analysis of highly integrated technologically heterogeneous systems – a domain-based approach for modelling system failure logic," in Proceedings of the 24th International System Safety Conference (ISSC), Albuquerque, USA, 2006.

[29] M. Rausand and A. Høyland, System Reliability Theory: Models, Statistical Methods and Applications. Hoboken, NJ: Wiley-Interscience, 2004.