

Using Safety Contracts to Guide the Maintenance of Systems and Safety Cases: An Example

Omar Jaradat*

*School of Innovation, Design, and Engineering
Mälardalen University
Västerås, Sweden
Email: omar.jaradat@mdh.se
Telephone: +46 (21) 101369, Fax: +46 (21) 101460

Iain Bate*†

†Department of Computer Science
University of York
York, United Kingdom
Email: iain.bate@york.ac.uk
Telephone: +44 (1904) 325572, Fax: +44 (1904) 325599

Abstract—Changes to safety critical systems are inevitable and can impact the safety confidence about a system as their effects can refute articulated claims about safety or challenge the supporting evidence on which this confidence relies. In order to maintain the safety confidence due to changes, system developers need to re-analyse and re-verify the system to generate new valid items of evidence. Moreover, identifying the effects of a particular change is a crucial step in any change management process as it enables system developers to estimate the required maintenance effort and reduce the cost by avoiding wider analyses and verification than strictly necessary. This paper presents a sensitivity analysis-based technique which aims at measuring the ability of a system to contain a change (i.e., robustness) without the need to make a major re-design. The technique exploits the safety margins in the assigned failure probabilities to the events of a probabilistic fault-tree analysis to compensate some potential deficits in the overall failure probability budget due to changes. The technique also utilises safety contracts to provide prescriptive data for what is needed to be revisited and verified to maintain system safety when changes happen. We demonstrate the technique on a realistic safety critical system.

Keywords—*sensitivity analysis, safety case, change impact, failure probabilities, maintenance.*

I. INTRODUCTION

System safety is a major property that should be adequately assured during the development process, the deployment and the operation life of safety critical systems. System safety is not assured by chance but rather it must be engineered and evaluated in a systematic manners that might be mandated by safety standards, best practices, experts' recommendations. Hence, safety critical systems are often subject to a compulsory or advisory certification process which often necessitates building the systems in compliance with domain-specific safety standards. For example, some automotive systems (e.g., passenger cars) might be built in compliance with ISO 26262, railway systems in compliance with EN 50126, certain airborne systems in compliance with DO-178B, etc. Based on the level of risks posed by a system, the safety standards help to define the required safety assurance level (e.g., Safety Integrity Levels (SILs), Design Assurance Level (DAL), etc.) and prescribe the tools, techniques and methods that should be adopted by the development and assessment lifecycle [13].

Following the standards prescriptions leads system developers to generate a lot of artefacts during and after the

development of their systems. These artefacts are used as safety evidence to prove that the standards obligations and recommendations were carried out. However, if the generated artefacts are not demonstrated and explained properly, there will be less certainty about their importance which may lead the overall confidence being undermined. Therefore, developers of some safety critical systems construct a *safety case* (also known as "*assurance case*") to demonstrate the safety aspect of a system by identifying all potential risks and describing, in the light of the available evidence, how these risks have been eliminated or duly mitigated to ALARP (As Low As Reasonably Practicable). More clearly, a safety case comprises both safety evidence (e.g. safety analyses, software inspections, or functional tests) and a *safety argument* explaining that evidence [25]. The safety argument shows which claims the developer uses each item of evidence to support and how those claims, in turn, support broader claims about system behaviour, hazards addressed, and, ultimately, acceptable safety. That is, safety cases provide evidential information about the safety aspect of a system by which a regulatory body can reasonably conclude that the system is acceptably safe and thus grants the certification stamp. Safety cases have received industrial attention and they have become common practice as they are mandatory in some domains, and even when they are not mandatory they are considered as a good practice. However, safety cases are costly since they need a significant amount of time and effort for them to be produced. In fact, the cost of obtaining certification is significant, with estimates such as 30% of lifecycle costs [9] and 25-75% of development costs [35] are spent on certification [5].

Typically, safety critical systems are evolutionary and they are always exposed to both predicted and unpredicted changes during the different stages in their lifecycle. System changes are due to the need of improving the systems performance (i.e., perfective changes), correct discovered faults and errors (i.e., corrective changes), or perhaps incorporate new system modes and conditions (i.e., adaptive changes), etc. Changes to a system can negatively affect the gained confidence because these changes have the potential to compromise the safety evidence which has been already collected. More clearly, evidence after a change might no longer support the developers' claims because it reflects old development artefacts or old assumptions about operation or the operating environment. In the updated system, existing safety claims might be nonsense, no longer

reflect operational intent, or be contradicted by new data [25].

In order to maintain the confidence in the safety performance, developers must update the safety case which, in turn requires identifying, re-analysing, and re-checking the impacted parts of the system and generate a new valid set of evidence. This is more complicated task than it might first appear. Change requests should be assessed before decision makers decide whether or not to accept them. The assessment should reveal if the change can cause unreasonable risks, and the required cost to implement the change. Hence, system developers should understand the change and the potential risks that it might carry before they identify the impacted parts. For example, a change might turn some implicit assumptions about the context in which a system should operate to be wrong. Misunderstanding the change might lead to skip those parts of the system which are dependent on that assumptions. Also, the developers need to understand the dependencies between the system parts to identify the affected parts correctly. For example, the effect of a change can propagate to other parts of the system — creating a ripple effect — and cause unforeseen violations of the acceptable safety limits. If the impact of change is not clear, developers might be conservative and do wider analyses and verification (i.e., check more elements than strictly necessary), and this will exacerbate the cost problem of safety cases. It is also necessary for the developers to describe how the change affects the system parts — that are listed as affected — in order to correctly estimate the cost of the response to that change. Otherwise, the response to a change might generate unplanned further changes to which the system must again respond [37], and this requires more cost than originally calculated.

Despite clear recommendations to adequately maintain and review the systems and their safety cases by safety standards, existing standards offer little or no advice on how such operations can be carried out [36]. Hence, there is an increasing need for globally acceptable methods and techniques to enable easier change accommodation in safety critical systems without incurring disproportionate cost compared to the size of the change. However, since broader re-verification and re-validation require more effort and time, it is important for any proposal aims at facilitating system changes to localise the impact of the changes. More specifically, to alleviate the cost of updating both a system and its safety case due to a change, it is crucial to minimise the effects of that change and prevent these effects from propagating into other parts of the system as far as it is practically possible. In other words, systems need to be more resilient to changes. In order to enable such resilience, system designers need descriptions of predicted changes, during the design phase of a system, to construct the system design in a way that they decouple (or minimise the coupling) the affected part from the other parts to prevent the propagation. The problem though is that system changes and their details cannot be fully predicted and made available up front [17].

In our previous work [24], we introduced a Sensitivity ANalysis for Enabling Safety Argument Maintenance (SANESAM) technique that supports system engineers to anticipate potential changes. We also developed SANESAM+ [17] as another version of SANESAM that covers wider variety of change scenarios. The key principle of SANESAM and

SANESAM+ is to determine the flexibility (or robustness) 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 events that will be, most likely, related to the future changes. We use safety contracts to record the information of the maximum allowed changes to those events before violating the minimum acceptable safety limits. Those contracts can be used as part of later change impact analysis to advise the engineers what to consider and check when changes actually happen. The focus in this paper is not to show how SANESAM(+) can help anticipating potential changes; rather the main contribution of this paper is to propose a new approach through which SANESAM(+) is used to contain (i.e., localise) the potential changes in the smaller possible part of a system. More clearly, we compare the calculated MAFP (Maximum Allowed Failure Probability) of the events with new estimated FP of those events due to a change. If a new estimate FP of an event is \leq MAFP, then the change will not, necessarily, require a considerable system modification, otherwise, it means that there will be a deficit in that FP and more effort should be considered. There could be several ways to respond to the latter case, but some responses might require large planning and massive re-engineering effort. Alternatively, we suggest, in this paper, to use the FP margins(s) of other events to compensate the resultant deficit. The paper uses the aircraft Wheel Braking System (WBS) to illustrate different examples of changes containment.

This paper is composed of five further sections. In Section II, we present necessary background information to make the paper self-contained. In Section III, we describe two techniques to facilitate the maintenance of safety cases. We use this description as a basis to introduce a new technique to facilitate the maintenance of safety critical systems and safety cases in Section IV. In Section V, we use the WBS system as an illustrative example. Finally, we conclude and propose potential future works in Section VI.

II. BACKGROUND AND MOTIVATION

A. Fault Tree Analysis (FTA)

In 1962, Bell Telephone Laboratories introduced the fault tree technique as a means to evaluate safety in the launching system of the intercontinental *Minuteman* missile [30]. The Boeing Company improved the technique and introduced computer programs for both qualitative and quantitative fault tree analysis. Today FTA is the most commonly used technique for safety and reliability studies.

FTA is a failure analysis method which focuses on one particular undesired event and provides a method for determining causes of this event [2]. In other words, FTA uses abductive reasoning to identify different causes to critical states (from a safety or reliability standpoint). These states might be associated with component hardware failures, human errors, software errors, or any other pertinent events. FTA helps safety engineers to identify plausible causes (i.e., faults) of undesired events [34]. A fault tree illustrates the logical interrelationships of the system's components (*Basic Events*) that lead to the undesired event or the system's state (*Top Event*) [34], [30]. The logical interrelationships are called *Logical Gates*.

Moreover, FTA is used as a method to achieve Probabilistic Safety Analysis (PSA). More specifically, probability of failure is assigned to each of the failure events based on historical data, and the failure probability of the top event is determined [33]. Quantitative FT evaluation techniques produce three types of results: (1) numerical probabilities, (2) quantitative importance, and (3) sensitivity evaluations [2]. Both SANESAM and SANESAM+ utilise the failure probabilities of PSA(s) to measure how sensitive a system design is to a particular aspect of individual event.

B. 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*” [32]. The analysis helps to establish reasonably acceptable confidence in the model by studying the uncertainties that are often associated with variables in models. Many variables in system analysis or design models represent quantities that are very difficult, or even impossible to measure to a great deal of accuracy [29]. In practice, system developers are usually uncertain about variables in the different system models and they estimate those variables. Sensitivity analysis allows system developers to determine what level of accuracy is necessary for a parameter (variable) to make the model sufficiently useful and valid [8].

There are different purposes for using sensitivity analysis. The analysis can be used to provide insight into the robustness of model results when making decisions [10]. Also, the analysis can be used to enhancing communication from modelers to decision makers, for example, by making recommendations more credible, understandable, compelling or persuasive [28]. In safety domains, sensitivity analysis can be used in risk analysis models to determine the most significant exposure or risk factors so to speak, and thus, it can support the prioritisation of the risk mitigation. Sensitivity analysis methods can be classified in different ways such as mathematical, graphical, statistical, etc. In this paper we use the sensitivity analysis to identify the safety argument parts (i.e., sensitive parts) that might require unneeded painstaking work to update with respect to the benefit of a given change. The results of the analysis should be presented in the safety argument so that it is always available up front to get developers’ attention.

SANESAM and SANEMSAM+ exploit 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 [24], [17]. In Section III, we provide more descriptions about SANESAM and SANESAM+.

C. Safety Contracts

The term ‘contract’ is defined in English as: “*A written or spoken agreement, especially one concerning employment, sales, or tenancy, that is intended to be enforceable by law*” [3]. A contract is intended to (1) establish a binding relationship between one party’s offer and the acceptance of that offer by one or more parties, and (2) set out the terms and conditions that constrain this relationship. Using the contracts is familiar in software development. For instance, Design by Contract (DbC) was introduced by Meyer [22], [23] to constrain the interactions that occur between objects. Moreover, 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 behaviour [7].

In 1969, Hoare introduced the pre- and postcondition technique to describe the connection (dependency) between the execution results (R) of a program (Q) and the values taken by the variables (P) before that program is initiated [14]. Hoare introduced a new notation to describe this connection, as follows:

$$P \{Q\} R$$

This notation can be interpreted as: “*If the assertion P is true before initiation of a program Q , then the assertion R will be true on its completion*” [14].

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 post-conditions 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 [6], [39], [31]. 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 [15], [27], [12].

The following is an example that depicts the most common used form of contracts [17]:

Guarantee: The WCET of task X is ≤ 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

In this paper, we use safety contracts to record the dependencies among failure probabilities of FTA’s events.

D. Safety Case

There are different definitions of safety case [18], [21]. Most of the available definitions indicate the consensus that a safety case is oriented to demonstrate how a system reduces risk of specific losses to an acceptable level and thus enable a regulator to assess whether the system is acceptably safe to operate. It is worth pointing out that the definition of safety case by the UK Defence Standard 00-56 [38] is the most common. The standard defines the safety case as: “A structured argument, supported by evidence, intended to justify that a system is acceptably safe for a specific application in a specific operating environment”. Hence, a safety case comprises both safety evidence (e.g. safety analyses, software inspections, or functional tests) and a safety argument explaining that evidence [25]. The development of a safety case as a means of demonstrating acceptable risk began in the nuclear industry but the application of this means was uncommon in other industries. From 1990s onwards the development of safety cases spread across many other major hazard industries, such as the automotive, avionics, railways, offshore oil, gas facilities, etc. [40]. It is worth mentioning that the terms ‘safety case’ and ‘assurance case’ are, sometimes, used interchangeably.

E. Safety Argument

The main purpose of a safety case is to communicate an argument. The argument demonstrates how someone can reasonably conclude that a system is acceptably safe from the evidence available [19]. An argument in the safety case definition is called a ‘safety argument’ or ‘safety case argument’ and it can be defined as a hierarchically connected series of claims supported by evidence. Safety arguments are intended to demonstrate to the reader that a system is acceptably safe as an overall claim. The claim is defined as: *A proposition being asserted by the author or utterer that is a true or false statement* [26]. The evidence is defined as: *Information or objective artifacts being offered in support of one or more claims* [26].

In order for safety cases to be developed, discussed, challenged, presented and reviewed amongst stakeholders, as well as maintained throughout the product lifecycle, it is necessary for the (1) argument to be clearly structured and (2) items of evidence to be clearly asserted to support the argument [1]. Figure 2 shows an overview of the safety case elements and the relationships between them. There are several ways to represent safety arguments (e.g., textual, tabular, graphical, etc.). In this paper, however, a graphical representation is used. More specifically, we use the Goal Structuring Notation (GSN) [4], which provides a graphical means of communicating

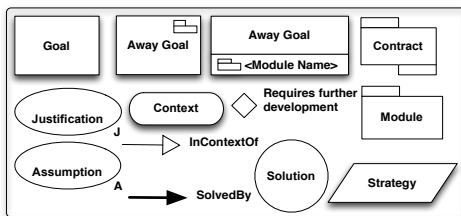


Fig. 1. Notation Keys of the GSN

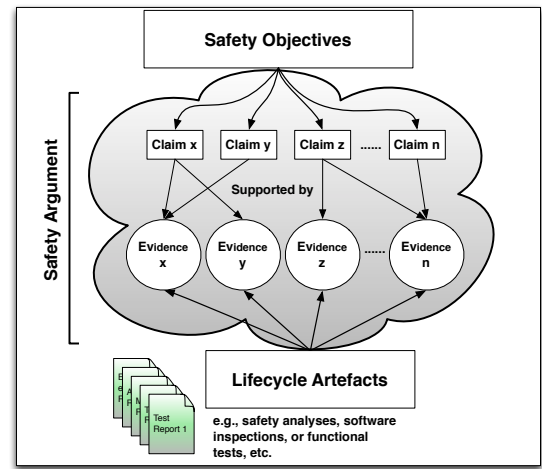


Fig. 2. Overview of a safety case and its elements [16]

(1) safety argument elements, claims (goals), argument logic (strategies), assumptions, context, evidence (solutions), and (2) the relationships between these elements. The principal symbols of the notation are shown in Figure 1 (with example instances of each concept).

A goal structure shows how goals are successively broken down into ('solved by') sub-goals until eventually supported by direct reference to evidence. Using the GSN can clarify the argument strategies adopted (i.e., how the premises imply the conclusion), the rationale for the approach (assumptions, justifications) and the context in which goals are stated.

F. Safety Cases and Maintenance

Safety assurance and certification are amongst the most expensive and time-consuming tasks in the development of safety-critical embedded systems [11]. A key reason behind for this is the increasing complexity and size of these systems combined with their growing market demands. One of the biggest challenges that affects safety case revision and maintenance is that a safety case documents a complex reality that comprises a complex web of interdependent elements. That is, safety goals, evidence, argument, and assumptions about operating context are highly interdependent. Hence, seemingly minor changes may have a major impact on the contents and structure of the safety argument. Basically, operational or environmental changes may invalidate a safety case argument for two main reasons as follows:

- 1) Evidence is valid only in the operational and environmental context in which it is obtained, or to which it applies. During or after a system change, evidence might no longer support the developers' claims because it could reflect old development artefacts or old assumptions about operation or the operating environment.
- 2) Safety claims, after introducing a change, might be nonsense, no longer reflect operational intent, or be contradicted by new data. Changing safety claims might change the argument structure.

Hence, maintaining safety cases after implementing a system

change is a painstaking process, and this because of:

1) the lack of documentation of dependencies among the safety cases contents, 2) the lack of traceability between a system and its safety case, and 3) system changes and their details cannot be fully predicted and made available up front to contain them by design.

In this paper, we refer to “Maintainability” or “Maintenance” as the ability to repair or replace the impacted elements of a safety case argument, without having to replace still valid elements, to preserve the validity of the argument. This includes:

- 1) Identifying the impacted elements and those that are not impacted.
- 2) Minimising the number of impacted elements.
- 3) Reducing the work needed to make the impacted elements valid again.

III. SANESAM AND SANESAM+

In this section, we give an overview of SANESAM [24] and SANESAM+ [17]. The key principle of both techniques 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 the techniques as a method to determine the range of failure probability parameter for each component. The techniques assume 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, they assume the existence of the required failure probability for the top event $FP_{Required}(Topevent)$, where the FTA is considered unreliable if:

$$FP_{Current}(Topevent) > FP_{Required}(Topevent) \quad [24].$$

A. SANESAM Process

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

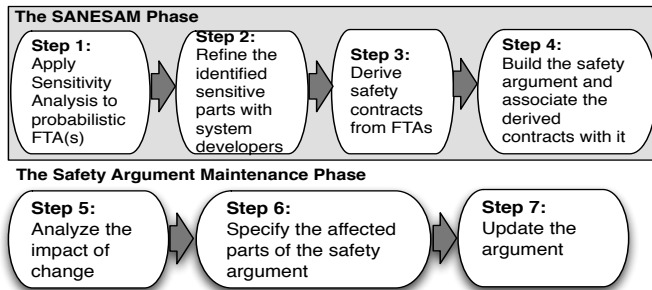


Fig. 3. Process diagram of SANESAM [24]

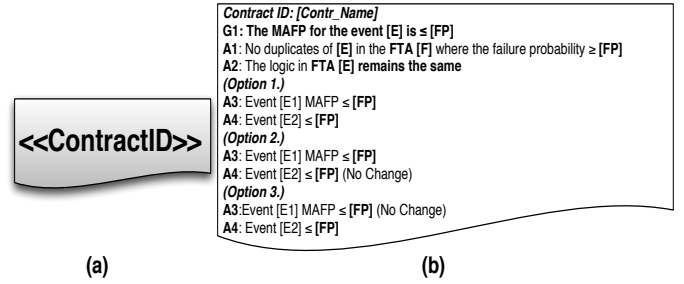


Fig. 4. (a) FTA Safety contract notation, (b) Derived safety contract

if the set cannot be reduced without losing its status as a cut set” [30].

- 2) Calculate the maximum possible increment to the failure probability parameter of event x before the top event $FP_{Required}(Topevent)$ is no longer met, where $x \in MC$ and $(FP_{Increased|event(x)} - FP_{Current|event(x)}) \neq FP_{Increased}(Topevent) > FP_{Required}(Topevent)$.
- 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 to 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, as shown in Figure 4-a. We also create a template to document the derived safety contracts. Figure 4-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.

Essentially, SANESAM calculates the maximum possible increment to the failure probability parameter of only one event at a time before the top event $FP_{Required(Topevent)}$ is no longer met. In addition, it considers the events within the *MC* only.

B. SANESAM+ Process

In [17], we suggested to extend SANESAM by SANESAM+ to resolve some identified limitations. Briefly, SANESAM+ was introduced to provide more freedom by considering multiple events at a time and not only the events in the *MC*. SANESAM+ gives the systems' developers two options: 1) SANESAM+ without considering any predicted changes, 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, 2) 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+.

The main difference between the SANESAM process and SANESAM+ process is observed while applying the sensitivity analysis (i.e., Step 1). In addition, SANESAM+ does not require any refinements of the sensitive events. Hence, Step 2 in Subsection III-A shall be completely neglected for SANESAM+. All other steps are identical.

1) *SANESAM+ For Arbitrary Changes:* The detailed instructions of how to apply sensitivity analysis (Step 1) of SANESAM+ for arbitrary changes are as follows [17]:

- 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 descendant 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 descendant 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 descendant events.

LOOBS1 and LOOBS2 in Figure 7 represent an example of this case.

Case 2. if the descendant events do not share the same *FP*, then $FP_{(Topevent)}$ is distributed over them unevenly, but rather based on the *FP* ratio of every descendant event to $\Delta FP_{(Ancestor)}$ as described by Equation 3:

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

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

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

LOOBS1 and SWFSIS1P in Figure 7 represent an example of this case.

- 3) Repeat steps 1 and 2 until *FP* of the basic events get increased. Unlike SANESAM, SANESAM+ process distinguishes 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:

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

2) *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 $\Delta FP_{(Top\ event)}$ over all FTA's events might increase the change impact tolerance of some events that are 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:

- 1) Find the difference between the current and required FP of the top event $\Delta FP_{(Top\ event)}$.
- 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 $\Delta FP_{(Top\ event)}$ 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 run should 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 descendant 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 descendant 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:

$$\begin{aligned} & FP_{Increased(Ancessor)} \\ & - FP_{Current(Unaffected|Desc(x))} \\ & = FP_{Increased(Desc(x))} \end{aligned} \quad (6)$$

Otherwise, the ratio of a descendant event to its ances-

tor and the granted increment to an affected event is calculated by Equation 7 as follows:

$$\begin{aligned} & FP_{Increased(Desc(x))} \\ & = \left(\frac{FP_{Current(Desc(x))}}{FP_{Current(Ancessor)} - \sum FP_{Current(Unaffected)}} \right) \\ & \quad * FP_{Increased(Ancessor)} + FP_{Current(x)} \end{aligned} \quad (7)$$

- b) In this sub-step, we distribute the increment to the FP of an ancestor event over its affected descendant events in the presence of an AND gate. The increment calculation is dependent on five cases, as follows:

Case 1. Two descendant 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 descendant event as presented in Equation 8:

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

Case 2. All descendant events are affected and share the same FP value. In this case, we apply:

$\sqrt[n]{FP_{Increased(Ancessor)}}$, where n is the number of the descendant events.

Case 3. All descendant events are affected and do NOT share the same FP value. In this case, we apply equations (3) and (4) as described in Section III-B1.

Case 4. NOT all descendant events are affected where the affected ones share the same FP value. In this case, we apply Equation 9 as follows:

$$\begin{aligned} & FP_{Increased(Desc(x))} \\ & = \sqrt[n]{\left(\frac{FP_{Increased(Ancessor(x))}}{\sum FP_{Current(Unaffected(x_1)*(x_2)*...*(x_n))}} \right)} \end{aligned} \quad (9)$$

where n is the number of the affected events.

Case 5. NOT all descendant events are affected where the affected ones do NOT share the same FP value. In this case, we use Equation 10, as follows:

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

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

IV. SAFETY CONTRACTS DRIVEN MAINTENANCE

The way we suggest to cope with some types of changes is to contain their effects in the smallest possible set of events to prevent (or minimise) the ripple of these effects from propagation. In this section, we describe a new technique that enables the containment of certain class of changes in safety critical systems and safety cases. It is worth noting that this technique utilises the same rules by which SANESAM and SANESAM+ calculate the sensitivities and associate them with a safety argument via safety contracts. However, the technique adds additional steps to enable effective usage of the safety margins in a probabilistic FTA. The new technique provides solutions to accommodate a change even if the change broke one or more safety contracts. The only needed input for the

process of the technique is a probabilistic FTA. The process comprises 6 steps that can fall into two main phases, before and after introducing a change. The three steps before performing a change are similar to the first phase of SANESAM and SANSAM+ as shown in Figure 3. However, we have made some non-substantial changes to some of these steps, where we describe the change to each step when we describe the step itself. Steps 4-6 are novel and they were designed and specified for the new technique.

Steps before performing a change:

- 1) **Apply the sensitivity analysis to a probabilistic FTA:** This step is performed exactly as instructed in the process of either SANESAM or SANESAM+.
- 2) **Derive safety contracts:** In this step, we need to derive safety contracts from FTAs as described in Section III. However, there are two main differences in the derivation of safety contracts in this work. First, the guaranteed MAFPs in the safety contracts are basically the results of either multiplication or summation of multiple children events. Hence, there is no point to derive contracts for basic events in FTA because they simply do not have children events. The second main difference is that the contracts should provide multiple options for developers to measure the tolerance of a change's impact. More clearly, each derived safety contract should assume that only one child event is affected, multiple children events are affected or all of them are affected.
- 3) **Associate the derived contracts with safety arguments:** Unlike the same step in SANESAM (in Section III) and to enable more freedom, the proposed technique in this work considers that the construction of safety arguments is not necessarily a part of the process. Hence, we assume the existence of a safety argument no matter how it is represented (e.g., textual, tabular, graphical, etc.). The most important for us is the association itself because this association highlights the suspect elements in the argument to bring them to developers' attention.

Steps after performing a change:




- 4) **Check the ability of FTA to contain greater FP(s) than those already exist:** The key principle of this step is to compare the new estimated FP of an affected event with the guaranteed MAFP in the safety contract of that event, or probably in the safety contracts of higher events. As a quick check, we can determine the *MC* and calculate the expected FP of the top event taking into consideration the new increased FPs of the affected events and the

current FPs (not the MAFP) of the unaffected ones. If the new calculated FP of the top event is \geq MAFP, then the change is not containable and this means that there will be a deficit in the overall FP budget where the response to the change might require re-engineering effort. Dealing with such a situation is beyond the scope of this paper. In contrast, if the new calculated FP of the top event is \leq MAFP, this means that the change's impact is containable somewhere in the FTA, but we need to know which safety contract contains it. To do that, the new estimated FP of an impacted event should be checked against the guaranteed MAFP in the safety contract of that event, where it is containable iff it is \leq MAFP. If the change's effect (i.e., difference between the new estimated FP and the MAFP) is not containable in the safety contract of the ancestor event, then the safety contract of the ancestor event should be investigated as whether or not it can contain it. If the change's effect still cannot be contained by the ancestor, safety contracts in one more level up should be investigated and so on and so forth until a safety contract contains it. Once the contract which contains the change's effect is identified, all associated claims with this contract together with their supporting arguments and evidence should be highlighted as suspect.

- 5) **Re-balance the FPs of the FTA's events as a preparation for future changes.** If any event has received a change that necessitates increasing its failure probability where the increment is still within the MAFP threshold in its safety contract, then it can be said that the safety contract in question still holds (intact) and the change is containable with no further significant maintenance. However, we need to re-balance the FPs of the FTA's events after accommodating a change to prepare for further accommodation(s) of future potential changes. Hence, we need to find $\Delta FP_{(T_{opevent})}$ which is the difference between the required FP and the new FP of the top event after containing a change. The current FPs of all unaffected events together with the new FPs of the affected events are used to calculate $FP_{New(T_{opevent})}$ based on the determined *MC* from Step 4. The resultant $FP_{New(T_{opevent})}$ is subtracted from $FP_{Required(T_{opevent})}$. The calculated $\Delta FP_{(T_{opevent})}$ might be equal to 0 or it can be an insignificant fraction which is not worth further effort. In this case, Step 1-ii should be omitted (i.e., no need for distribution) and we only need to update the safety contracts as described in the next step.
- 6) **Update the affected safety contracts with the new FPs:** The contracts should be updated by the latest failure probability value(s) after containing a change. This step can be seen as Step 2, the difference is that we do not derive new contracts, but we rather modify/update the ones we derived earlier. Figure 5 shows a flowchart diagram that summarises Steps 4, 5 and 6.

In order to enhance the visibility of a change's impact on the system design and safety case, we highlight the parts of the system design and the elements of the safety case that are related to the affected events in FTA. Three levels of impact based on the impact propagation within FTA were defined, namely, *Low*, *Medium* and *High*. Table I categories

TABLE I. RATING THE IMPACT OF CHANGE

Impact Level	Highlight Colour	Description	Impact on Safety case	
			Argument	Evidence
Low		Change to an event is contained within its safety contract	No change necessary	Might reuse the same evidence
Medium		Change to an event is not contained within its safety contract, however is contained by another higher level safety contract with sufficient margin	No change necessary	Might need new evidence
High		The change is not contained within any of the derived safety contracts and the overall failure target of the system cannot be met	Major impact on the safety argument structure	Need new evidence

the changes and suggest highlighting/representing them with different colours based on the rating of changes' impact.

V. ILLUSTRATIVE EXAMPLE

We apply our proposed technique described in Section IV to the Wheel Braking System (WBS) in which we assume three different change request scenarios and evaluate their impacts on the safety case.

A. Wheel Braking System (WBS): System Description

The WBS is described in Appendix L of Aerospace Recommended Practice ARP-4761 [2] for safety assessment processes. 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 grey background in Figure 6). 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 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 [20]. Figure 6 shows high-level view of the *BSCU* implementation. More details about the *BSCU* implementation can be found in ARP-4761 [2]. Figure 7 shows the “Loss

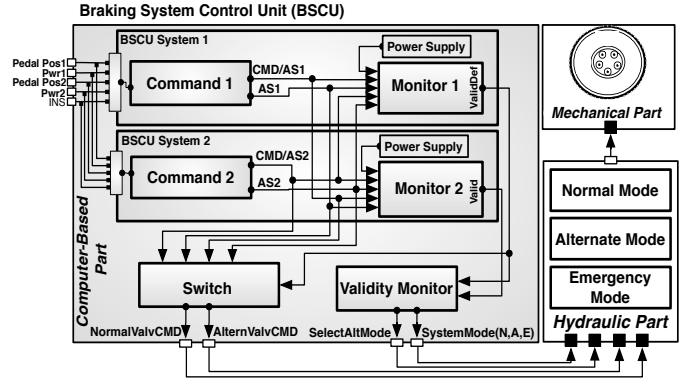


Fig. 6. A high-level view of the WBS [24]

of Braking Commands” probabilistic FTA (the original FTA is without the grey shapes) whilst Figure 8 shows a GSN fragment of the WBS safety argument.

B. Safety Contracts Driven Maintenance: An Example

Since we have now all the required inputs for our technique (i.e., probabilistic FTA in addition to top event MAFP and current FP), we can start applying the *Steps before performing a change* (Steps 1-3 in Section IV), as follows:

- 1) **Apply the sensitivity analysis:** In this example we apply SANESAM+ For Arbitrary Changes.

- i) Find $\Delta FP_{(Topevent)}$. The required and current probabilities of the top event should be clearly defined. Appendix L of the ARP-4761 states, as a safety requirement on the *BSCU*, that: “The probability of *BSCU* fault causes Loss of Braking Commands shall be less than $3.30E-05$ per flight”. This means that: $FP_{Required}(Topevent) < 3.30E-05$. On the other hand, the calculated $FP_{Current}(Topevent)$ by the example in the same appendix is $\approx 1.50E-06$. Hence, $\Delta FP_{(Topevent)} = 3.15E-05$.
- ii) Distribute $\Delta FP_{(Topevent)}$ over all events in FTA as $3.15E-05 > 0$. Figure 7 shows the current FPs as well as the MAFPs (in the grey boxes) for all of the events. The MAFP of an event is basically the current FP of that event plus its share from the distributed $\Delta FP_{(Topevent)}$.

- 2) **Derive safety contracts:** After calculating the MAFPs for all of the events in *WBS_FTA*, a safety contract was derived for each event non basic event. Each derived contract considers multiple assumptions options based on the number of the children events. For example, Figure 9 shows the derived safety contract for **LOOBS1** in which the summation of two FPs (i.e., the FPs for **BSS1EF** and **BSS1PSF**) is guaranteed. A change to any FP or two of them will result in a change to the guaranteed FP for the **LOOBS1**. Hence, the **Contr_LOOBS1** considers three options:

- a) a change to **BSS1EF** and **BSS1PSF**,
- b) a change to **BSS1EF** only, and
- c) a change to **BSS1PSF** only.

The template of the safety contract in Figure 4-a was used to represent the derived safety contracts. Also the contract

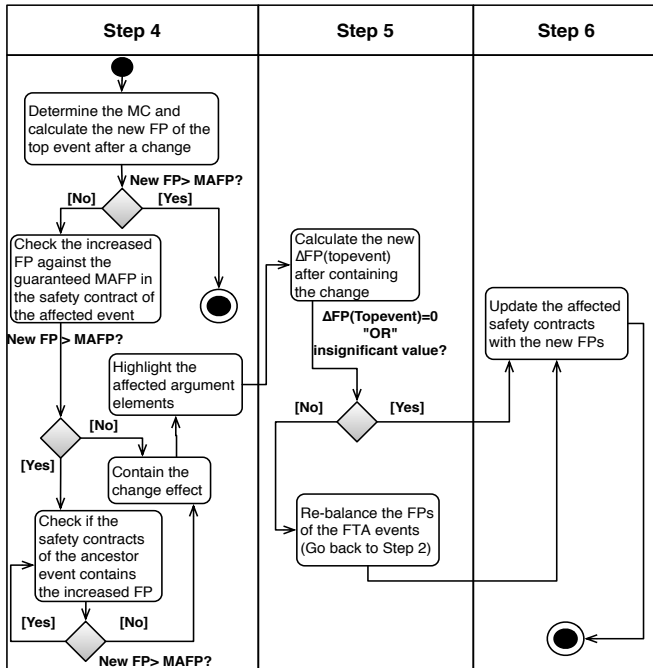


Fig. 5. Flowchart presentation of steps 4-6

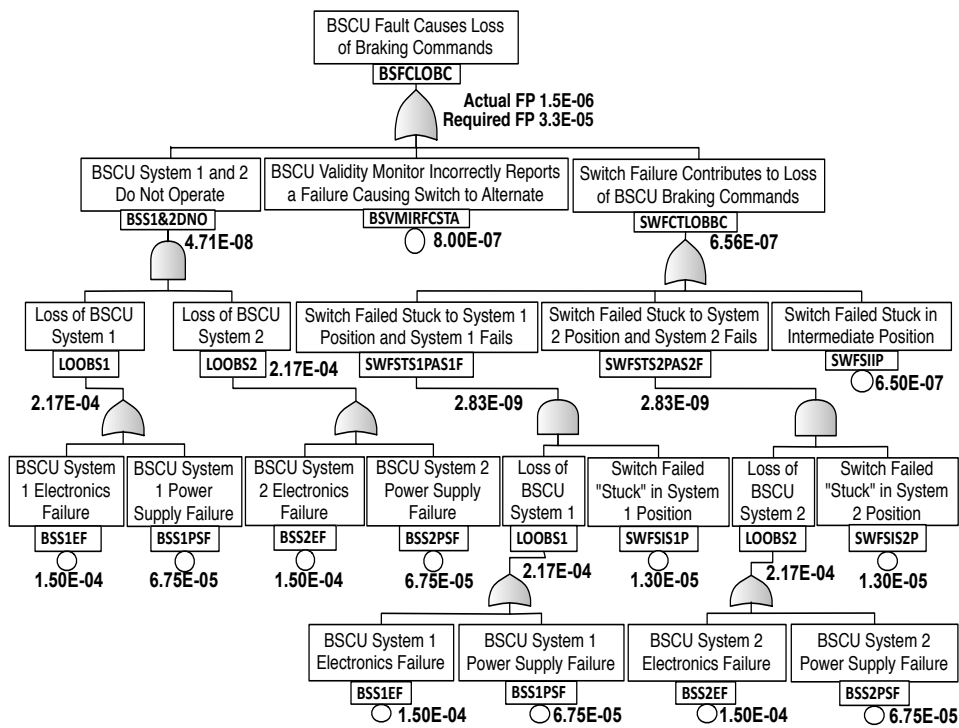


Fig. 7. Loss of Braking Commands FTA [2]

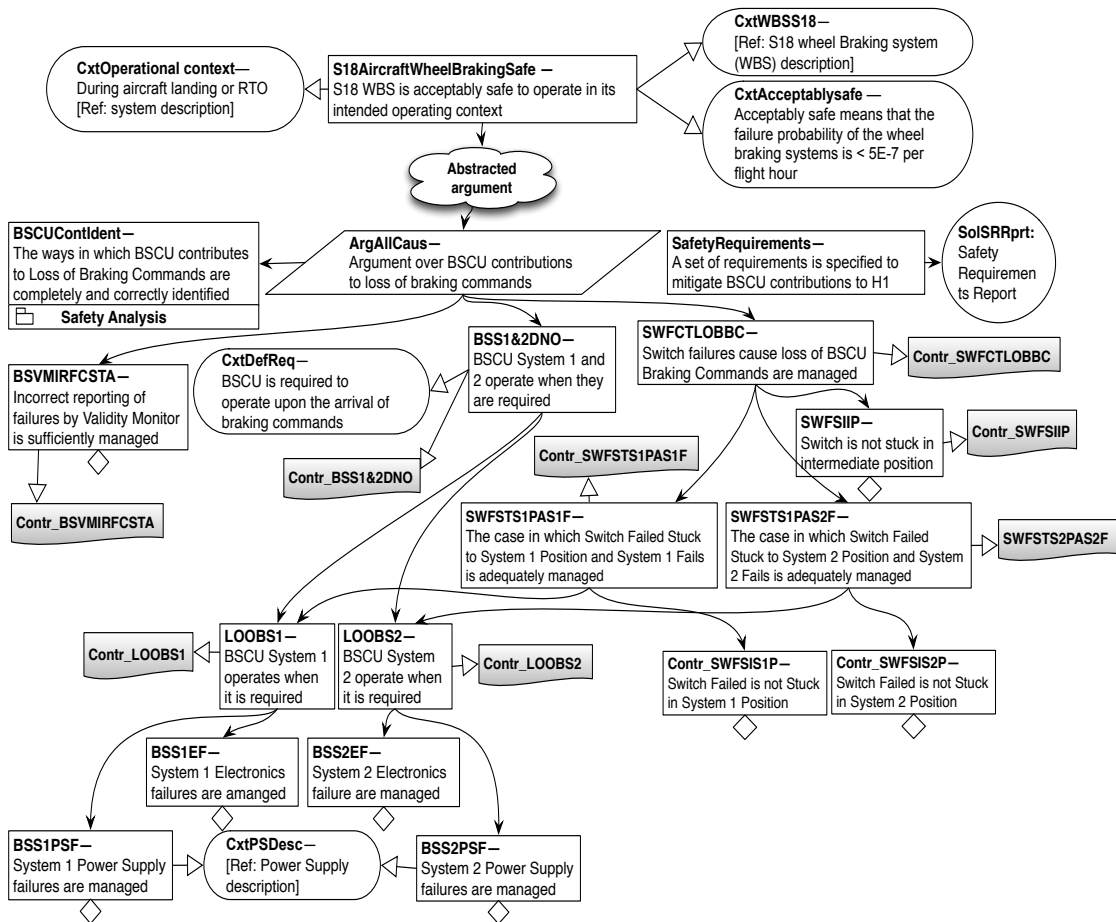


Fig. 8. Safety argument fragment for WBS

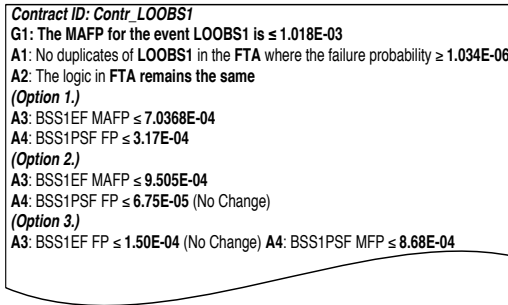


Fig. 9. A derived safety contract

notation in Figure 4-b was used to annotate the contracted events in FTA as shown in Figure 7.

- 3) **Associate the derived contracts with the safety argument:** In this step, we assume the existence of a safety argument no matter how it is represented (e.g., textual, tabular, graphical, etc.). In this example, we use a GSN argument fragment to show the association. Figure 8 shows how the derived safety contracts from FTA are associated with a safety argument fragment for WBS using the proposed contract notation in Figure 4-a. We do not want to affect the way GSN is being produced but we want to bring additional information for developers' attention. It is worth mentioning that a safety contract should be associated with all claims that are related to the event which the contract is derived for. For example, the safety contract *Contr_SWFSTS2PAS2F* should be associated with any articulated claims about the state when Switch Failed Stuck to System 2 Position and System 2 Fails.

Now, let's assume some change scenarios that can resemble real life change requests.

Change request scenario (1)

The WBS developers have received a change request from the senior management asking to replace the current installed power supplies in BSCU 1 and 2 by a different model. Based on the provided product specifications by the new power supplies manufacturer, the FP of that model is **3.00E-04**, which means that it is 344.4% greater (i.e., less reliable) than the FP of the current model in use (i.e., 6.75E-05). Subsequently, step 5 should be followed to assess the impact of the given change scenario.

- 4) **Check the ability of the FTA to contain greater FP(s) than those already exist:** In this step, we compare the FP of the new power supply model with the guaranteed MAFP of the current power supplies model. To do so we first need to identify the events that represent the power supplies in the FTA, and they are **BSS1PSF** and **BSS2PSF** as shown in Figure 7. It is worth noting, however, that events related the power supplies are duplicated in the FTA which means that we should be concerned about four events. These four events are basic events and since it is not possible to guarantee the MAFP of basic events based on FP of lower level events, thus there were no direct safety contracts derived for them. That is, the

MAFP of **BSS1PSF**, **BSS2PSF** and their duplicates are recorded as assumptions in the safety contracts that were derived for their parents event. This will lead us to four contracts, namely, **Contr_LOOBS1**, **Contr_LOOBS2**, **Contr_LOOBS1_D** and **Contr_LOOBS2_D**. As a quick check and before start comparing the new FP with the MAFPs in the safety contracts, we want to update the FPs of the affected events based on the new given FPs and calculate the new FP of the top event. This is because we want to check whether or not the new calculated FP of the top event exceeds the required FP by the safety requirements.

Based on the *MC* of the FTA, $FP_{Current(Topevent)} = 8.00E-07 + 6.50E-07 + (1.50E-04 * 1.50E-04) + (1.50E-04 * 3.00E-04) + (1.50E-04 * 1.30E-05) + (3.00E-04 * 1.50E-04) + (3.00E-04 * 3.00E-04) + (3.00E-04 * 1.30E-05) + (1.50E-04 * 1.30E-05) + (3.00E-04 * 1.30E-05) = 1.646E-06$.

Since **1.646E-06** < **3.3E-05**, the increments to the FPs of **BSS1PSF** and **BSS2PSF** is tolerated (i.e., containable) in the FTA but the question now is: **Where can they be contained?**

The answer to this question is obtained by revisiting the four identified safety contracts and check whether or not they still hold in the light of the new FP. As discussed in Step 3, each contract contains different options in the assumptions list (as shown in Figure 9). For the change scenario under discussion, we choose Opt.1 in the four contracts and check if the MAFPs of **BSS1PSF** or **BSS2PSF** can contain the new FP. Since 3.16E-04 (MAFP) > 3.00E-04 (new FP), the increments to **BSS1PSF** and **BSS2PSF** are contained in **Contr_LOOBS1**, **Contr_LOOBS2**, **Contr_LOOBS1_D** and **Contr_LOOBS2_D** and these contracts still hold. This implies that replacing the power supply is rated as a GREEN change which means — according to Table I — that there is no need to make any structural changes to the system design nor the safety argument. However, a manual check for the argument is still needed to replace out of date information about the old power supply with new valid information. For example, the description which the context **CxtPSDesc** refers to (in Figure 8) is out of date and should be replaced by the new power supply description. Table II summarises the impact on the FTA, system design and the safety case due to the power supply replacement.

- 5) **Re-balance the FPs of the FTA's events as a preparation for future changes:** In this step, we need

TABLE II. POTENTIAL IMPACTS OF A CHANGE

Change No.	Source of Impact (Design)	Source of Impact (FTA)	Impact size on FP of the source event (%)	No. of affected Events	No. of Affected Components	Affected Argument elements	Affected Evidence
1	BSCU System 1 power supply	BSS1PSF	Current FP: 6.75E-05 Increase 344.4%	2	BSCU System 1 Power Supply	0*	Evidence supporting claims about old power supply 1
2	BSCU System 2 power supply	BSS2PSF	Current FP: 6.75E-05 Increase 344.4%	2	BSCU System 2 Power Supply	0*	Evidence supporting claims about old power supply 2

* A manual check to all related argument elements is still needed to replace the name, descriptions, specifications, vendor information, etc. of the old power supply model with the new one.

<p>Contr_LOOBS1 G1: The MAFP for the event LOOBS1 is $\leq 1.0157E-03$ A1: No duplicates of LOOBS1 in the FTA where the failure probability $\geq 1.0157E-03$ A2: The logic in WBS1_FTA remains the same (Option 1.) A3: BSS1EF MAFP $\leq 6.771E-04$ A4: BSS1PSF FP $\leq 3.3857E-04$ (Option 2.) A3: BSS1EF MAFP $\leq 7.157E-04$ A4: BSS1PSF FP $\leq 3.00E-04$ (No Change) (Option 3.) A3: BSS1EF FP $\leq 1.50E-04$ (No Change) A4: BSS1PSF MFP $\leq 8.657E-04$</p>
--

Fig. 10. The updated version of *Contr_LOOBS1* after Scenario 1

to re-balance the FPs of the FTA's events. In other words, the reduction in the margins of the **BSS1PSF** and **BSS2PSF** FPs should be shared by all of the events in the FTA. That is, all current FPs should contribute to make up the contraction of BSS1PSF and BSS2PSF FP margins due to the power supply replacement. To do so, we need to:

- a) Find $\Delta FP_{(Topevent)}$ which is the difference between the required FP (i.e., **3.30E-05**) and the new $FP_{Current(Topevent)}$ after containing the change which we have determined in Step 4 (i.e., **1.646E-06**). $\Delta FP_{(Topevent)} = \mathbf{3.136E-05}$.
 - b) Repeat Step 2 to distribute **3.136E-05** over all FPs' margins in the FTA (i.e., calculate the new MAFP for all events including BSS1PSF and BSS2PSF). The grey squashed rectangles in Figure 7 represent the new calculated MAFPs after replacing the power supply. To verify that the calculation of the new MAFPs was done successfully, the determined MC can be used to check whether the MAFP of the top event remains the same.
- 6) **Update the affected safety contracts with the new FPs:** Since new MAFPs have been calculated for all the FTA events, all derived safety contracts should be updated to reflect the new MAFP values. Figure 10 shows the updated version of *Contr_LOOBS1*.

Change request scenario (2)

This scenario is similar to scenario (1). The only difference though is the FP value of the new power supply model, which is in this case equals to **5.00E-03**, which means that it is 7307.41% greater than the current FP (i.e., $6.75E-05$). Since Steps 6 and 7 are very similar to what we have done in the previous scenario and for the sake of the space, we will apply only Step 4 in this scenario, as follows:

Check the ability of the FTA to contain the change. As a quick check, the FP of the top event after introducing the change is **2.8106E-05**, which means that it is $< FP_{Required(Topevent)}$. Hence, the change is containable and we need to find the safety contract which contains it. Again in this scenario the very direct contracts that are affected by the change are **Contr_LOOBS1**, **Contr_LOOBS2**, **Contr_LOOBS1_D** and **Contr_LOOBS2_D**. The FP of the

new power supply **5.00E-03 + 1.50E-04** (another assumption in the contracts) = **5.15E-03** and this is greater than the guaranteed MAFP in those contracts (i.e., **1.018E-03**). Hence, all direct contracts are broken and cannot contain the change, and all of the events in their assumption lists are affected by the change. Consequently, safety contracts of higher events in the FTA should be checked for the change containment. Since each pair of the contract are derived for events that are duplicated in two different locations in the FTA, we need to check the safety contracts of the parent events in the two locations: 1) for **Contr_LOOBS1** and **Contr_LOOBS2**, we should check **Contr_BSS1&2DNO**, and 2) for **Contr_LOOBS1_D** and **Contr_LOOBS2_D**, we should check **Contr_SWFSTS1PAS1F** and **Contr_SWFSTS2PAS2F**.

Before start checking **Contr_BSS1&2DNO**, the new FP of **BSS1&2DNO** after the change should be calculated. Since **LOOBS1** and **LOOBS2** are the events that make up **BSS1&2DNO** and they are connected through an AND gate, this implies **5.15E-03 * 5.15E-03 = 2.65225E-05**.

However, the guaranteed FP for **BSS1&2DNO** is $1.0362E-06$ which is $<$ its calculated FP after introducing the change. That is, **Contr_BSS1&2DNO** is broken, where **LOOBS1** and **LOOBS2** are declared as affected events. To proceed, a higher safety contract for a higher event in the FTA should be investigated. The direct parent event for **BSS1&2DNO** is the top event itself (i.e., **BSFCLOBC**).

The derived safety contract for the top event (**Contr_BSFCLIBC**) guarantees the MAFP for the entire FTA and it assumes that the MAFPs of **BSS1&2DNO**, **BSVMIRFCSTA** and **SWFCTLOBBC** are **2.1571E-06**, **8.00E-07** and **3.005E-05**, respectively. Since **2.65225E-05 > 2.1571E-06**, this contract still does not contain the change. The only available solution is to contain the change is to check whether or not the added margin to the **SWFCTLOBBC** FP has $2.43654E-05$ ($2.65225E-05 - 2.1571E-06$) as a surplus to contain the change. To do that, the other location of the FTA, where **BSS1PSF** and **BSS2PSF** are duplicated should be checked whether or not it contains the change. **Contr_SWFSTS1PAS1F** and **Contr_SWFSTS2PAS2F** are broken and all of the events in their assumption lists are affected. This is because **5.15E-03 * 1.30E-05** (**SWFSIS1P** or **SWFSIS2P**) = **6.695E-08** and this is greater than what **Contr_SWFSTS1PAS1F** and **Contr_SWFSTS2PAS2F** guarantee. Consequently, we need to check **Contr_SWFCTLOBBC** as it is the contract of the parent event of both **SWFSTS1PAS1F** and **SWFSTS2PAS2F**.

The contract in question guarantees the MAFP to be **1.4432E-05** and assumes the FPs of **SWFSTS1PAS1F**, **SWFSTS2PAS2F** and **SWFSIIP** as **6.875E-06**, **6.875E-06** and **6.50E-07**, respectively. Since **6.875E-06 > 6.695E-08** this contract contains the change and prevents its effect to ripple up further. Now we need to ensure that the first location prevents the ripple of the change's effect. To this end, we calculate the surplus in the guaranteed MAFP in **SWFCTLOBBC** and **BSVMIRFCSTA**, and then use this surplus to compensate the deficit in the FP of **BSS1&2DNO**. The minimum required FP for **SWFCTLOBBC** is: **6.695E-08 + 6.695E-08 + 6.50E-07 = 7.839E-07**, and **BSVMIRFCSTA** will remain **8.00E-07** as it is not affected by the change. This means that the total surplus

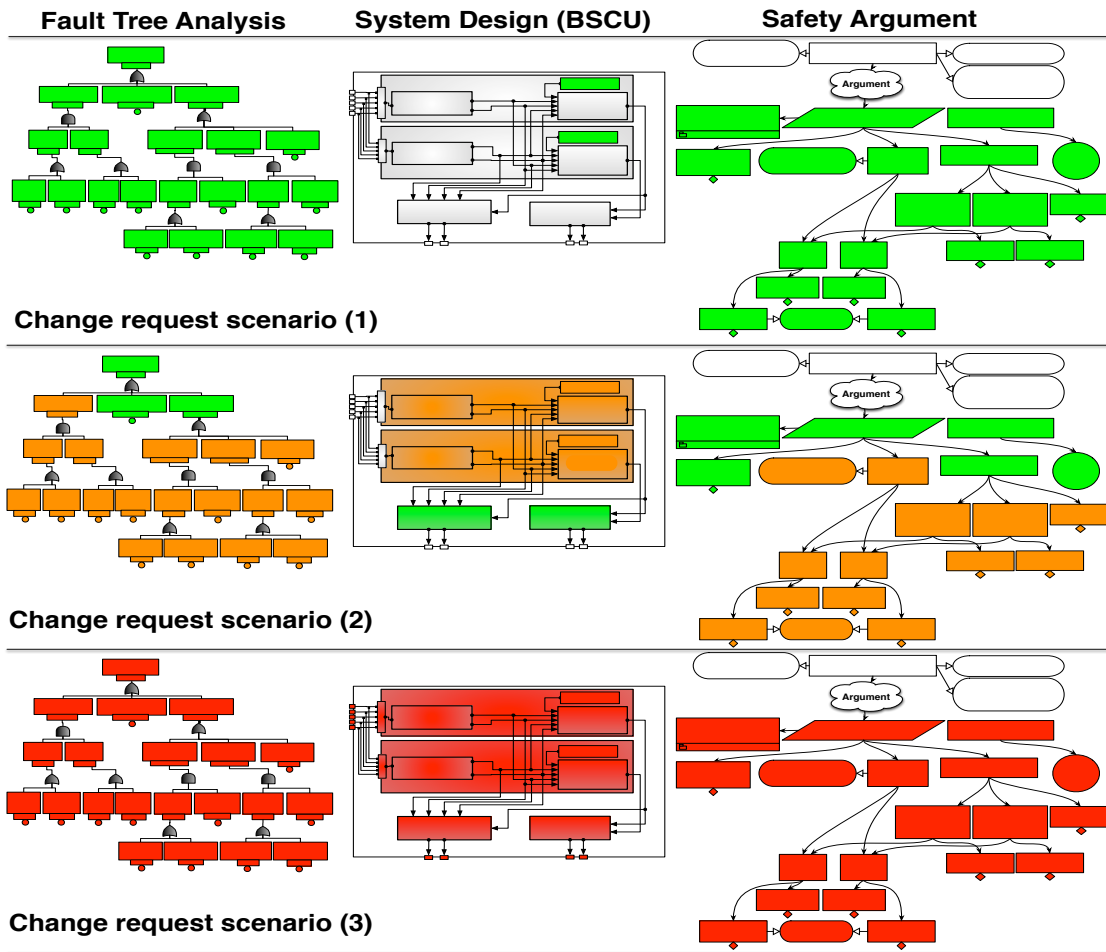


Fig. 11. The effect of change on the FTA, system design and the safety argument: An overview of the three scenarios

is $3.3E-05 - (7.839E-07 + 8.00E-07) = 3.14161E-05$. That is, the surplus can compensate the deficit in **Contr_BSS1&2DNO** since it is $> 2.43654E-05$ and thus the effect of change is also contained in this contract.

Change request scenario (3)

This scenario is similar to the previously discussed scenarios (2) and (3). The difference here is that the FP value of the new power supply model is equal to $6.00E-03$, which means that it is 8788.89% greater than the current FP (i.e., $6.75E-05$). As a quick check, the FP of the top event should be calculated in light of the new value of the power supply FP. The new calculated FP for the top event is equal to $3.9432E-05$ and it is $> 3.3E-05$ (the MAFP for the top event). That is, the resultant change effects due to replacing the power supply by this specific model is not containable and the entire FTA is going to be impacted. Hence, the WBS cannot meet its current safety requirements without considering a major structural changes/updates.

Figure 11 shows a high level view of the change effects in the FTA that is caused by replacing the power supply in the three discussed change scenarios. The figure also shows how the safety contracts are used to highlight the affected parts in the WBS design and the safety argument.

VI. CONCLUSION AND FUTURE WORK

Changes are often only performed long time after the initial design of the system making it hard for the system designers to know the impact of these changes on their system and its safety case. In our previous works [17], [24] we introduced SANESAM and SANESAM+ as techniques to facilitate the maintenance of safety cases using safety contracts. In this paper, we use the key principle of SANESAM and SANESAM+ to introduce a new technique that can save huge efforts in re-verification or re-certification due to some design changes. The technique can serve as a first impact analysis layer that helps system's developers to estimate the size of effort needed to accommodate a design change. The technique can also guide the developers to avoid massive re-engineering efforts when it is not really needed. Although the technique can be effective in maintaining safety systems and safety cases, the scope of the changes addressed by it may seem limited in the general maintenance scenario. However, these types of changes are the most critical from a safety perspective and they are worth making the emphasis. Future work will focus on considering different properties other than failure probabilities (e.g., timing) in order to consider additional types of changes. In addition, development of an automation tool is considered as a potential direction. We also intend to perform a case study to validate both the feasibility and efficacy of the technique.

ACKNOWLEDGMENT

This work has been partially supported by the Swedish Foundation for Strategic Research (SSF) (through SYNOPSIS and FiC Projects) and the EU-ECSEL (through SafeCOP project). We thank Sasikumar Punnekkat for his fruitful help and comments.

REFERENCES

- [1] GSN Community Standard, Version 1; (c) 2011 Origin Consulting (York) Limited. <http://www.goalstructuringnotation.info>.
- [2] SAE ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, Dec. 1996.
- [3] *Oxford Dictionary of English (3 ed.)*. Oxford University Press, 2010.
- [4] GSN community standard version 1. Technical report, Origin Consulting (York) Limited, Nov. 2011.
- [5] I. Bate, H. Hansson, and S. Punnekkat. Better, faster, cheaper, and safer too – is this really possible? In *Proceedings of the 17th IEEE International Conference on Emerging Technologies for Factory automation*, 2012.
- [6] 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 on Formal Methods for Components and Objects*, pages 200–225, Springer Berlin Heidelberg, 2007.
- [7] 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, HSCC '08*, pages 58–71, Berlin, Heidelberg, 2008. Springer-Verlag.
- [8] L. Breierova and M. Choudhari. An introduction to sensitivity analysis. Technical report, Massachusetts Institute of Technology (MIT), September 1996.
- [9] R. Cleaveland. Formal certification of aerospace embedded software. In *National Workshop on Aviation Software Systems: Design for Certifiably Dependable Systems*, 2006.
- [10] A. Cullen and H. Frey. *Probabilistic techniques in Exposure assessment*. Plenum Press, New York, 1999.
- [11] H. Espinoza, A. Ruiz, M. Sabetzadeh, and P. Panaroni. Challenges for an open and evolutionary approach to safety assurance and certification of safety-critical systems. In *Proceedings of the 1st International Workshop on Software Certification (WoSoCER)*, pages 1–6, Nov 2011.
- [12] 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.
- [13] I. Habli and T. Kelly. Process and product certification arguments: Getting the balance right. *SIGBED Rev.*, 3(4):1–8, Oct. 2006.
- [14] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, Oct. 1969.
- [15] J. L. Fenn, R. Hawkins, P. J. Williams, T. Kelly, M. G. Banner, 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*, pages 135–140. IET, 2007.
- [16] O. Jaradat. Enhancing the maintainability of safety cases using safety contracts. Mälardalen University, Västerås, Sweden. <http://www.es.mdh.se/publications/4082->, November 2015.
- [17] O. Jaradat and I. Bate. Deriving hierarchical safety contracts. In *The 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2015)*, November 2015.
- [18] T. Kelly. Introduction to safety cases, lecture notes, 2007. [online]. available: http://www.omg.org/news/meetings/workshops/SWA_2007_Presentations/00-T3_Kelly.pdf.
- [19] T. Kelly. A systematic approach to safety case management. In *Proceedings of SAE 2004 World Congress, Detroit*. The Society for Automotive Engineers, March 2004.
- [20] 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 *The 24th International System Safety Conference (ISSC)*, Albuquerque, USA, 2006.
- [21] R. Maguire. *Safety Cases and Safety Reports: Meaning, Motivation and Management*. Ashgate Publishing, Ltd., 2012.
- [22] B. Meyer. Design by contract. Technical Report TR-EI-12/CO, Interactive Software Engineering Inc., 1986.
- [23] B. Meyer. *Object-Oriented Software Construction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1st edition, 1988.
- [24] 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 (Ada-Europe)*, pages 162–176, June 2015.
- [25] O. Jaradat, P. Graydon and I. Bate. An approach to maintaining safety case evidence after a system change. In *Proceedings of the 10th European Dependable Computing Conference (EDCC)*, Newcastle, UK, August 2014.
- [26] Object Management Group (OMG). *Structured Assurance Case Meta-model (SACM)*, Technical report, Version 1.0, 2013. [Online]. Available: <http://www.omg.org/spec/SACM/1.0/PDF/>.
- [27] P. Conmy, J. Carlson, R. Land, S. Björnander, O. Bridal, I. Bate. Extension of techniques for modular safety arguments. Deliverable d2.3.1, technical report, Safety certification of software-intensive systems with reusable components (SafeCer), 2012.
- [28] D. J. Pannell. Sensitivity analysis of normative economic models: theoretical framework and practical strategies. *Agricultural Economics*, 16(2):139 – 152, 1997.
- [29] J. Pate, K. R. Edlin, and K. I. Kawano. Sensitivity analysis of hardware-in-the-loop (HWIL) simulation systems. In *Proceedings of Modelling and Simulation*. ACTA, May 2005.
- [30] M. Rausand and A. Høyland. *System Reliability Theory: Models, Statistical Methods and Applications*. Wiley-Interscience, Hoboken, NJ, 2004.
- [31] S. Bauer, A. David, R. Hennicker, K. G. 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, FASE'12*, pages 43–58, Berlin, Heidelberg, 2012. Springer-Verlag.
- [32] A. Saltelli. *Global sensitivity analysis: the primer*. John Wiley, 2008.
- [33] P. Shankar, J. Mathieson, R. Ramachandran, J. D. Summers, and G. M. Mocko. Can design evaluation tools predict/prevent change propagation? In *Proceedings of Tools and Methods of Competitive Engineering*, 2012.
- [34] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback. *Fault Tree Handbook with Aerospace Applications*. Handbook, National Aeronautics and Space Administration, Washington, DC, 2002.
- [35] N. R. Storey. *Safety Critical Computer Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.
- [36] T. Kelly and J. McDermid. A systematic approach to safety case maintenance. In *Proceedings of the Computer Safety, Reliability and Security*, volume 1698 of *Lecture Notes in Computer Science*, pages 13–26. Springer Berlin Heidelberg, 1999.
- [37] N. Tracey, A. Stephenson, J. Clark, and J. McDermid. A safe change oriented process for safety-critical systems. *Heslington, York*, 1999.
- [38] U.K. Ministry of Defence. *00-56 Defence Standard — Safety Management Requirements for Defence Systems*, December 1996.
- [39] W. Damm, H. Hungar, J. Bernhard, 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*, pages 1–6, 2011.
- [40] R. B. Whittingham. *Preventing corporate accidents : An Ethical Approach*. Elsevier Ltd, 2008.