A Functional Failure Analysis Method of Identifying and Mitigating Spurious System Emissions From a System of Interest in a System of Systems

Van Bossuyt, Douglas L.; Arlitt, Ryan M.

Published in:
Journal of Computing and Information Science in Engineering

Link to article, DOI:
10.1115/1.4046991

Publication date:
2020

Document Version
Peer reviewed version

Citation (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
A Functional Failure Analysis Method of Identifying and Mitigating Spurious System Emissions from a System of Interest in a System of Systems

Douglas L. Van Bossuyt∗
Systems Engineering Department
Naval Postgraduate School
Monterey, California 93943
Email: douglas.vanbossuyt@nps.edu

Ryan M. Arlitt
Department of Mechanical Engineering
Technical University of Denmark
DK- 2800 Kgs. Lyngby
Denmark
Email: rmarl@mek.dtu.dk

Increasingly tight coupling and heavy connectedness in systems of systems (SoS) presents new problems for systems designers and engineers. While the failure of one system within a loosely coupled SoS may produce little collateral damage beyond a loss in SoS capability, a highly interconnected SoS can experience significant damage when one member system fails in an unanticipated way. It is therefore important to develop systems that are “good neighbors” with the other systems in a SoS by failing in ways that do not further degrade a SoS’s ability to complete its mission.

This paper presents a method to (1) analyze a system of interest (SoI) for potentially harmful spurious system emissions (failure flows that exit the SoI’s system boundary and may cause failure initiating events in other systems within the SoS), and (2) choose mitigation strategies that provide the best return on investment for the SoS. The method is intended for use during the system architecture phase of the system design process where functional architectures are under development, and analysis of alternatives and trade-off studies are being conducted.

1 INTRODUCTION

1.1 SPECIFIC CONTRIBUTIONS

In this paper, we present a method intended for the early system architecture phase of the system engineering process where functional architectures are under development. The method helps to identify and mitigate potential failure flows exiting a SoI’s system boundary – spurious system emissions (SSEs) – that otherwise may not have been identified or may have been discounted. This method helps to develop strategies to mitigate SSEs from the very earliest functional modeling efforts of a new SoI. Systems engineers can use the method to identify potential SSE sources from a SoI into the SoS, and propose mitigation strategies to address the identified SSEs. While other methods such as PRA and FMECA can be used to investigate potential SSEs, those methods are generally employed either later in the system development process after system architectures have been selected and frozen or do not directly integrate into existing functional

∗Address all correspondence to this author.

1A version of this paper appeared at the 2019 ASME IDETC/CIE [1].
2 BACKGROUND AND RELATED WORK

The work presented in this article is set within the context of the systems engineering process that takes a system from initial concept to production, customer delivery and use, maintenance and upgrade, and disposal [2, 3]. Of particular interest to this research is the early phase of systems engineering that is encompassed by the system architecting process where customer needs statements, design reference missions, system requirements, functional system models and architectures, trade-off studies, and a variety of other activities occur [3, 4]. Mission engineering [5–8] and SoS engineering also play a significant role in many SoI system architecting efforts [9, 10].

Functional modeling helps to develop an understanding of how systems work at the functional level [11] during system architecture development. There are a variety of different taxonomies available to produce functional models [12]. We prefer the Functional Basis for Engineering Design (FBED) [13, 14] and use it throughout this paper. A functional modeling taxonomy generally is composed of functions and flows where functions transform incoming flows to different outgoing flows. Functions and flows are connected to their physical component solutions through databases and repositories [15–17]. One function may have many potential component solutions such as a convert electrical energy to rotational energy function may be satisfied by several types of electrical motors; flows can similarly have multiple physical manifestations [18].

Throughout the system design process, a variety of failure analysis methods are often employed such as FMECA early in a system design process [19] and PRA which often is done after major system architecture decisions have been made [20]. PRA uses the concept of an initiating event – an event that is the starting point for a failure that propagates through a system [21]. Many systems engineered using PRA have the ability to react to incipient failures and either transition to a safe shutdown state or continue operating either nominally or in a degraded state while repairs are made [22, 23] although unanticipated failures can still occur which may lead to system failure.

Reliability block diagrams (RBDs) can be used to analyze the reliability of a system either from the component or functional level [24]. The function failure identification and propagation (FFIP) family of methods extends the concept of RBDs to understand how failures propagate through a system, how to detect incipient failures, and what may be done to mitigate such failure events [25, 26]. Some work has been done that includes the authors of this paper to understand how to redesign an SoI at the functional level to withstand SSEs that enter the SoI as unanticipated failure initiating events [27]. Our previous work is in contrast with the method advanced in this paper that specifically focuses on not allowing SSEs from a SoI to occur that may negatively affect the rest of a SoS.

In summary, within the scope of early system architecting efforts where functional models of a SoI are under development and major architectural design decisions have not been finalized, no existing method that we are aware of is available to systems engineers to systematically identify potential SSEs and propose mitigation strategies from a functional perspective.

3 METHODOLOGY

In this section we present a novel method to identify potential SSEs originating in a SoI that could negatively impact other members of a SoS, quantify potential SSE probabilities, identify potential mitigation strategies to prevent SSEs from causing harm to other systems in the SoS, and conduct trade-off studies to determine the best course of action moving forward with the SoI system design. The method is intended to be used during system architecture phase of system design where large changes to a SoI design can be made with relatively little impact to cost or schedule. Figure 1 depicts the seven steps of the methodology, the reusable dependencies, and the preparatory step; and their relations to one another.

3.1 CASE STUDY

In order to demonstrate and illustrate the method throughout the methodology section of this paper, we now introduce an illustrative case study of an autonomous vehicle SoI that is being designed to enter service with an autonomous logistics system (the SoS). The SoI is currently in the system architecture phase of the system design process and specifically the functional architecture is being refined. The SoS operates in a mountainous desert environment carrying materiel from a logistics depot to a forward operating base. This frees up military personnel and contractors from routine and potentially dangerous resupply missions [28] to concentrate on other high value activities. There are other constituent members of the SoS including the logistics depot and co-located ground control station, command and control relay stations, and other autonomous systems such as autonomous ground vehicles. Figure 2 shows a high level operational view of the SoS.

The system architecture process for the SoI has already down-selected to the production of an unmanned aerial vehicle (UAV) for the specific payloads and mission constraints identified during the development of the customer needs statement, the design reference mission, and the system requirements (shown in Table 1).

While the SoI presented in the case study of this paper is idealized, it takes inspiration from real systems [7, 29]. The case study is used to illustrate each step of the method presented below and has been intentionally simplified to better highlight the specific contributions of the method.

3.2 REUSABLE DEPENDENCIES

Prior to beginning the preparatory step of the method, reusable dependencies (e.g.: function to component relational database, cost and performance data, functional failure
modes, etc.) are identified and developed as necessary, and are specific to the SoI. Once created, these resources can support multiple SoI analyses when the SoIs are deemed similar enough by practitioners. Reusable dependency databases include historical function to component relationships [16, 17], function to failure data and relationships [26, 30, 31], cost and performance data of components (seeded automatically where practical [32]), and abstracted behavior failure behavior models [26, 30].

3.3 PREPARATORY STEP

Several preparations must be made within two categories: 1) prepare a FFIP model of the system, and 2) prepare information for the trade-off study conducted in Steps 6 and
Table 1. SoI requirements for a UAV used to carry cargo.

<table>
<thead>
<tr>
<th>Req #</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Carry 10kg 5km</td>
</tr>
<tr>
<td>#2</td>
<td>Complete round-trip transit with 99% success rate</td>
</tr>
<tr>
<td>#3</td>
<td>Communicate with ground control station at 1.5Mbit/sec TX/RX</td>
</tr>
</tbody>
</table>

3.4 STEP 1: ANALYSIS OF EACH FUNCTION AND WHAT IT CONCEIVABLY COULD EMIT

Previous to Step 1, reusable dependencies including a function to component relational database containing failure modes information were developed. Now the failure modes must be expanded to go beyond failures that have been previously observed. To identify a high proportion of all possible emitted failure flows beyond what has previously been identified with existing methods, we advocate working backwards from the flow taxonomy of FBED to disprove the hypothesis that each of the flow types can be emitted as a failure flow by the function in question. For instance, the Energy-Thermal flow may be generated by a function such as Control-Stop-Inhibit where the component solution is a metal barrier if the function receives a failure flow input such as Energy-Vibration where the flow’s physical solution is a high frequency, high amplitude vibration caused by an unanticipated failure somewhere else in the SoI. Table 3 presents an example of a function where potential received failure flows are connected to emitted failure flows and associated potential component solutions to the function. The crossed-out failure flow exports represent those exports that have been found to be impossible to create regardless of the failure flow import to the function.

The newly identified failure flow exports shown in Table 3 from the function are then appended to the function’s failure database entry.

3.5 STEP 2: EVALUATE ALL POTENTIAL FLOW PATHS THROUGH THE SYSTEM OF INTEREST

Next, a partial re-evaluation of the FFIP model of the SoI is conducted. All potential failure propagation paths that lead to a failure flow exiting the system boundary are identified as per the FFIP method; however, we do not assign probabilities to individual flow paths, functional failure events, or initiating events at this point in time. The proposed method intentionally does not assign probabilities at this step to avoid the pitfalls of truncation of failure flow paths that often occurs during FFIP-style analyses. A small sample of failure flow paths that exit the system boundary from the SoI are presented in Table 4.

3.6 STEP 3: DETERMINE PROBABILITIES OF SSEs EXITING THE SYSTEM OF INTEREST

After all failure flow paths have been identified, the next step is to quantify the probability of each failure flow emitted as a SSE from the SoI. This step diverges from established FFIP practices. Rather than stopping at producing cut-sets (the paths that a failure follows from initiation to exiting the system as a SSE) that are analyzed individually, the probability of each SSE is developed from aggregating cut-sets into groups based on the specific SSEs that they produce.

Table 5 shows a representative subset of cut-sets for the UAV SoI that only includes the SSEs identified through this method. Each SSE type and probability of occurrence, \( P_{O_e} \), is listed where the probability is an aggregation of all SSE failure flow path cut-sets that result in that particular failure flow emission type.

3.7 STEP 4: ANALYZE RESULTS

Next, the results of the prior steps are analyzed to understand the potential consequences to the other systems within the SoS. Table 2 provides a means for evaluating how SSEs from the SoI may impact other systems in the SoS. A mapping of SSEs produced by a SoI to initiating events of other systems in the SoS with a probability of negative conse-
Fig. 3. FFIP model of the SoI UAV developed using the FBED taxonomy. Note the dotted and dashed line represents the system boundary. The dashed line represents a SSE.

<table>
<thead>
<tr>
<th>Failure Flow Exports from System of Interest that Lead to Initiating Events for Other Systems in the SoS</th>
<th>Consequence</th>
<th>$C_e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy-Electrical</td>
<td>Static-electric discharges during dust storms caused by UAV rotors or propellers can short out onboard electronics of nearby vehicles leading to loss of both UAVs and UGVs</td>
<td>$5M$</td>
</tr>
<tr>
<td>Material-Solid-Particulate</td>
<td>Large particulate from crashed UAVs can clog air vents and cause overheating of UGVs leading to disabled systems</td>
<td>$1M$</td>
</tr>
<tr>
<td>Material-Control-Analog</td>
<td>Interference with radio transceivers causes UAVs to automatically land regardless of terrain or of potential adversary presence</td>
<td>$2M$</td>
</tr>
</tbody>
</table>

Table 2. Consequence data for an mixed UAV and UGV SoS where consequence the cost distribution function $C_e$ for the impact of the system on the SoS. In this example, each $C_e$ is a point distribution.

The next step is to develop approaches to mitigate SSEs before they leave the SoI. In this method, we advocate for addressing SSEs before they leave the system boundary in

$$EP_e = P_e \times C_e$$  \hspace{1cm} (1)

### 3.8 STEP 5: IDENTIFY SSE MITIGATION STRATEGIES

The next step is to develop approaches to mitigate SSEs before they leave the SoI. In this method, we advocate for addressing SSEs before they leave the system boundary in
<table>
<thead>
<tr>
<th>Failure Flow Imports</th>
<th>→</th>
<th>Failure Flow Exports</th>
<th>Component Solution(s) to Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>Secondary</td>
<td>Tertiary</td>
<td>Primary</td>
</tr>
<tr>
<td>Energy</td>
<td>Mechanical</td>
<td>Translational</td>
<td>→</td>
</tr>
<tr>
<td>Energy</td>
<td>Electrical</td>
<td>→</td>
<td>Liquid</td>
</tr>
<tr>
<td>Energy</td>
<td>Mechanical</td>
<td>Translational</td>
<td>→</td>
</tr>
<tr>
<td>Energy</td>
<td>Mechanical</td>
<td>Translational</td>
<td>→</td>
</tr>
<tr>
<td>Energy</td>
<td>Mechanical</td>
<td>Pneumatic</td>
<td>→</td>
</tr>
<tr>
<td>Energy</td>
<td>Electrical</td>
<td>→</td>
<td>Control</td>
</tr>
<tr>
<td>Energy</td>
<td>Electromagnetic</td>
<td>Solar</td>
<td>→</td>
</tr>
<tr>
<td>Energy</td>
<td>Electromagnetic</td>
<td>Solar</td>
<td>→</td>
</tr>
<tr>
<td>Energy</td>
<td>Electrical</td>
<td>→</td>
<td>Electromagnetic</td>
</tr>
<tr>
<td>Energy</td>
<td>Radioactive/Nuclear</td>
<td>→</td>
<td>Thermal</td>
</tr>
</tbody>
</table>

Table 3. Above is an example of examining a Channel-Guide-Rotate function to understand which potential failure flows can occur. Failure flows generated by specific component solutions are indicated on the right hand side of the table. Failure flow exports that have been reasonably proven to be impossible for the function to emit have been crossed out. In certain cases, multiple failure flow exports may be developed from the same failure flow import. Additionally, some failure flow exports may have multiple associated component solutions to the function or one component solution to the function may be associated with multiple potential failure flow exports.
Table 4. Subset of failure flow paths of the UAV SoI that exit the system boundary.

<table>
<thead>
<tr>
<th>Flow Path #</th>
<th>Failure Flow Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Energy-Electrical → Provision-Supply → Energy-Electrical → Channel-Export → Signal-Control-Discrete</td>
</tr>
<tr>
<td>#2</td>
<td>Material-Mixture-Gas/Solid → Channel-Guide-Translate → Material-Solid</td>
</tr>
<tr>
<td>#3</td>
<td>Provision-Supply → Energy-Electrical → Channel-Guide-Rotate → Energy-Thermal</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 5. Probability of occurrence of a representative subset of failure flows identified through the proposed method that exit the SoI boundary as SSEs.

<table>
<thead>
<tr>
<th>Failure Flows that Result in System Emissions</th>
<th>PO&lt;sub&gt;e&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy, Mechanical, Translational</td>
<td>2.2E-4/year</td>
</tr>
<tr>
<td>Material, Gas</td>
<td>4.3E-3/year</td>
</tr>
<tr>
<td>Signal, Status, Visual</td>
<td>5.6E-3/year</td>
</tr>
<tr>
<td>Material, Solid, Particulate</td>
<td>1.9E-2/year</td>
</tr>
<tr>
<td>Material, Liquid</td>
<td>8.3E-3/year</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

In order to understand what mitigation strategies are preferred, we proposed developing a mitigation probability (MP) which is similar in formulation to EP. To calculate a population of MPs where one mitigation strategy may be useful in mitigating several SSEs (or one emission may be addressed to different extents by different mitigation strategies), a matrix is populated with EPs that reflect a reduction in <i>P<sub>e</sub></i>, denoted by EP<sub>Reduced</sub>. Equation 2 demonstrates how to calculate EP<sub>Reduced</sub> where <i>e</i> is the SSE and <i>m</i> is the mitigation strategy.

$$\text{EP}_{\text{Reduced}}(e,m) = P_{Me(e,m)} \times C_{e(m)}$$  (2)

An example of the matrix that is populated by Equation 2 is shown in Matrix 3. Here, each row corresponds to a failure emission while each column corresponds to a mitigation strategy. Many of the cells resolve to zero in this matrix, indicating that the mitigation strategy has no impact on that emission.

Next, MP can be calculated for the matrix produced from Equation 2 as shown in Equation 4. In this equation, \(\vec{\text{EP}}\) represents a column vector of the original EPs as identified in Table 2.

$$\text{MP} = \vec{\text{EP}}^T \times \text{EP}_{\text{Reduced}} + M_C$$  (4)

3.9 STEP 6: DETERMINE WHAT MITIGATION STRATEGIES TO IMPLEMENT

In order to determine what mitigation strategies to implement in a SoI to reduce its potential negative impact on the SoS, we propose the mitigation rank priority (MRP) metric which converts MP into a metric than can be more easily rank-ordered. Equation 5 shows how to determine MRP for a specific mitigation strategy, <i>m</i>.

$$\text{MRP}_m = \text{rank}(\max(MP_m)) + \text{rank}(\text{mean}(MP_m)) + \text{rank}(\text{std}(MP_m))$$  (5)

MRP as presented in Equation 5 is only one potential formulation of MRP, where each term corresponds to the estimated worst case (max), average case (mean), and predictability (standard deviation) of the distribution. Practitioners may wish to change the formulation depending on, for instance, how much confidence they have in their data sources. The important aspect of MRP for the purposes of this method is that it can be used to develop rank orderings and trade space exploration analysis which may be useful to SoI system stakeholders and decision-makers for their understanding of SSE risks and mitigation strategies. In short: MRP helps in the communication of risk management to stakeholders and decision-makers.

At this point in the method, trade-off studies and optimization can be conducted between major SoI system constraints and requirements, mitigation strategies and their corresponding reduction in probability of a SSE from leaving...
Table 6. Representative subset of SSE EPs of the UAV Sol. $P_e$ comes from Table 5 and $C_e$ is from Table 2. Entries are rank ordered by EP to indicate which SSEs have the biggest negative impact on the SoS.

<table>
<thead>
<tr>
<th>Spurious System Emission</th>
<th>$P_e$</th>
<th>$C_e$</th>
<th>EP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy-Mechanical-Translational</td>
<td>5.2E-4/year</td>
<td>$5M$</td>
<td>$2600/yr$</td>
</tr>
<tr>
<td>Material-Solid-Particulate</td>
<td>1.9E-5/year</td>
<td>$1M$</td>
<td>$19/year$</td>
</tr>
<tr>
<td>Material-Control-Analog</td>
<td>2.6E-4/year</td>
<td>$2M$</td>
<td>$520/year$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(3)</td>
</tr>
</tbody>
</table>

the SoI system boundary that adversely impacts other systems within the SoS, and other important system performance metrics. An approach we suggest is to maximize total MRP (sum of all $MRP_m$ identified for implementation) within the constraint of a cost cap on total mitigation cost ($M_C$) for the SoI.

3.10 STEP 7: ITERATE AND REANALYZE

Now that mitigation strategies have been chosen and are ready for implementation into the SoI system functional model, we suggest iterating through the method at least once more to verify that the mitigation strategies have not introduced new failures into the SoI or SSEs that are undesirable. In particular, Table 7 indicates if there are new SSEs ($P_M f$) created by the proposed mitigation strategies. Re-analysis is further justified by the potential for the failure probability requirements set in the Preparatory Step being violated from unintended consequences of the mitigation strategies. For instance, a new rotor shroud on the UAV SoI may significantly reduce payload capacity thus violating Requirement #1 from Table 1.

Iteration of the SoI system design through the method stops when the requirements set in the Preparatory Step of the method are met. At this point, the practitioner can be relatively confident in a thorough consideration of potential SSEs having been conducted. Further, a practitioner can be reasonably assured that a significant assessment of potential mitigation strategies has been completed. The resulting SoI system design is expected to produce fewer SSEs that may damage other systems within the SoS.

4 DISCUSSION

The method presented above differs from existing methods of identifying and mitigating SSEs in a SoS context in several ways. Most existing methods such as requirements management, PRA and FMECA, and other similar techniques from the systems engineering community either only implicitly suggest that SSEs be examined and mitigated at the conceptual stage of design before functional architecture has been solidified or explicitly examine spurious systems emissions after functional architectures have been finalized and component design has begun. While our previous work looks at SSEs [27], it does so from the perspective of defending against the spurious systems emissions rather than preventing the emissions from occurring in the first place.

Successful implementation of the method in the system architecture phase of system design may benefit systems engineers by aiding in identifying SSEs earlier than they otherwise would have been – specifically during the development of functional architecture models. By identifying potential SSEs very early in the design process, systems engineers can implement strategies to prevent the SSEs from happening as part of the initial system architecting effort rather than implement remediation and/or mitigation strategies much later in the systems design process where costs are much higher and deviation from schedule may be significant.

During the development of the case study, we identified a few of the types of unexpected insights that the proposed method may uncover. For instance, we found that a variety of SSEs may make the UAV SolS in a SoS more detectable by adversaries. We also found that some failures in subsystems such as those involved electronic warfare countermeasures on the UAV Sol may cause SSEs that have a significant detrimental effect to the other systems in the SoS including a disruption in communications. The method provides a framework to not only identify these SSEs and communicate their impacts, but to weigh the trade-offs between mitigation strategies at the functional stage.

The method can be used in parallel on many different SoIs within a SoS which allows for a comparison across all mitigation strategies for all SoIs in a SoS to be conducted.
Table 7. Sample mitigation strategies for a subset of the UAV SoI SSEs.

<table>
<thead>
<tr>
<th>Physical Solution(s)</th>
<th>New Failure Flow(s)</th>
<th>Mitigation Strategy Function(s)</th>
<th>Material, Object</th>
<th>NFFLS?</th>
<th>New Failure Flow(s)</th>
<th>Mitigation Strategy Function(s)</th>
<th>Material, Object</th>
<th>NFFLS?</th>
<th>New Failure Flow(s)</th>
<th>Mitigation Strategy Function(s)</th>
<th>Material, Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shielding to prevent rotor strikes</td>
<td>No</td>
<td>Control, Stop, Inhibit</td>
<td>Material, Solid</td>
<td>Yes</td>
<td>No</td>
<td>Material, Solid</td>
<td>Liquid-Solid</td>
<td>Yes</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redundant control to verify system, control to verify signals before sending</td>
<td>No</td>
<td>Provision, Store, Contain</td>
<td>Material, Mixture</td>
<td>No</td>
<td>No</td>
<td>Material, Mixture</td>
<td>Liquid-Solid</td>
<td>Yes</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capture sub-system, to retain, any liquid generated by failed battery cells</td>
<td>4.7E-5/year</td>
<td>Channel, Export</td>
<td>Material, Liquid</td>
<td>No</td>
<td>No</td>
<td>Material, Liquid</td>
<td>Liquid-Solid</td>
<td>No</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long hose to direct liquid to ground</td>
<td>4.2E-5/year</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

During Step 6 to identify the biggest return on investment to buy down overall risk of SoS failure. Taking a larger SoS-level view may help to save significant cost and drastically increase probability of SoS mission success. An initial investigation of conducting the method in parallel on multiple SoIs indicates that the method presented above is quite extensible and flexible in this regard.

One significant challenge of the method is the amount of effort required to develop the various database products and analyses. However, we argue that similar efforts are needed for PRA and for other FFIP-based methods. In our experience, PRA analysis can be extremely data-intensive and often span many years in the case of complex systems such as nuclear reactors as evidenced by the lengthy PRA process that reactors must undergo before they are certified for construction and use [33].

One limitation of the method is that it is specifically designed to be used in the case where a practitioner has a good understanding of the SoS that the SoI being engineered will be placed within. In the case where an entirely new SoS is under development, additional methodological development is needed to manage the uncertainty posed by the situation. If nothing is known of the SoS, $C_e$ cannot be determined. The only information available to a practitioner would then be $P_0_e$.

Validation of the results of the method is an important step that we advocate be performed by a human. We intend for the method to include a human in the loop at every iteration in order to validate that the results are reasonable. While automating the validation may be possible in the future with a very robust failure and mitigation repository, such an undertaking is very resource-intensive.

A potential fruitful avenue of future work may be to develop a method that integrates failure analysis of an SoS by bridging the method presented in this paper and our previous work [27]. This may provide a new way of making large system architecture decisions while such decisions are still relatively inexpensive to implement. However, the implementation may be computationally prohibitive.

5 CONCLUSION

This paper presented a conceptual design method intended for use during the system architecture phase of the systems engineering process, and specifically during functional architecture development, to identify and mitigate potential SSEs originating from a SoI that can negatively impact a SoS. The method is conducted using functional models which are appropriate for early system architecture trade-off studies. A systematic way to identify potential low probability but high consequence SSEs is presented using the FBED flow taxonomy. Practitioners can use the method to identify and mitigate SSEs from a SoI to prevent damage to other systems within a SoS. An illustrative case study of a UAV SoI being designed to enter service with an existing SoS is presented to demonstrate the method.
Acknowledgements

This research is partially supported by the Naval Postgraduate School and the Technical University of Denmark. Any opinions or findings of this work are the responsibility of the authors, and do not necessarily reflect the views of the sponsors or collaborators.

References

[29] Hunsaker, L., 2015. “Arsenal reaches its ultimate goal of 50 autonomous uavs in flight”.

