2020-10

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.

ASME

Van Bossuyt, Douglas L., and Ryan M. Arlitt. "A functional failure analysis method of identifying and mitigating spurious system emissions from a system of interest in a system of systems." Journal of Computing and Information Science in Engineering 20.5 (2020): 054501.

http://hdl.handle.net/10945/67950

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[1]
Department of Systems Engineering,
Naval Postgraduate School,
Monterey, CA 93943
e-mail: douglas.vanbossuyt@nps.edu

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

Increasingly tight coupling and heavy connectedness in system of systems (SoS) present 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 an SoS by failing in ways that do not further degrade an 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 when functional architectures are being developed, and analysis of alternatives and trade-off studies are being conducted.[2]

[DOI: 10.1115/1.4046991]

Keywords: model-based systems engineering, system of systems, failure analysis, functional model

1 Introduction

As the field of system of systems (SoS) engineering has developed over the last several years, an emerging area of interest is how well member systems behave with each other. Many system engineers desire systems of interest (SoIs) that are "good neighbors" to other systems within the SoS in both nominal operation and in degraded or failed system states. While failure mode effects and criticality analysis (FMECA) and probabilistic risk assessment (PRA) techniques among others are currently being used to help design SoIs in later phases of the system design process, there is a need for an approach that analyzes the effects that an SoI operating in a degraded or failed state has on its SoS neighbors during the early system architecture phase of SoI development. Understanding the effects that an SoI may have on its neighbor systems very early in the system design process allows for large changes to be made at relatively low cost and with minimal impact to a system development schedule.

1.1 Specific Contributions. In this paper, we present a method intended for the early system architecture phase of the systems engineering process where functional architectures are under development. The method helps to identify and mitigate potential failure flows exiting an 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. System engineers can use the method to identify potential SSE sources from an 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 analysis methods.

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 [1,2]. Of particular interest to this research is the early phase of systems engineering that is encompassed by the system architecting process where customer need statements, design reference missions, system requirements, functional system models and architectures, trade-off studies, and a variety of other activities occur [2,3]. Mission engineering [4–7] and SoS engineering also play a significant role in many SoI system architecting efforts [8,9].

Functional modeling helps to develop an understanding of how systems work at the functional level [10] during system architecture development. There are a variety of different taxonomies available to produce functional models [11]. We prefer the functional basis for engineering design (FBED) [12,13] 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 [14–16]. One function may have many potential component solutions, such as converting electrical energy function to rotational energy function may be satisfied by several types of electrical motors and flows can similarly have multiple physical manifestations [17].

Throughout the system design process, a variety of failure analysis methods are often employed such as FMECA early in a system design process [18] and PRA which often is done after major system architecture decisions have been made [19]. PRA uses the concept of an initiating event—an event that is the starting point for a failure that propagates through a system [20]. 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 [21,22] 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 [23]. 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 [24,25]. 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

[1] Corresponding author.

[2] A version of this paper appeared at the 2019 ASME IDETC/CIE (Van Bossuyt, D. L., and Arlitt, R., 2019, "Toward a Functional Failure Analysis Method of Identifying and Mitigating Spurious System Emissions in a System of Systems," International Design Theory and Methodology Conference, IDETC/CIE, ASME, Anaheim, CA).

Manuscript received August 29, 2019; final manuscript received April 8, 2020; published online May 14, 2020. Assoc. Editor: Yan Wang.

events [26]. Our previous work is in contrast with the method advanced in this paper that specifically focuses on not allowing SSEs from an SoI to occur that may negatively affect the rest of an SoS.

In summary, within the scope of early system architecture 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 system 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 an SoI that could negatively impact other members of an 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 the system architecture phase of system design where large changes to an 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 material from a logistics depot to a forward operating base. This frees up military personnel and contractors from routine and potentially dangerous resupply missions [27] 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 downselected 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 [6,28]. 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 [15,16], function to failure data and relationships [25,29,30], cost and performance data of components (seeded automatically where practical [31]), and abstracted behavior failure behavior models [25,29].

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 7 prior to entering the main method.

To develop a FFIP model, a functional model of the SoI must first be developed. In order to do this, a functional taxonomy must be chosen to match the taxonomy used in the reusable dependency

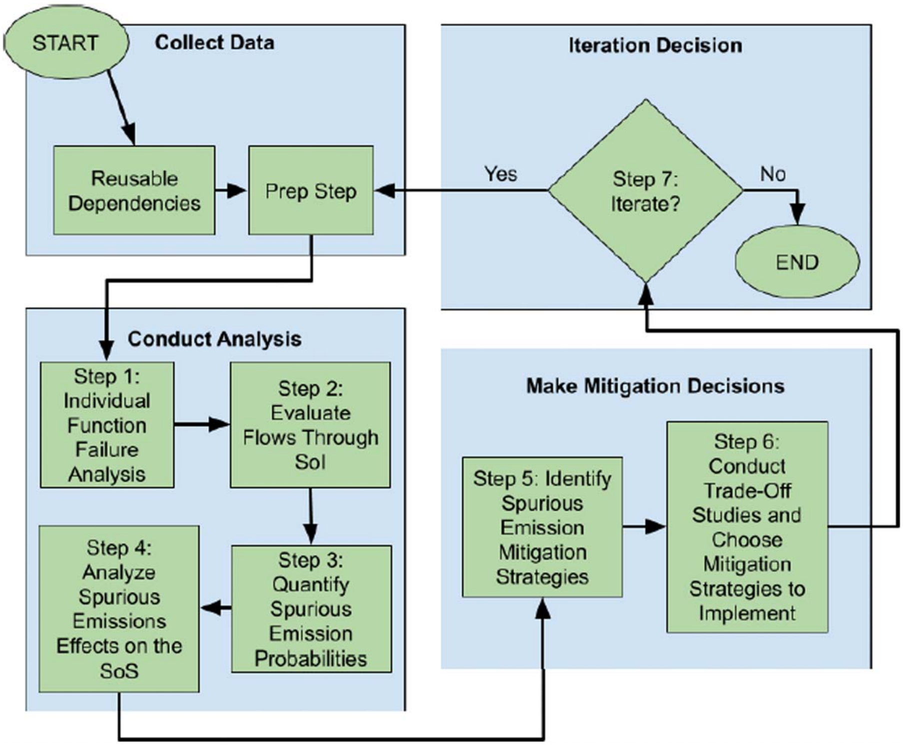
START
Collect Data
Reusable Dependencies
Prep Step
Iteration Decision
Yes
Step 7: Iterate?
No
END
Conduct Analysis
Step 1: Individual Function Failure Analysis
Step 2: Evaluate Flows Through SoI
Step 3: Quantify Spurious Emission Probabilities
Step 4: Analyze Spurious Emissions Effects on the SoS
Make Mitigation Decisions
Step 5: Identify Spurious Emission Mitigation Strategies
Step 6: Conduct Trade-Off Studies and Choose Mitigation Strategies to Implement

Fig. 1 High level flowchart of the relationships between the seven steps of the method, the preparatory step, and the reusable dependencies

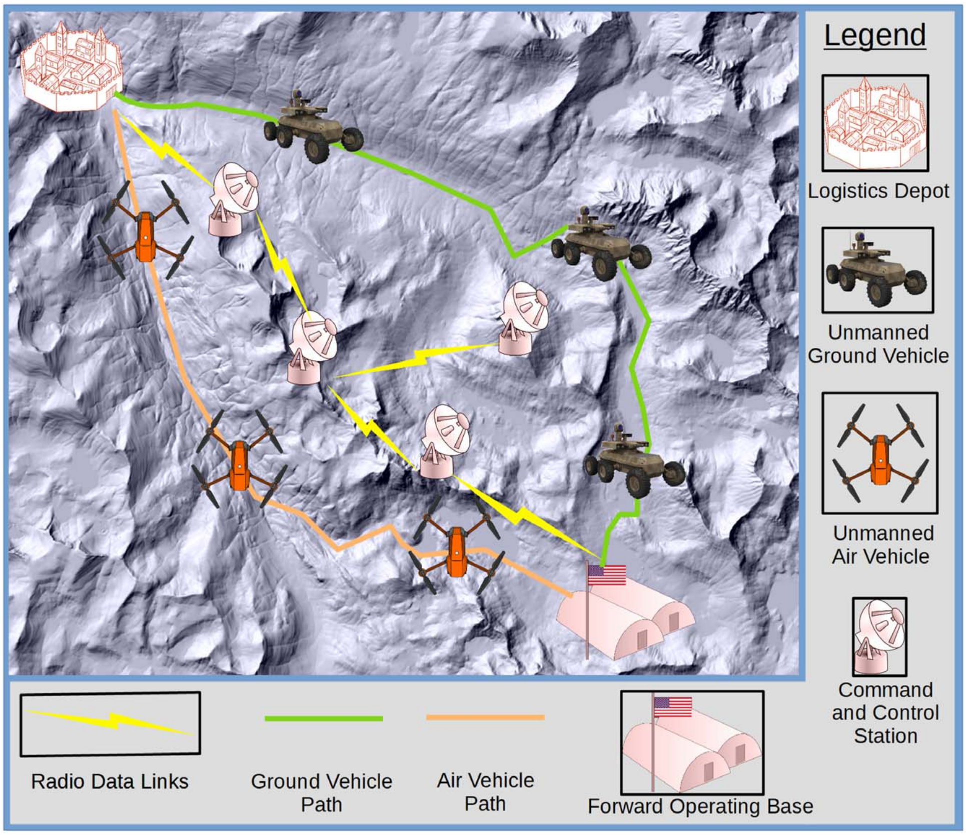

Fig. 2 High-level operational concept view of the SoS. The existing SoS member systems include unmanned ground vehicles, a logistics depot and ground control station, command and control relay stations, and the forward operating base. The new SoI UAVs are also represented on the graphic.

databases (e.g., Ref. [12]). In many cases, a nominal system design process will already have developed a functional model as part of modeling the system [3]. Figure 3 shows a high-level FFIP model for the case study SoI UAV developed using FBED.

Next, system requirements information must be collected including performance metrics and system constraints. Cost and failure probability constraints in particular are required to use this method. Other requirements and constraints will vary depending upon the specific SoI. Generally, these data will already have been developed as part of the system design process. Table 1 shows the requirements for the UAV SoI. This information will help to set goals for the trade-off studies conducted in step 6 of the method.

Finally, an analysis of the SoI's place in a larger SoS environment must be undertaken which will be used in steps 6 and 7 of the method. Questions to ask are (1) what other systems are present, (2) how important is it that each system continues to function, (3) what is the cost of having a system fail, and (4) what external event(s) may cause a system to fail. The resulting information then is recorded in terms of consequences for other systems within the SoS failing. Consequence data are shown in Table 2 for the case study. The consequence is determined by the cost distribution function, C_e, defined as the probability density of the cost of a system damaging other systems within a SoS from emitted failure flows.

Table 1 SoI requirements for a UAV used to carry cargo

Req	Requirement
1	Carry 10 kg 5 km
2	Complete round-trip transit with 99% success rate
3	Communicate with ground control station at 1.5Mbit/s TX/RX
⋮	⋮

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.

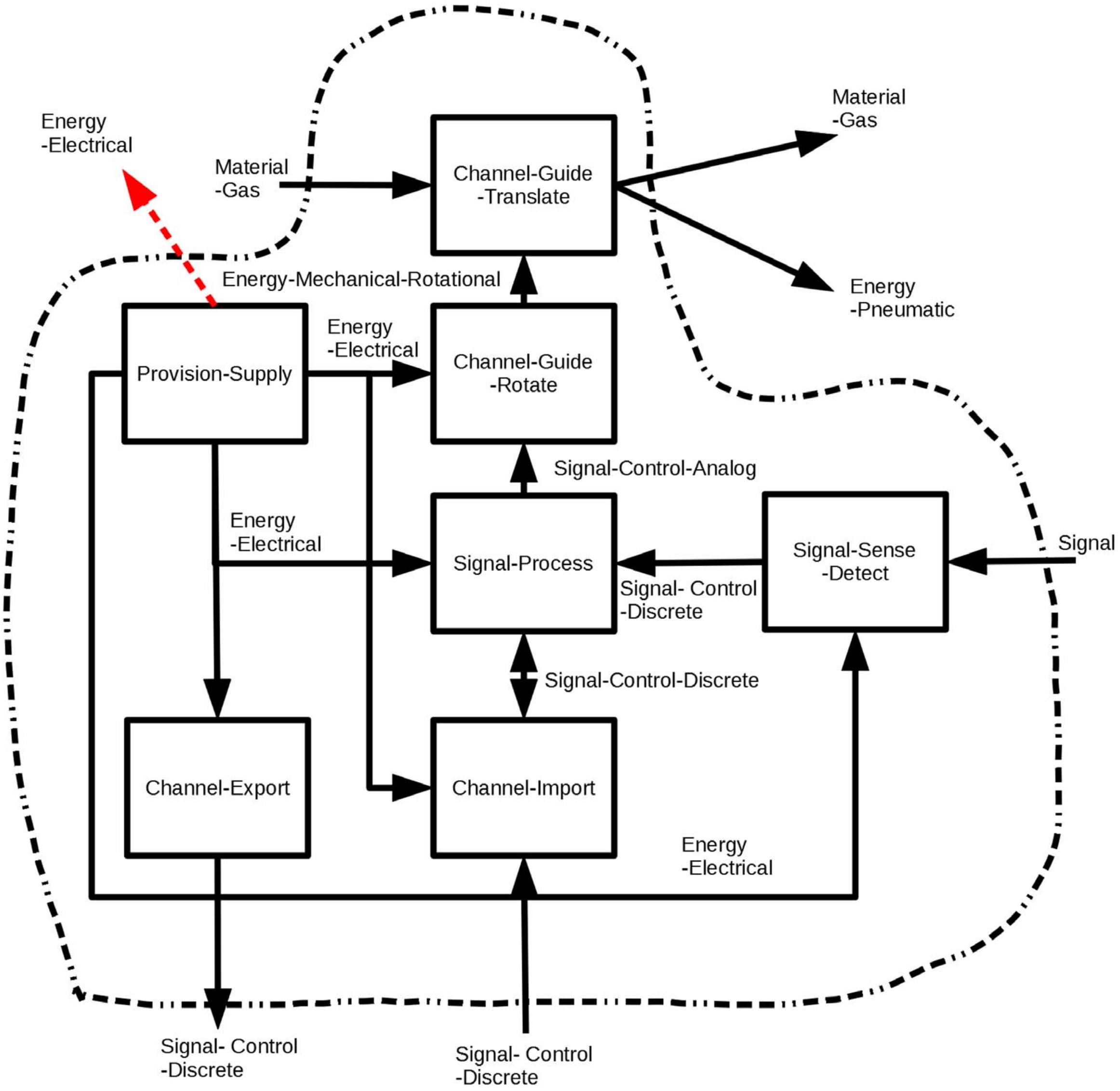

Fig. 3 An FFIP model of the SoI UAV developed using the FBED taxonomy. The dotted dashed line represents the system boundary. The dashed line represents a SSE.

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 is presented in Table 4.

3.6 Step 3: Determine Probabilities of Spurious System Emissions Exiting the Systemof 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, PO_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 an SoI to initiating events of other systems in the SoS with a probability of negative consequence (P_e) to the other systems can be produced. We propose that P_e can be combined with C_e developed in Table 2 into an emission priority (EP) which we suggest as a metric to make comparisons between SSEs similar to how the risk priority number that FMECA produces is used and the general understanding of how engineering risk is calculated (e.g., risk = probability × consequence). The calculation to produce EP for a given flow emission e is provided in Eq. (1). Table 6 provides a representative subset of EPs for the UAV SoI that are rank ordered based on the EP which indicates which SSEs have the biggest negative impact on the SoS.

$$EP_e = P_e \times C_e \tag{1}$$

Table 2 Consequence data for an mixed UAV and unmanned ground vehicle (UGV) SoS where consequence is the cost distribution function C_e for the impact of the system on the SoS

Failure flow exports from system of interest that leads to initiating events for other systems in the SoS	Consequence	C_e
Energy-electrical	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	\$5M
Material-solid-particulate	Large particulate from crashed UAVs can clog air vents and cause overheating of UGVs leading to disabled systems	\$1M
Material-control-analog	Interference with radio transceivers causes UAVs to automatically land regardless of terrain or of potential adversary presence	\$2M
⋮	⋮	⋮

Note: In this example, each C_e is a point distribution.

3.8 Step 5: Identify Spurious System Emission 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 order for the SoI to not potentially initiate failures in neighboring systems.

We recommend information on mitigation strategies including both the functional representation and the physical solution to each mitigation strategy. Additionally, information on (1) the likelihood of completely mitigating the SSE, (2) other failure flows that may be created by the mitigation strategy, and (3) other relevant failure data should be captured at this stage. Table 7 shows an example of several mitigation strategies for the SoI where P_{Me} is the probability of a mitigated SSE still occurring in spite of the mitigation strategy. New failure flow leaving system (NFFLS) represents if a new failure flow produced by a failure within the implementation of the mitigation strategy may leave the SoI system boundary as a SSE. P_{Mf} is the probability of an NFFLS leaving the SoI as a SSE. Note that P_{Mf} does not provide insight into if the new SSE can damage other systems within the SoS. M_C is the mitigation implementation cost.

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 P_e, denoted by $\mathbf{EP}_{\mathbf{Reduced}}$. Equation (2) demonstrates how to calculate $\mathbf{EP}_{\mathbf{Reduced}}$ where e is the SSE and m is the mitigation strategy.

$$\mathbf{EP}_{\mathbf{Reduced}_{(\mathbf{e,m})}} = P_{Me_{(e,m)}} \times C_{e_{(e,m)}} \tag{2}$$

An example of the matrix that is populated by Eq. (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.

$$\mathbf{EP}_{\mathbf{Reduced}_{\mathbf{e,m}}} = \begin{bmatrix} \mathrm{EP}_{\mathrm{Post-Mitigation}_{1,1}} & \mathrm{EP}_{\mathrm{Post-Mitigation}_{1,2}} & 0 \\ 0 & 0 & \mathrm{EP}_{\mathrm{Post-Mitigation}_{2,3}} \\ 0 & \mathrm{EP}_{\mathrm{Post-Mitigation}_{3,2}} & 0 \\ \mathrm{EP}_{\mathrm{Post-Mitigation}_{4,1}} & 0 & \mathrm{EP}_{\mathrm{Post-Mitigation}_{4,3}} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{3}$$

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

$$\mathbf{MP} = \overrightarrow{\mathrm{EP}}^{\mathrm{T}} \times \mathbf{EP}_{\mathbf{Reduced}} + \mathbf{M}_{\mathbf{C}} \tag{4}$$

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, m.

$$\begin{aligned} \mathrm{MRP}_m = \mathrm{rank}(\max(\mathrm{MP}_m)) \\ + \mathrm{rank}(\mathrm{mean}(\mathrm{MP}_m)) \\ + \mathrm{rank}(\mathrm{std}(\mathrm{MP}_m)) \end{aligned} \tag{5}$$

MRP as presented in Eq. (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 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, and 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_{Mf}) 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. Furthermore, 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 an SoS context in several ways. Most existing methods such as requirements management, PRA and FMECA, and other similar techniques from the systems

Table 3 An example of examining a channel-guide-rotate function to understand which potential failure flows can occur

Failure flow imports			→	Failure flow exports			
Primary	Secondary	Tertiary	→	Primary	secondary	Tertiary	Component solution(s) to function
				Material	~~Human~~		
Energy	Mechanical	Translational	→		Gas		DC motor
Energy	Electrical		→		Liquid		AC motor
Energy	Mechanical	Translational	→		Solid	Object	AC motor, DC motor
Energy	Mechanical	Translational	→			Particulate	AC motor, DC motor, pneumatic motor
						~~Composite~~	
					~~Plasma~~		
					~~Mixture~~	~~Gas–gas~~	
						~~Liquid–liquid~~	
						~~Solid–solid~~	
						~~Solid–liquid~~	
						~~Liquid–gas~~	
						~~Solid–gas~~	
						~~Solid–liquid–gas~~	
						~~Colloidal~~	
				Signal	Status	~~Auditory~~	
						~~Olfactory~~	
						~~Tactile~~	
						~~Taste~~	
Energy	Mechanical	Pneumatic	→			Visual	Pneumatic motor
Energy	Electrical		→		Control	Analog	AC motor
Energy	Electromagnetic	Solar	→	”	”	”	AC motor
						~~Discrete~~	
				Energy	~~Human~~		
					~~Acoustic~~		
					~~Biological~~		
					~~Chemical~~		
					~~Electrical~~		
Energy	Electrical		→		Electromagnetic	Optical	DC motor
						~~Solar~~	
					~~Hydraulic~~		
					~~Magnetic~~		
					~~Mechanical~~	~~Rotational~~	
						~~Translational~~	
						~~Pneumatic~~	
					~~Radioactive/nuclear~~		
Energy	Radioactive/nuclear		→		Thermal		AC, DC, pneumatic motor

Note: 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.

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 [26], it does so from the perspective of defending against the spurious system emissions rather than preventing the emissions from occurring in the first place.

Successful implementation of the method in the system architecture phase of the system design may benefit system engineers by

Table 4 Subset of failure flow paths of the UAV SoI that exit the system boundary

Flow path	Failure flow path
1	Energy-electrical → provision-supply → energy-electrical → channel-export → signal-control-discrete
2	Material-mixture-gas/solid → channel-guide-translate → material-solid
3	Provision-supply → energy-electrical → channel-guide-rotate → energy-thermal
⋮	⋮

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

Failure flows that result in system emissions	PO_e
Energy, mechanical, translational	2.2×10^{-4}/year
Material, gas	4.3×10^{-3}/year
Signal, status, visual	5.6×10^{-3}/year
Material, solid, particulate	1.9×10^{-2}/year
Material, liquid	8.3×10^{-3}/year
⋮	⋮

Table 6 Representative subset of SSE EPs of the UAV SoI

Spurious system emission	P_e	C_e	EP
Energy-mechanical-translational	5.2×10^{-4}/year	\$5M	\$2600/year
Material-solid-particulate	1.9×10^{-5}/year	\$1M	\$19/year
Material-control-analog	2.6×10^{-4}/year	\$2M	\$520/year
⋮	⋮	⋮	⋮

Note: 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 7 Sample mitigation strategies for a subset of the UAV SoI SSEs

Spurious system emission	Mitigation strategy function(s)	Physical solution(s)	P_{Me}	New failure flow(s)	NFFLS?	P_{Mf}	M_C
Energy, mechanical, translational	Control magnitude, stop, inhibit	Shielding to prevent rotor strikes	4.7×10^{-5}/year	Material, solid, object	Yes	3.5×10^{-3}/year	$300k
Signal, status, visual	Signal, process	Redundant control system to verify visual control signals before sending	4.2×10^{-5}/year	No	No	0	$1M
Material, liquid	Provision, store, contain	Catchment subsystem to retain any liquid generated by failed battery cells	5.2×10^{-5}/year	No	No	0	$500k
”	Channel, export	Long hose to direct liquid to ground	6.2×10^{-5}/year	Material, mixture, liquid–solid	Yes	3.1×10^{-5}/year	$250k
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Note: P_{Me} is the probability of a mitigated SSE still occurring. NFFLS represents if a new SSE may leave the SoI as a result of the mitigation strategy function. P_{Mf} is the probability distribution function of a new SSE leaving the system. M_C is the mitigation cost distribution function.

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, system 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 SoIs in an SoS more detectable by adversaries. We also found that some failures in subsystems such as those involved electronic warfare countermeasures on the UAV SoI 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 also to weigh the trade-offs between mitigation strategies at the functional stage.

The method can be used in parallel on many different SoIs within an SoS which allows for a comparison across all mitigation strategies for all SoIs in an SoS to be conducted 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 conducing 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 [32].

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 PO_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 ties together failure analysis of an SoS by bridging the method presented in this paper and our previous work [26]. 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 an 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 an SoI to prevent damage to other systems within an SoS. An illustrative case study of a UAV SoI being designed to enter service with an existing SoS is presented to demonstrate the method.

Acknowledgment

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

[1] Blanchard, B., and Fabrycky, W., 2006, *Systems Engineering and Analysis*, 4th ed. (International Series in Industrial and Systems), Prentice-Hall, Upper Saddle River, NJ.

[2] Kapurch, S., 2010, *NASA Systems Engineering Handbook*, DIANE Publishing Company, Hanover, MD.

[3] Crawley, E., Cameron, B., and Selva, D., 2015, *System Architecture: Strategy and Product Development for Complex Systems*. Always Learning. Pearson, Hoboken, NJ.

[4] Gold, R., 2016, “Mission Engineering,” 19th Annual NDIA Systems Engineering Conference, Springfield, VA, Oct. 24–27.

[5] Hernandez, A. S., Karimova, T., Nelson, D. H., Ng, E., Nepal, B., and Schott, E., 2017, “Mission Engineering and Analysis: Innovations in the Military Decision Making Process,” Proceedings of the American Society for Engineering Management (ASEM) 2017 International Annual Conference: Reimagining Systems Engineering and Management, Huntsville, AL, Oct. 18–21, pp. 521–530.

[6] Giles, K., and Giammarco, K., 2019, "A Mission-Based Architecture for Swarm Unmanned Systems," Syst. Eng., **22**(3), pp. 271–281.

[7] Beery, P., and Paulo, E., 2019, "Application of Model-Based Systems Engineering Concepts to Support Mission Engineering," Systems, **7**(3), p. 44.

[8] Jamshidi, M. and Sage, A. P., eds., 2008, *System of Systems Engineering: Innovations for the Twenty-first Century*, Vol. 1, Wiley, Hoboken, NJ.

[9] Sousa-Poza, A., Kovacic, S., and Keating, C., 2008, "System of Systems Engineering: An Emerging Multidiscipline," Int. J. Syst. Syst. Eng., **1**(1–2), pp. 1–17.

[10] Otto, K., and Wood, K., 2001, *Product Design: Techniques in Reverse Engineering and New Product Development*, Prentice Hall, Englewood Cliffs, NJ.

[11] van Eck, D., McAdams, D. A., and Vermaas, P. E., 2007, "Functional Decomposition in Engineering: A Survey," ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Las Vegas, NV, Sept. 4–7, American Society of Mechanical Engineers, pp. 227–236.

[12] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Res. Eng. Des., **13**(2), pp. 65–82.

[13] Stone, R. B., and Wood, K. L., 2000, "Development of a Functional Basis for Design," ASME J. Mech. Des., **122**(4), pp. 359–370.

[14] Bohm, M. R., Stone, R. B., and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," J. Comput. Inf. Sci. Eng., **5**(4), pp. 360–372.

[15] Bohm, M. R., Vucovich, J. P., and Stone, R. B., 2008, "Using a Design Repository to Drive Concept Generation," ASME J. Comput. Inf. Sci. Eng., **8**(1), p. 014502.

[16] Bohm, M. R., Stone, R. B., Simpson, T. W., and Steva, E. D., 2008, "Introduction of a Data Schema to Support a Design Repository," Comput. Aided Des., **40**(7), pp. 801–811.

[17] Van Wie, M., Bryant, C. R., Bohm, M. R., McAdams, D. A., and Stone, R. B., 2005, "A Model of Function-Based Representations," AI EDAM, **19**(2), pp. 89–111.

[18] Gilchrist, W., 1993, "Modelling Failure Modes and Effects Analysis," Int. J. Qual. Reliab. Manage., **10**(5), pp. 16–23.

[19] Stott, J. E., Britton, P. T., Ring, R. W., Hark, F., and Hatfield, G. S., 2010, "Common Cause Failure Modeling: Aerospace Versus Nuclear.", Proceedings of the 10th International Conference on Probabilistic Safety Assessment and Management, Seattle, WA, June 7–11, IAPSAM, pp. 1– 12, Paper 371.

[20] Knochenhauer, M., and Louko, P., 2004, "*Guidance for External Events Analysis*," *Probabilistic Safety Assessment and Management*, C. Spitzer, U. Schmocker, and V. N. Dang, eds., Springer, New York, pp. 1498–1503.

[21] Sorensen, J. N., Apostolakis, G. E., Kress, T. S., and Powers, D. A., 1999, "On the Role of Defense in Depth in Risk-Informed Regulation," Proceedings of PSA '99, International Topical Meeting on Probabilistic Safety Assessment, Washington, DC, Aug. 22–26, American Nuclear Society, La Grange Park, IL, pp. 408–413.

[22] Modarres, M., 2009, "Advanced Nuclear Power Plant Regulation Using Risk-Informed and Performance-Based Methods," Reliab. Eng. Syst. Saf., **94**(2), pp. 211–217.

[23] Henley, E. J., and Kumamoto, H., 1981, *Reliability Engineering and Risk Assessment*, Vol. 568, Prentice-Hall, Englewood Cliffs, NJ.

[24] Kurtoglu, T., Tumer, I. Y., and Jensen, D. C., 2010, "A Functional Failure Reasoning Methodology for Evaluation of Conceptual System Architectures," Res. Eng. Des., **21**(4), pp. 209–234.

[25] Jensen, D., Tumer, I. Y., and Kurtoglu, T., 2009, "Flow State Logic (FSL) for Analysis of Failure Propagation in Early Design," International Design Theory and Methodology Conference, IDETC/CIE, San Diego, CA, Aug. 30–Sept. 2, ASME.

[26] Van Bossuyt, D. L., O'Halloran, B. M., and Arlitt, R. M., 2019, "A Method of Identifying and Analyzing Irrational System Behavior in a System of Systems," Syst. Eng., **22**(6), pp. 519–537.

[27] Coldren, L. O., 1985, "Afghanistan in 1984: The Fifth Year of the Russo-Afghan War," Asian Surv., **25**(2), pp. 169–179.

[28] Hunsaker, L., 2015, "ARSENL Reaches Its Ultimate Goal of 50 Autonomous UAVs in Flight," https://nps.edu/-/arsenl-reaches-its-ultimate-goal-of-50-autonomous-uavs-in-flight

[29] Kurtoglu, T., and Tumer, I. Y., 2008, "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems," ASME J. Mech. Des., **130**(5), p. 051401.

[30] O'Halloran, B. M., 2013, "A Framework to Model Reliability and Failures in Complex Systems During the Early Engineering Design Process," Ph.D. thesis, School of Mechanical, Industrial, and Manufacturing Engineering, Corvallis, OR.

[31] Anderson, D., 2018, "A Design Process for Design Automation," Ph.D. thesis, Singapore University of Technology and Design, Singapore.

[32] Westinghouse Electric Company, LLC, 2004, *AP1000 Probabilistic Risk Assessment*, revision 7th ed., Westinghouse Electric Company, LLC, Pittsburgh, PA.

[33] Van Bossuyt, D. L., and Arlitt, R., 2019, "Toward a Functional Failure Analysis Method of Identifying and Mitigating Spurious System Emissions in a System of Systems," International Design Theory and Methodology Conference, IDETC/CIE, ASME, Anaheim, CA.