2018-08-26

Toward a generative human-in-the-loop approach for conceptual design exploration using flow failure frequency in functional models

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

ASME

Arlitt, Ryan M., and Douglas L. Van Bossuyt. "Toward a Generative Human-in-the-Loop Approach for Conceptual Design Exploration Using Flow Failure Frequency in Functional Models." ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, 2018.

http://hdl.handle.net/10945/62530

Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC/CIE 2018
August 26-29, 2018, Quebec City, Quebec, Canada

DETC2018-85490

TOWARD A GENERATIVE HUMAN-IN-THE-LOOP APPROACH FOR CONCEPTUAL DESIGN EXPLORATION USING FLOW FAILURE FREQUENCY IN FUNCTIONAL MODELS

Ryan M. Arlitt
SUTD-MIT International Design Centre
Singapore University of Technology and Design
Singapore, 487372
Email: arlitt.ryan@gmail.com

Douglas L. Van Bossuyt*
Department of Systems Engineering
Naval Postgraduate School
Monterey, California, 93940
Email: douglas.vanbossuyt@nps.edu

ABSTRACT

A challenge systems engineers and designers face when applying system failure risk assessment methods such as Probabilistic Risk Assessment (PRA) during conceptual design is their reliance on historical data and behavioral models. This paper presents a framework for exploring a space of functional models using graph rewriting rules and a qualitative failure simulation framework that presents information in an intuitive manner for human-in-the-loop decision-making and human-guided design. An example is presented wherein a functional model of an electrical power system is iteratively perturbed to generate alternatives. The alternative functional models suggest different approaches to mitigating an emergent system failure vulnerability in the electrical power system's the heat extraction capability. A preferred functional model configuration that has a desirable failure flow distribution can then be identified. The method presented here helps systems designers to better understand where failures propagate through systems and guides modification of systems functional models to adjust the way in which systems fail to have more desirable characteristics.

INTRODUCTION

The design, manufacture, and deployment of complex systems requires extensive investment of personnel, resources, time, and money to produce systems that meet requirements [1, 2]. Schedule and cost overruns are common on large systems such as aircraft, spacecraft, power plants, ships, and other systems [3]. A significant percentage of schedule and cost overruns, and reduced systems capabilities as compared to original requirement documents can be traced back to architectural decisions made during the conceptual phase of system design [4]. Architectural decisions that are made with incorrect or missing information, or that are made with high degrees of uncertainty in the data can lead to incorrect decisions being made that then leads to cost increases and schedule slips [5]. As a result, it is important that architectural decisions are made with good, complete information to increase the likelihood of systems being delivered on time, on budget, and meeting requirements.

Of particular interest to this research is how potential system failures are assessed and acted upon during the conceptual phase of system design. Common techniques of identifying failure risks and then mitigating them such as Failure Mode and Effects Analysis (FMEA) [6] and Probabilistic Risk Assessment (PRA) [7, 8] can miss emergent system behaviors and, while some information is provided to designers to aid in decision-making, little guidance is given on specific flow impacts due to failure events. Extensive work has been done to understand failure paths from a component and/or functional basis [9–15] but comparatively little effort has been expended in looking at flows of material, energy, and data through systems, and how their disruption or failure can impact overall system failure.

*Address all correspondence to this author.

Specific Contributions

This paper contributes a method to identify functional models that have a desirable distribution of flow failure events across a large space of failure scenarios. The method identifies flows that are most often associated with failure events, and automatically explores a variety of potential alternative functional models to identify models that have lower flow failure concentrations. Visualizations of these alternatives are presented to the user, allowing quick iteration of functional architectures in the context of limited embodiment information. This contribution arises from the combination of a generative approach for building functional models and an evaluation approach that qualitatively simulates the failure performance of each functional model.

RELATED WORK

Large, complex systems such as power plants, mass transit systems, aircraft, and ships are designed, manufactured, and deployed using a design process that begins with early ideation and conceptual design studies, progresses through subsystem and component design, verification and validation, manufacturing, and finally into deployment and maintenance [1, 16]. System architecture decisions made in the conceptual phase of the design process have a significant impact on all other activities that follow [2]. An incorrect architecture decision can cause significant cost and schedule overruns, or may lead to a system that does not meet all performance requirements [17]. Therefore it is important to make correct and timely architecture and design decisions during the conceptual phase of design.

Within the conceptual phase of design, there are several distinct steps including 1) ideation, 2) early system architecture studies, 3) and system modeling and trade studies [18]. During the last step of conceptual design, high level and black box models produced in the previous step are refined into subsystem, functional, and component models [19]. A variety of modeling techniques and methods are commonly used to help make informed decisions based on trade studies such as functional models; risk, reliability, failure, availability, and robustness models; and other related modeling and assessment methods [6, 20–26]. These design decisions directly impact later subsystem and component design, and if made incorrectly due to a lack of information or a misunderstanding of the fundamental nature of the system's design, significant rework and redesign costs can be incurred [27, 28]. Timely information on which to base design decisions is critical for the delivery of an on-time and on-budget system that performs as intended [21, 26].

A number of modeling paradigms exist to model systems during conceptual design [16, 29]. Of particular interest to this research, functional and flow methods of modeling systems during the conceptual phase of design can be used to help free engineers from component considerations and allows more creativity with finding new system design solutions [19]. While there are many different functional and flow taxonomies and grammars [30–60], this research uses the Functional Basis for Engineering Design taxonomy [19] (herein referred to as FB) to represent functions and flows within systems. The FB taxonomy abstracts functions and flows from the physical components and transported material, energy, or data that they represent. Of particular value is the potential for simulating abstract models constructed using FB, which is possible so long as that model has (1) topological consistency and (2) conservation of material and energy [61].

Grammar rules have been developed to aid designers and automated design tools in identifying conceptual design configurations that are likely to be realizable in physical component design [62–64]. Helms et. al. [65] prescribe a general approach for synthesis of product architectures using the Function Behavior Structure framework [66]. This model supports synthesis of component architectures from a functional model, and makes explicit the need for simulation and evaluation to close the synthesis loop. Similarly, Kurtoglu and Campbell developed grammar rules to convert functional models into component-level configuration flow graphs [62]. More specific to the domain of functional architectures, Sridharan and Campbell [63] generated 69 grammar rules from 30 products located in the Design Repository [67] to create a framework for generating functional models.

It should be noted that there is significant heterogeneity of modeling languages in which grammars are implemented. For this research, the selection of the FB modeling taxonomy was intentional. Not only is FB is a functional description with high generality, but there exist several computational tools for evaluating FB models which is required to close the computational design synthesis loop. The recent development of several simulation approaches to evaluating failures in functional models [9, 10, 68] enables a new generative design loop for examining reliability of functional architectures.

Evaluation methods and decision support tools have been developed to aid systems designers to make conceptual architectural decisions. These methods and tools can be broadly categorized as: simulation-function, simulation-component, expert knowledge and experience, and historical function/component. A high-level review of tools useful for failure analysis and related analysis techniques that fall within the four categories listed above is provided below.

Within the simulation-function category, the Function Failure Identification and Propagation (FFIP) method and related Flow State Logic method identify potential failure flow pathways through a functional model [9, 10]. The Inherent Behavioral in Functional Models (IBFM) framework extends FFIP to include the ability to generate multiple functional models to drive toward a solution that can balance the cost and risk of a system, and a pseudo time step [15, 68, 69]. A number of other risk and failure analysis tools have been developed from FFIP including the Uncoupled Failure Flow State Reasoner [11, 70], a method of building prognostic systems in response to failure modeling [12],

and other related methods and tools [13, 14, 71–73]. Several tools for ontology-driven metamodeling and early conceptual design down-selection were produced as part of the Defense Advanced Research Program Agency (DARPA) Adaptive Vehicle Make project [74–76]. While these methods are useful for identifying and understanding failure sources within a system, they generally lack the ability to identify specific flow paths that are more often implicated in potential system failure events.

Several simulation-component methods exist including the Reliability Block Diagram method [77] widely used in industry and a method developed by O'Halloran et. al. that simulates component performance at varying levels of fidelity based on model fidelity [78]. While these types of methods are useful for understanding reliability of a system and O'Halloran's method is useful for simulating expected system performance, both rely upon historical data. This limits the ability of this class of method to identify emergent system behaviors. Further, little guidance is provided by the results of these methods to identify specific flows within the system that are at higher risk of failure.

Expert knowledge and experience plays a large role in several methods that are important to industry. Failure Mode and Effects Analysis (FMEA) [6] and the related Failure Modes, Effects, and Criticality Analysis (FMECA) [79] use expert knowledge and system experience to identify and understand potential failure scenarios within a proposed system. Expert elicitation is often used in producing fever charts and other graphical representations of risk within a system [80]. Expert knowledge and experience methods in general do not adequately capture potential emergent system behaviors – especially complex failure events.

Several methods have examined the link between historical performance of functions and components, and their expected behavior in new systems. The Function Failure Design Method [81] provides a matrix-based approach to linking a function to potential component solution failure modes. The Risk in Early Design method [82] connects historical risk information to ongoing design efforts and provides a fever chart view for ease of understanding by novice risk analysts. While these methods do well at identifying historical failure information on a functional level, they do not adequately uncover emergent system behaviors.

Many other methods of failure and risk analysis exist that can help system designers to make risk and failure-informed architectural decisions during conceptual design. Probabilistic Risk Assessment (PRA) combines fault tree analysis and event tree analysis [7, 8] with an analysis of potential initiating events that can lead to failure [83]. The nuclear industry heavily uses PRA to identify potential emergent system behaviors and ensure safety of nuclear power plants [84]. A popular method of identifying potential failures uses Markov chains that are built to model state transitions in a system where probabilities of state transitions are known or can be assumed. The Markov chains are then randomly walked using Monte Carlo sampling to determine the probability of being in each state [85–88]. The Markov chain Monte Carlo sampling approach is especially applicable in the PRA context (e.g., [89]) because of its relative efficiency of approximating Bayesian posteriors. The method presented in this paper differs in that the failure simulation is deterministic for a large set of different state spaces. Repetition of this simulation on single functional model occurs only by sampling from different combinations of initiating failure events.

Given that the method presented in this paper is intended to facilitate exploration over a population of graphs, some heuristics are necessary to combat combinatorial explosion. Subsampling a representative space achieves this goal, but requires a method to calculate graph similarity prior to evaluation. Graph similarity algorithms can be classified as edit distance, feature extraction, and iterative [90]. Feature extraction is selected here due to simplicity of implementation, speed of evaluation, and existing evidence for a correlation between graph-level features (e.g., diameter and node degree) and system-level reliability (e.g., [91, 92]). Additionally, the bag-of-functions feature approach has been successfully used to measure similarity between functional models [93, 94].

In the area of software debugging with model checking, one common strategy is to validate an abstraction of values, states, and transitions [95]. This type of model checking is in many ways analogous to the approach presented in this paper. While both execute abstractions of the system to search for issues, the method presented in this paper combines a formalism for abstracting and simulating complex systems with a means to search the design space.

In summary, the conceptual phase of the systems engineering design process provides systems designers with an opportunity to make significant architectural decisions that can drastically impact the outcome of the design process and the performance of the system. A variety of tools and methods are available to help support engineers in making informed decisions during the conceptual phase. Many such tools and methods rely on functional modeling techniques and a number of methods exist to analyze failure within this context. However, none of the existing methods surveyed is able to directly assess failures from a flow perspective over a space of related functional models and use that information to help make architectural decisions.

METHODOLOGY

The method presented below is specifically intended for use during the conceptual phase of design when architectural decisions are being made and the design has not been finalized. The method's inputs include a single functional model from the user, a library of IBFM simulation components, and (optionally) a specification of each IBFM state's probability to serve as an initiating failure event. The method's output is a visualization

of several alternative functional models and the vulnerability of each flow therein to failures.

Develop Functional Model

The first step is for the designer to create a functional model for the system of interest. This model will be used as a seed to begin the process of analyzing failure flows.

Develop IBFM Simulation

Given a seed model, an IBFM simulation is prepared [15]. This simulation must capture the designer's abstract knowledge about the system. This includes the following:

1. Functions, including the operational modes and mode transition conditions applicable to each
2. Flows
3. Modes and the associated flow behaviors associated with each
4. Conditions and the flow state behavior associated with triggering them

Given these elements, IBFM enables qualitative simulation of the functional model. More details about IBFM can be found in [15].

Specify Probabilities

The method presented in this paper can be performed with either internal initiating events caused by failed modes of functions within the system or by external events that occur outside of the system boundary and propagate into the system as failure flows. The case study below uses internal initiating events as a demonstration.

For internal initiating events, each failed mode of each function is treated as equally likely to occur as the default approach. However, if a probability of occurrence is known for an internal initiating event, then that probability is used instead. With external initiating events, the authors recommend only using probabilities that are grounded in reality and are realistic. When not using probabilities specific to a function's failed state, the frequency of occurrence of failure flows associated with each flow can be ascertained on a normalized basis. With specific probabilities available, these frequencies can be weighted according to their expected likelihood.

Automatically Generate Similar Functional Models

Using the designer's functional model as a seed, automatically generate locally similar functional models according to a limited set of graph grammar rules (e.g., Table 1). These grammars perturb the model by removing functions and by reinserting functions that are already present – new functionality is not added. The result is a means to generate different functional architectures while preserving the gist of the design intent. These grammars must be capable of both adding and removing elements, and must conform to topological consistency and conservation rules for FB.

Validate Automatically Generated Functional Models

For a functional model to be simulatable, two main requirements must be met: (1) conservation of mass and energy, and (2) each function's inputs and outputs must be consistent with established semantics [61]. This can be done at generation time through careful construction of grammars, or naively by iteratively discarding non-compliant models and then generating replacements. Active model checking requires software that captures the two requirements – like that developed in [61].

Run Simulation on Each Functional Model

Next, each model in the population is simulated using IBFM. By default, an IBFM experiment runs simulations using every possible failure state as an initiating event. Scenarios are then run for all paired combinations of simultaneous initiating events, and the number of simultaneous events increases until a prescribed cutoff. The failure rate of each flow in the model is captured as described in Algorithm 1.

Algorithm 1 Functional Model Population Simulation Process

```
for each model M in the population do
    Initialize a zero vector of failure counts F to capture the
    failure frequencies of all flow edges in the model
    Generate a list of scenarios S containing initiating events
    and their corresponding nodes
    for each specified scenario S do
        Simulate M under conditions of S until the model
        reaches steady state
        for each failed edge in the resulting M do
            Increment its total failure count in F, normalized by
            the probability of the initiating event
        end for
    end for
    Take max(F) to describe this model's resiliency
end for
```

Depending on the available computing power, this simulation can be repeated with valid combinations of multiple initiating events. While here it is recommended to characterize each model according to its most vulnerable edge $max(F)$, other performance measures can be used (e.g., the mean and variance of the edge failure frequency distribution).

Iterate Best Performing Models

Iteration consists of two steps, (1) selecting a parent population and (2) generating a child population.

A diverse parent population of models is sampled from this local space using roulette wheel selection (with replacement) and a performance measure that linearly combines resiliency R (defined as the ability of the system to continue to function in spite of failure events occurring) and uniqueness U (as proposed in Equation 1). A model's resiliency R is normalized to the maximum resiliency in the population $R_{population\_max}$. A model's uniqueness U can be quantified by applying a clustering algorithm such as DBSCAN [96], and then taking the inverse of the number of total models in that model's cluster. A full pairwise distance matrix between models is needed to support this clustering, and can be generated from the graph feature representation using cosine distance. A weighting factor k between 0 and 1 captures preference for resiliency versus uniqueness.

$$p_{selection} = k\frac{R}{R_{population\_max}} + (1-k)U \tag{1}$$

Next, a child population is generated by applying one randomly selected grammar rule to each parent in a randomly selected location. If a branching factor greater than 1 is applied, the process closely resembles breadth first tree search. If so, pruning the child population back to the initial population size after simulation mitigates combinatorial explosion of IBFM simulations. This process is visualized in Figure 2.

Stop Iteration After Performance Metrics Have Been Met

The steps of generating models, simulating their performance, and iterating are repeated until stopping criteria are met.

Two parameters capture the stopping criteria: The first dictates an acceptable level of uniqueness U specified by the user. The second dictates a performance threshold (in this case model resiliency R is quantified by the model's most vulnerable edge). When there exists a set of N models (where N is user-specified) in the most recent generation where all N models exceed the performance threshold and the uniqueness threshold, the process stops. Given that the population size is held constant, it is feasible to quantify the uniqueness of each model via clustering on the full pairwise comparison matrix using vector space similarity measures (e.g., cosine similarity) and the child's lineage. An alternative approach for large populations of constant size halts the search when the explained variance ratio of the principal component analysis of the data set's feature representation dips below a given threshold.

Assess Final Population of Functional Models

After the stopping criteria are met, a subset of models is selected from the full history of generated models. These models are selected to possess (1) high or low resiliency as desired and (2) high uniqueness with respect to each other. The rates at which flows on these models failed are indicated by both thickness and color of the line segments. The functionality to show both good and bad examples is motivated by conceptual design exploration tools like MEMIC [97], which provides creative stimulus by showing both highly common and highly uncommon component configurations to match a given functional model. Given this stimulus, the designer can assesses which topology to pursue and iterate upon, or draw inspiration to make tweaks to the functional model.

Any number of methods can be used for determining uniqueness U, though all but the most naive will rely on some means of clustering the final population. This may include straightforward clustering (e.g., k-means), projection of the bag-of-features representation into lower dimensions (e.g., Principle Component Analysis), or sampling from far-apart sections of the search tree according to each model's lineage.

CASE STUDY

This section contains an illustrative case study to demonstrate the workings of the method presented above. It should be noted that the example, while similar to a real, physically embodied system, has been intentionally fictionalized. The results of the case study are illustrative of the method's capabilities but cannot be taken as evidence of how to design the specific system presented below. No real world design decisions should be made based upon this case study.

The following case study demonstrates the mechanism of the method on a simplified functional model of the ADAPT electrical power system testbed [98]. Various model descriptions of this system have been used in prior work to demonstrate failure simulation in conceptual design for FFIP [9] and IBFM [15]. In general, the model consists of a battery, an inverter, and three loads – a fan, a pump, and an indicator light. The model also contains a switch and several breakers. The functionality of this system – which is used as a the seed model – is captured in Figure 1. The remainder of this section will address the question, "in what ways might we redesign the functional architecture of this system to improve system reliability?"

For this example, the IBFM simulation is specified as in [15], and failure mode probabilities are assumed to be equal – analogous to a non-informative prior.

After specifying the seed model to define the local search space, alternatives are iteratively generated. To facilitate this example, a simple set of grammar rules is shown in Table 1. A much more comprehensive and data-driven graph rewriting language for functional models of electromechanical products was

presented in [63]. Figure 1 shows an application of the rule "Add Parallel Subgraph" between two randomly selected edges, indicated by the dashed lines. The backbone of the inserted subgraph is shown via the same dashed lines. Additional nodes and edges are added to this new subgraph until the resulting model adheres to conservation of mass and energy. These additional components are indicated with long dashed lines.

This process is repeated to generate a population of randomly perturbed models in the local design space. Next, each model in the population is simulated using IBFM, and a score is calculated for the performance of each model. Snippets of two failure heat maps for two generated concepts are shown in Figure 3 and Figure 4. These snippets capture the flows with the highest failure rate in each model. While the model in Figure 3 would be characterized by its highest flow failure rate of 50, the model in Figure 4 would be quantified according to its (comparatively better) worst-case flow failure rate of 35.

Next, candidates from the current population are selected for iteration according to performance and uniqueness, as illustrated in Figure 2. While Figure 3 has poor performance, it may still have a high probability of selection if it is extremely different from the rest of the current population. After selection, the next generation is iteratively resampled and created until the stopping criteria are met.

Ultimately, a series of varied heat maps as shown in Figure 3 and Figure 4 are presented to the user. Based on the model in Figure 3, a user may realize that they need to pursue alternative functions for cooling the inverter function, while the model in Figure 4 may persuade the user to investigate adding parallel cooling functionality.

DISCUSSION

The method presented in this paper contains several benefits for practitioners as well as a few open questions on the philosophy of failure events. This section discusses the benefits and open questions of the method.

A significant benefit of the method is the ability for systems engineers to identify functional models that conform to desired flow failure concentration levels. The systems engineer can drive model iteration toward either a highly concentrated flow failure paradigm or a distributed flow failure paradigm. While the case study above demonstrates evolving a model toward a solution that distributes failure flow concentrations across the model by adding in redundancy, specific system design considerations may warrant concentrating failed flows into a few specific flows. Concentrating failure flow into a few flows may be beneficial, for instance, if systems engineers are including sacrificial subsystems [72]. In other situations, it may be beneficial to spread out failure flows across several redundant subsystems [99].

No other method that the authors are aware of provides practitioners with the ability to easily understand what flow paths failures preferentially follow as the model changes. As compared to standard IBFM, this generative method provides insights into how the distribution of emergent failures changes with subtle shifts in functional architecture. Additionally, most other function-and-flow-based methods of failure and risk analysis used during the conceptual phase of system design are focused on failure of functions. Examining the flows rather than the functions can provide new insights into which flows are the most likely to be implicated in failure events. This in turn can lead to systematic design efforts to mitigate those specific failure flows.

A benefit of the heat mapping of failure flow concentrations is that emergent failure flow behaviors that otherwise would be missed can be examined by systems engineers. This may provide new insights on emergent system behavior that otherwise would not be available. Emergent system behavior is a significant concern in complex systems and has been implicated in several past noteworthy failures [100–102].

It should be noted that this is a stochastic design space search method with a loose definition of optimality. Because the goal of this method is to facilitate human-in-the-loop exploration of system concepts, Pareto optimality (as a function of performance and uniqueness) is useful only as an approximation. Uniqueness in particular depends on contextual factors including the designer's preferences and the other models in the population. Further, designers should be aware of the limitations of Arrow's Theorem with respect to multi-variable optimization, especially with human-guided preferences [103].

One open area of research on the method presented above is what happens in the case where two models are simulated where one has no redundancies and the other has many parallel redundancies. IBFM currently does not heavily penalize the cost of adding new nodes. It may be desirable to adjust the penalty function parameters for adding redundancies to a system model to assist in the trade-off between the costs associated with adding redundancy and the benefits of added redundancy to mitigate potential failures. However, systems engineers must consider if parallel flow redundancy provides true benefit in stopping a failure flow before the flow leads to system failure, or if redundant flows merely provide alternative pathways to system failure as in the case of a drop in electrical voltage propagating through redundant power feeds in a data center. In the data center's case, had the energy flows been truly independent and redundant, a failure flow caused by a voltage drop on one of the power lines coming into the facility likely would not have impacted the other redundant power lines and electrical distribution systems in the facility.

An area of future work is to combine the concept of "cut sets" derived from PRA and used in some FFIP-based methods with the vulnerability of each type of flow, redundant subsystems, and comparing different models with global metrics (e.g.: ratio of failed flows per model). Further refining the IBFM's

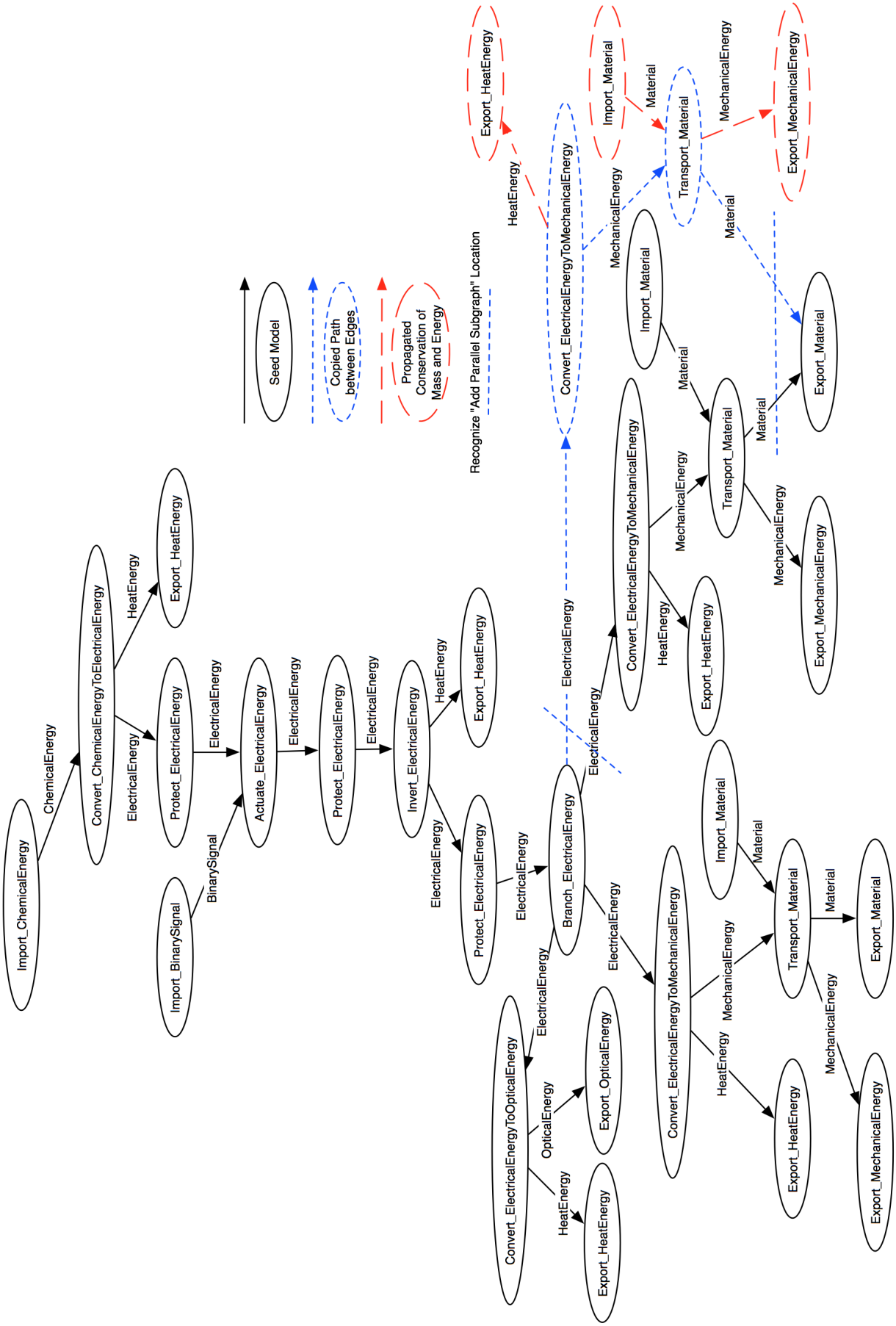

FIGURE 1. Functional Model of EPS System

TABLE 1. Naive Generative Grammar Language

Rule	Recognize	Apply
Add Parallel Path	Any two edges on the graph with a valid connecting path	Add a parallel copy of the shortest path between those edges
Add Parallel Subgraph	Any two edges on the graph with a valid connecting path	Perform "Add Parallel Path" for all paths in between those edges. Propagate copy forward and backward to satisfy conservation of mass and energy.
Add Series	Any function	Insert a copy of function in series connected by function's own flow type
Remove Node	Any function	Remove that function and connected flows. Repeat on nodes that fail a validation check until model is valid or empty.

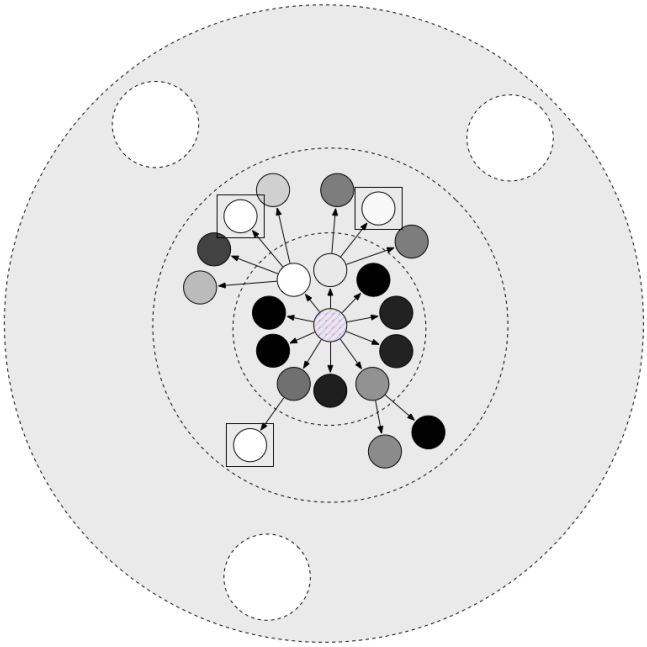

FIGURE 2. Visualization of roulette wheel sampling with branching factor of 1. Generated models expand outward into the search space toward local regions that are potentially interesting (as opposed to optimal). Higher fitness is represented as light, lower fitness is dark. When the search concludes, results are selected for presentation to the user with respect to performance and global uniqueness.

method of optimization within the context of the method presented in this paper is expected to be a useful area of further research.

While many PRA methods are by definition concerned with both the likelihood and consequence of failures, the approach in the paper addresses only likelihood. Because of the high level of abstraction of functional models, and the necessity of using contextual information to assess the consequences of a failure, evaluating failure severity is purposely left to the user. The chal-

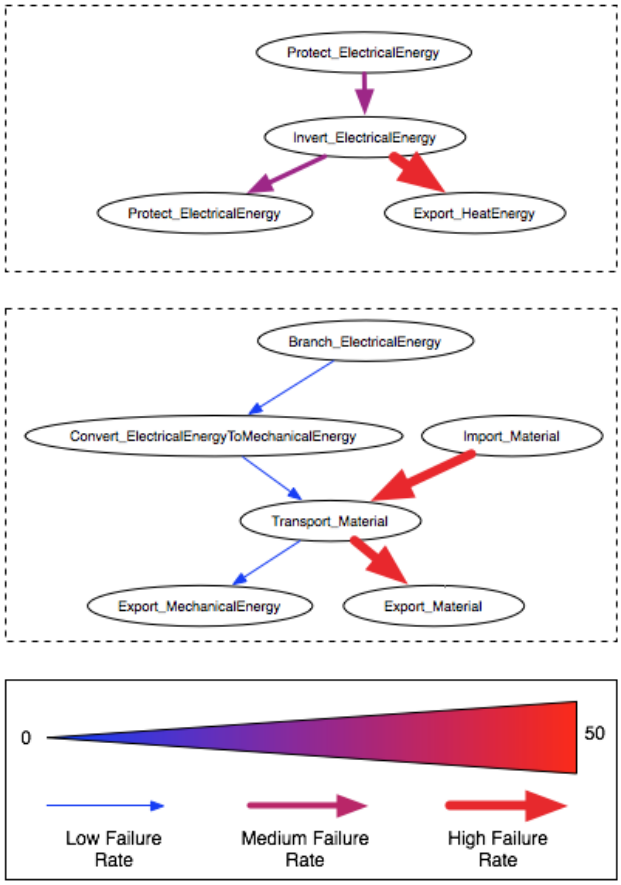

FIGURE 3. A snippet heat map of a model with poor performance. The fan module fails in many scenarios, indicated as a high failure rate in the flows related to cooling the inverter. In some cases the failure propagates to the flows related to the inverter, which increases the failure rate of those flows.

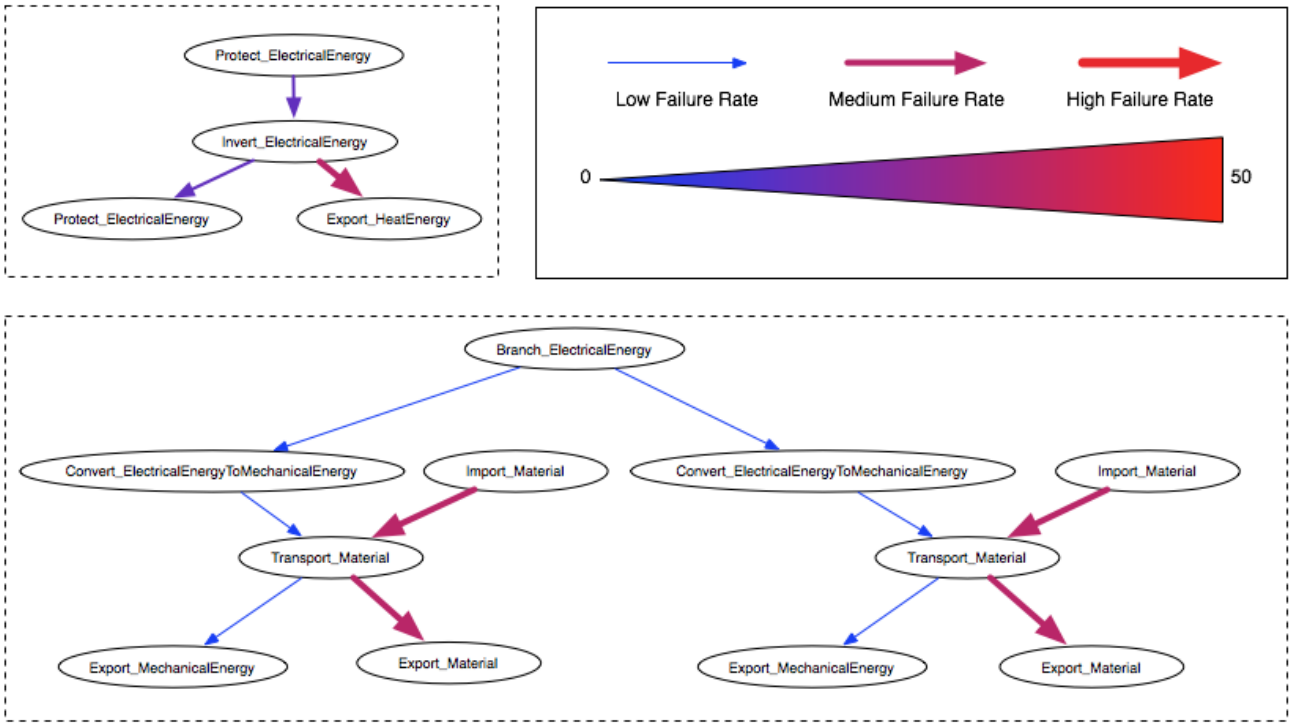

FIGURE 4. A snippet heat map of a model with medium performance. In this case grammar rules have added an additional subgraph for exporting material, which led to a reduced rate of failure in the associated flows.

lenge of capturing context and failure consequences is deferred to future work.

CONCLUSION

The framework presented in this paper represents a way to generatively explore a space of functional models, assess their vulnerability to failure, and present a designer with a variety of alternative options. The approach is human-in-the-loop; the designer must interpret the results according to the specific context of the problem. Given a library of IBFM models and a graph rewriting language for perturbing functional models, this approach enables a designer to make quick risk-of-failure-informed-decisions about functional architectures. These decisions are founded not on only experience or historical data, but on (1) qualitative simulation of potential failure propagation and (2) a set of solutions automatically generated to mitigate those failures. This allows systems designers to make large system architectural decisions very early in the conceptual design process where the cost of making decisions and significantly changing the design is relatively inexpensive both in cost and in schedule time.

ACKNOWLEDGMENT

This research is partially supported by the Naval Postgraduate School (NPS) and the Singapore University of Technology and Design (SUTD). 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] Walden, D. D., Roedler, G. J., Forsberg, K., Hamelin, R. D., and Shortell, T. M., 2015. *Systems engineering handbook: A guide for system life cycle processes and activities*. John Wiley & Sons.

[2] Ullman, D. G., 2015. *The mechanical design process*. McGraw-Hill Science/Engineering/Math.

[3] Browning, T. R., and Eppinger, S. D., 2002. "Modeling impacts of process architecture on cost and schedule risk in product development". *IEEE transactions on engineering management,* ***49***(4), pp. 428–442.

[4] Browning, T. R., 1998. "Sources of schedule risk in complex system development". In Proceedings of the Eigth Annual International Symposium of INCOSE.

[5] Wang, J. X., and Roush, M. L., 2000. *What every en-*

gineer should know about risk engineering and management. CRC Press.

[6] Stamatis, D. H., 2003. *Failure mode and effect analysis: FMEA from theory to execution*. ASQ Quality Press.

[7] Ericson, C., 2005. "Event tree analysis". *Hazard Analysis Techniques for System Safety*, pp. 223–234.

[8] Ericson, C. A., 2005. "Fault tree analysis". *Hazard analysis techniques for system safety*, pp. 183–221.

[9] Kurtoglu, T., Tumer, I. Y., and Jensen, D. C., 2010. "A functional failure reasoning methodology for evaluation of conceptual system architectures". *Research in Engineering Design,* ***21***(4), pp. 209–234.

[10] Jensen, D., Tumer, I. Y., and Kurtoglu, T., 2009. "Flow state logic (fsl) for analysis of failure propagation in early design". In ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 1033–1043.

[11] O'Halloran, B. M., Papakonstantinou, N., and Van Bossuyt, D. L., 2015. "Modeling of function failure propagation across uncoupled systems". In Reliability and Maintainability Symposium (RAMS), 2015 Annual, IEEE, pp. 1–6.

[12] L'her, G., Van Bossuyt, D. L., and O'Halloran, B. M., 2017. "Prognostic systems representation in a function-based bayesian model during engineering design". *International Journal of Prognostics and Health Management,* ***8***(2), p. 23.

[13] O'Halloran, B. M., Papakonstantinou, N., and Van Bossuyt, D. L., 2016. "Cable routing modeling in early system design to prevent cable failure propagation events". In Reliability and Maintainability Symposium (RAMS), 2016 Annual, IEEE, pp. 1–6.

[14] Dempere, J., Papakonstantinou, N., O'Halloran, B., and Van Bossuyt, D. L., 2017. "Risk modeling of variable probability external initiating events". In Reliability and Maintainability Symposium (RAMS), 2017 Annual, IEEE, pp. 1–9.

[15] McIntire, M. G., Keshavarzi, E., Tumer, I. Y., and Hoyle, C., 2016. "Functional models with inherent behavior: Towards a framework for safety analysis early in the design of complex systems". In ASME 2016 International Mechanical Engineering Congress and Exposition, American Society of Mechanical Engineers, pp. V011T15A035–V011T15A035.

[16] Haskins, C., Forsberg, K., Krueger, M., Walden, D., and Hamelin, D., 2006. *Systems engineering handbook*. INCOSE.

[17] Eppinger, S. D., 1991. "Model-based approaches to managing concurrent engineering". *Journal of Engineering Design,* ***2***(4), pp. 283–290.

[18] Otto, K., and Wood, K., 2001. "Product design: techniques in reverse engineering and new product design". *Prentice-Hall*.

[19] Stone, R. B., and Wood, K. L., 2000. "Development of a functional basis for design". *Journal of Mechanical design,* ***122***(4), pp. 359–370.

[20] Sage, A. P., and Rouse, W. B., 2009. *Handbook of systems engineering and management*. John Wiley & Sons.

[21] Kapurch, S. J., 2010. *NASA systems engineering handbook*. Diane Publishing.

[22] Cornford, S. L., Feather, M. S., and Hicks, K. A., 2001. "Ddp-a tool for life-cycle risk management". In Aerospace Conference, 2001, IEEE Proceedings., Vol. 1, IEEE, pp. 1–441.

[23] Stamatelatos, M., Dezfuli, H., Apostolakis, G., Everline, C., Guarro, S., Mathias, D., Mosleh, A., Paulos, T., Riha, D., Smith, C., et al., 2011. Probabilistic risk assessment procedures guide for nasa managers and practitioners. Tech. rep., NASA.

[24] Otto, K. N., and Antonsson, E. K., 1991. "Trade-off strategies in engineering design". *Research in Engineering Design,* ***3***(2), pp. 87–103.

[25] Estefan, J. A., et al., 2007. "Survey of model-based systems engineering (mbse) methodologies". *Incose MBSE Focus Group,* ***25***(8), pp. 1–12.

[26] Wertz, J. R., Everett, D. F., and Puschell, J. J., 2011. *Space mission engineering: the new SMAD*. Microcosm Press.

[27] Sen, P., and Yang, J.-B., 2012. *Multiple criteria decision support in engineering design*. Springer Science & Business Media.

[28] Blanchard, B. S., and Fabrycky, W. J., 1998. *Systems engineering and analysis*. Prentice Hall, Inc., Upper Saddle River, NJ (United States).

[29] Friedenthal, S., Moore, A., and Steiner, R., 2014. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann.

[30] Erden, M. S., Komoto, H., van Beek, T. J., D'Amelio, V., Echavarria, E., and Tomiyama, T., 2008. "A review of function modeling: Approaches and applications". *Ai Edam,* ***22***(2), pp. 147–169.

[31] Houkes, W., and Vermaas, P. E., 2010. *Technical functions: On the use and design of artefacts*, Vol. 1. Springer Science & Business Media.

[32] Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N. V., and Wood, K. L., 2011. "Computer-based design synthesis research: an overview". *Journal of Computing and Information Science in Engineering,* ***11***(2), p. 021003.

[33] Umeda, Y., Takeda, H., Tomiyama, T., and Yoshikawa, H., 1990. "Function, behaviour, and structure". *Applications of artificial intelligence in engineering V,* ***1***, pp. 177–194.

[34] Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., and Tomiyama, T., 1996. "Supporting conceptual design

based on the function-behavior-state modeler". *Ai Edam,* ***10***(4), pp. 275–288.
[35] Umeda, Y., Tomiyama, T., and Yoshikawa, H., 1995. "Fbs modeling: modeling scheme of function for conceptual design". In Proc. of the 9th Int. Workshop on Qualitative Reasoning, pp. 271–8.
[36] Umeda, Y., and Tomiyama, T., 1997. "Functional reasoning in design". *IEEE expert,* ***12***(2), pp. 42–48.
[37] Tomiyama, T., Umeda, Y., and Yoshikawa, H., 1993. "A cad for functional design". *CIRP Annals-Manufacturing Technology,* ***42***(1), pp. 143–146.
[38] Yoshioka, M., Umeda, Y., Takeda, H., Shimomura, Y., Nomaguchi, Y., and Tomiyama, T., 2004. "Physical concept ontology for the knowledge intensive engineering framework". *Advanced Engineering Informatics,* ***18***(2), pp. 95–113.
[39] Shimomura, Y., Yoshioka, M., Takeda, H., Umeda, Y., and Tomiyama, T., 1998. "Representation of design object based on the functional evolution process model". *Journal of Mechanical Design,* ***120***(2), pp. 221–229.
[40] Kitamura, Y., Kashiwase, M., Fuse, M., and Mizoguchi, R., 2004. "Deployment of an ontological framework of functional design knowledge". *Advanced Engineering Informatics,* ***18***(2), pp. 115–127.
[41] Goel, A. K., Rugaber, S., and Vattam, S., 2009. "Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language". *Ai Edam,* ***23***(1), pp. 23–35.
[42] Goel, A. K., and Bhatta, S. R., 2004. "Use of design patterns in analogy-based design". *Advanced Engineering Informatics,* ***18***(2), pp. 85–94.
[43] Bhatta, S., Goel, A., and Prabhakar, S., 1994. "Innovation in analogical design: A model-based approach". In Artificial Intelligence in Design94, Springer, pp. 57–74.
[44] Yaner, P. W., and GOEL, A. K., 2006. "From form to function: from sbf to dssbf". In *Design Computing and Cognition06*. Springer, pp. 423–441.
[45] Bracewell, R. H., and Sharpe, J., 1996. "Functional descriptions used in computer support for qualitative scheme generationschemebuilder". *Ai Edam,* ***10***(4), pp. 333–345.
[46] Welch, R. V., and Dixon, J. R., 1992. "Representing function, behavior and structure during conceptual design.". In 4 th International Conference on Design Theory and Methodology, pp. 11–18.
[47] Welch, R. V., and Dixon, J. R., 1994. "Guiding conceptual design through behavioral reasoning". *Research in Engineering Design,* ***6***(3), pp. 169–188.
[48] Deng, Y.-M., Britton, G., and Tor, S. B., 2000. "Constraint-based functional design verification for conceptual design". *Computer-Aided Design,* ***32***(14), pp. 889–899.
[49] Deng, Y.-M., 2002. "Function and behavior representation in conceptual mechanical design". *AI EDAM,* ***16***(5), pp. 343–362.
[50] Chakrabarti, A., and Bligh, T. P., 2001. "A scheme for functional reasoning in conceptual design". *Design Studies,* ***22***(6), pp. 493–517.
[51] Chakrabarti, A., Sarkar, P., Leelavathamma, B., and Nataraju, B., 2005. "A functional representation for aiding biomimetic and artificial inspiration of new ideas". *Ai Edam,* ***19***(2), pp. 113–132.
[52] 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.
[53] Gero, J. S., 1990. "Design prototypes: a knowledge representation schema for design". *AI magazine,* ***11***(4), p. 26.
[54] Gero, J. S., and Kannengiesser, U., 2004. "The situated function–behaviour–structure framework". *Design studies,* ***25***(4), pp. 373–391.
[55] Dorst, K., and Vermaas, P. E., 2005. "John geros function-behaviour-structure model of designing: a critical analysis". *Research in Engineering Design,* ***16***(1-2), pp. 17–26.
[56] Snooke, N., and Price, C., 1998. "Hierarchical functional reasoning". *Knowledge-based systems,* ***11***(5-6), pp. 301–309.
[57] Chandrasekaran, B., and Josephson, J. R., 2000. "Function in device representation". *Engineering with computers,* ***16***(3-4), pp. 162–177.
[58] Chandrasekaran, B., 2005. "Representing function: relating functional representation and functional modeling research streams". *Ai Edam,* ***19***(2), pp. 65–74.
[59] Keuneke, A. M., 1991. "Device representation-the significance of functional knowledge". *IEEE expert,* ***6***(2), pp. 22–25.
[60] Keuneke, A., and Allemang, D., 1989. "Exploring the no-function-in-structure principle". *Journal of Experimental & Theoretical Artificial Intelligence,* ***1***(1), pp. 79–89.
[61] Sen, C., Summers, J. D., and Mocko, G. M., 2011. "A protocol to formalise function verbs to support conservation-based model checking". *Journal of Engineering Design,* ***22***(11-12), pp. 765–788.
[62] Kurtoglu, T., and Campbell, M. I., 2009. "Automated synthesis of electromechanical design configurations from empirical analysis of function to form mapping". *Journal of Engineering Design,* ***20***(1), pp. 83–104.
[63] Sridharan, P., and Campbell, M. I., 2005. "A study on the grammatical construction of function structures". *AI EDAM,* ***19***(3), pp. 139–160.
[64] Campbell, M. I., 2009. A graph grammar methodology for generative systems. Tech. rep.
[65] Helms, B., and Shea, K., 2012. "Computational synthesis of product architectures based on object-oriented graph grammars". *Journal of Mechanical Design,* ***134***(2), p. 021008.

[66] Qian, L., and Gero, J. S., 1996. "Function–behavior–structure paths and their role in analogy-based design". *AI EDAM,* ***10***(4), pp. 289–312.

[67] Bohm, M. R., Stone, R. B., and Szykman, S., 2005. "Enhancing virtual product representations for advanced design repository systems". *Journal of Computing and Information Science in Engineering,* ***5***(4), pp. 360–372.

[68] McIntire, M. G., 2016. "From functional modeling to optimization: Risk and safety in the design process for large-scale systems". PhD thesis, Oregon State University.

[69] Keshavarzi, E., McIntire, M., Goebel, K., Tumer, I. Y., and Hoyle, C., 2017. "Resilient system design using cost-risk analysis with functional models". In ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V02AT03A043–V02AT03A043.

[70] Slater, M. R., and Van Bossuyt, D. L., 2015. "Toward a dedicated failure flow arrestor function methodology". In ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V02AT03A050–V02AT03A050.

[71] Short, A. R., and Van Bossuyt, D. L., 2015. "Active mission success estimation through phm-informed probabilistic modelling". In Annual Conference of the Prognostics and Health Management Society.

[72] Short, A.-R., Lai, A. D., and Van Bossuyt, D. L., 2018. "Conceptual design of sacrificial sub-systems: failure flow decision functions". *Research in Engineering Design,* ***29***(1), pp. 23–38.

[73] Arlitt, R., Van Bossuyt, D. L., Stone, R. B., and Tumer, I. Y., 2017. "The function-based design for sustainability method". *Journal of Mechanical Design,* ***139***(4), p. 041102.

[74] Lynch, K., Ramsey, R., Ball, G., Schmit, M., and Collins, K., 2016. "Ontology-driven metamodel validation in cyber-physical systems". In *Information Technology: New Generations*. Springer, pp. 1255–1258.

[75] Sztipanovits, J., Bapty, T., Neema, S., Howard, L., and Jackson, E., 2014. "Openmeta: a model-and component-based design tool chain for cyber-physical systems". In Joint European Conferences on Theory and Practice of Software, Springer, pp. 235–248.

[76] Simko, G., Lindecker, D., Levendovszky, T., Neema, S., and Sztipanovits, J., 2013. "Specification of cyber-physical components with formal semantics–integration and composition". In International Conference on Model Driven Engineering Languages and Systems, Springer, pp. 471–487.

[77] Henley, E. J., and Kumamoto, H., 1981. *Reliability engineering and risk assessment*, Vol. 568. Prentice-Hall Englewood Cliffs (NJ).

[78] OHalloran, B. M., Haley, B., Jensen, D. C., Arlitt, R., Tumer, I. Y., and Stone, R. B., 2014. "The early implementation of failure modes into existing component model libraries". *Research in Engineering Design,* ***25***(3), pp. 203–221.

[79] , 1949. *Procedures for Performing a Failure Mode, Effects and Criticality Analysis*.

[80] Tumer, I., Barrientos, F., and Mehr, A. F., 2005. "Towards risk based design (rbd) of space exploration missions: a review of rbd practice and research trends at nasa". In ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 687–695.

[81] Stone, R. B., Tumer, I. Y., and Van Wie, M., 2005. "The function-failure design method". *Journal of Mechanical Design,* ***127***(3), pp. 397–407.

[82] Lough, K. G., Stone, R., and Tumer, I. Y., 2009. "The risk in early design method". *Journal of Engineering Design,* ***20***(2), pp. 155–173.

[83] IAEA, 1993. Defining initiating events for purposes of probabilistic safety assessment. Tech. Rep. IAEA-TECDOC-719, International Atomic Energy Agency.

[84] Zamanali, J., 1998. "Probabilistic-risk-assessment applications in the nuclear-power industry". *IEEE transactions on reliability,* ***47***(3), pp. SP361–SP364.

[85] Gilks, W. R., Richardson, S., and Spiegelhalter, D., 1995. *Markov chain Monte Carlo in practice*. CRC press.

[86] Gilks, W. R., 2005. "Markov chain monte carlo". *Encyclopedia of Biostatistics*.

[87] Brooks, S., Gelman, A., Jones, G., and Meng, X.-L., 2011. *Handbook of markov chain monte carlo*. CRC press.

[88] David, P., Idasiak, V., and Kratz, F., 2010. "Reliability study of complex physical systems using sysml". *Reliability Engineering & System Safety,* ***95***(4), pp. 431–450.

[89] Beck, J. L., and Au, S.-K., 2002. "Bayesian updating of structural models and reliability using markov chain monte carlo simulation". *Journal of engineering mechanics,* ***128***(4), pp. 380–391.

[90] Koutra, D., Parikh, A., Ramdas, A., and Xiang, J., 2011. "Algorithms for graph similarity and subgraph matching". In Proc. Ecol. Inference Conf.

[91] Mehrpouyan, H., Haley, B., Dong, A., Tumer, I. Y., and Hoyle, C., 2015. "Resiliency analysis for complex engineered system design". *AI EDAM,* ***29***(1), pp. 93–108.

[92] Haley, B. M., Dong, A., and Tumer, I. Y., 2016. "A comparison of network-based metrics of behavioral degradation in complex engineered systems". *Journal of Mechanical Design,* ***138***(12), p. 121405.

[93] Fu, K., Chan, J., Cagan, J., Kotovsky, K., Schunn, C., and Wood, K., 2013. "The meaning of near and far: the impact

of structuring design databases and the effect of distance of analogy on design output". *Journal of Mechanical Design,* ***135***(2), p. 021007.

[94] Poppa, K., Arlitt, R., and Stone, R., 2013. "An approach to automated concept generation through latent semantic indexing". In IIE Annual Conference. Proceedings, Institute of Industrial and Systems Engineers (IISE), p. 151.

[95] Clarke, E. M., Grumberg, O., and Peled, D., 1999. *Model checking*. MIT press.

[96] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al., 1996. "A density-based algorithm for discovering clusters in large spatial databases with noise.". In Kdd, Vol. 96, pp. 226–231.

[97] Arnold, C. R. B., Stone, R. B., and McAdams, D. A., 2008. "Memic: an interactive morphological matrix tool for automated concept generation". In IIE Annual Conference. Proceedings, Institute of Industrial and Systems Engineers (IISE), p. 1196.

[98] Poll, S., Patterson-Hine, A., Camisa, J., Garcia, D., Hall, D., Lee, C., Mengshoel, O. J., Neukom, C., Nishikawa, D., Ossenfort, J., et al., 2007. "Advanced diagnostics and prognostics testbed". In Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07), pp. 178–185.

[99] Keller, W., and Modarres, M., 2005. "A historical overview of probabilistic risk assessment development and its use in the nuclear power industry: a tribute to the late professor norman carl rasmussen". *Reliability Engineering & System Safety,* ***89***(3), pp. 271–285.

[100] Bly, M., 2011. *Deepwater Horizon accident investigation report*. Diane Publishing.

[101] Ramp, I. J., and Van Bossuyt, D. L., 2014. "Toward an automated model-based geometric method of representing function failure propagation across uncoupled systems". In ASME 2014 International Mechanical Engineering Congress and Exposition, American Society of Mechanical Engineers, pp. V011T14A007–V011T14A007.

[102] Dekker, S., Cilliers, P., and Hofmeyr, J.-H., 2011. "The complexity of failure: Implications of complexity theory for safety investigations". *Safety Science,* ***49***(6), pp. 939–945.

[103] Scott, M. J., and Antonsson, E. K., 1999. "Arrow's theorem and engineering design decision making". *Research in Engineering Design,* ***11***(4), pp. 218–228.