Proceedings of the the ASME 2022
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE 2022
August 14-17, 2022, St. Louis, Missouri

DETC2022-90072

SYNTHETIC FAULT MODE GENERATION FOR RESILIENCE ANALYSIS AND FAILURE MECHANISM DISCOVERY

Daniel Hulse[1,*], Lukman Irshad[2]

[1]NASA Ames Research Center, Moffett Field, California
[2]NASA Ames Research Center (KBR, Inc.), Moffett Field, California

ABSTRACT

Traditional risk-based design processes seek to mitigate operational hazards by manually identifying possible faults and corresponding mitigation strategies—a tedious process which critically relies on the designer's limited knowledge. Resilience-based design, on the other hand, seeks to embody generic hazard-mitigating properties in the system to mitigate unknown hazards, often by modelling the system's response to potential randomly-generated hazardous events. This work creates a framework to adapt these scenario generation approaches to the traditional risk-based design process to synthetically generate fault modes, by representing them as a unique combination of internal component health-states which can then be injected and simulated in a model of the system failure dynamics. The design process may then reduce the risk of unknown internal hazards by iteratively mitigating the effects of these modes. The performance of this approach is evaluated in a model of an autonomous rover, where cluster analysis shows that elaborating the faulty state-space in the drive system using this approach uncovers a wider range of possible hazardous trajectories and failure consequences within each trajectory. However, this increase in hazard information gained from exhaustive mode sampling comes at a high computational expense, highlighting the need for advanced, efficient methods to search and sample the faulty state-space.

Keywords: Resilience, Fault Modelling, Simulation

1. INTRODUCTION

One key motivator for the incorporation of resilience in complex engineered systems is that one cannot always foresee the hazards which the system may encounter in operations [1–3]. Since individual hazard scenarios that are likely to unfold are not fully known, it is desirable for the system to have generic hazard-mitigating properties and behaviors, to achieve what is referred to as "graceful extensibility" [4]–the ability of the system to adapt to surprise events. That is, rather than solely designing the system to specifically mitigate a certain set of identified hazards (as is traditionally done in the FMEA process [5, Chp. 10]), one additionally wishes to design the system to be able to mitigate hazards which are unknown and have not yet been identified.

While many approaches for resilience incorporation have been presented, most (e.g., [6–8]) do not consider unknown hazards directly, instead focusing on improving the system's dynamic response to known hazards. We place methods that do account for unknown hazards in three categories:

* *Discursive design approaches* involve enabling the identification of potential fault modes by providing information (e.g., historical data [9–11] or failure archetypes [12]) to help the designer better expand their understanding of potential failure causes and behaviors. While these approaches enable the designer to identify more events that would have otherwise been unknown, they are limited by the historical data that exists.

* *Structural design approaches* involve changing the structural, behavioral, and parametric properties of the system to achieve a desired level of inherent resilience. This includes network structure-based design–where the network structure of the system is modified to prevent failures from propagating and provide redundant functionality in the case of a failure [13–15]–and capacity-based design–where a buffer is placed on the system's known failure limits to account for failure causes outside the designer's knowledge [16]. While these approaches enable the designer to incorporate the generic property of resilience into the system, they are limited in their ability to determine how the generic resilience should be leveraged to respond to unknown hazardous scenarios (i.e., the contingency management actions).

* Finally, *scenario-based design approaches* involve procedurally generating scenarios for the system to respond to, which can come from the representation of the system itself (e.g.,

*Corresponding author: daniel.e.hulse@nasa.gov

failures in individual or joint components) [17, 18] or some combination of randomly or selectively-generated parameters (internal or external) [19–21]. While these methods enable designers to uncover further scenarios they may not otherwise, they are limited in terms of the sampled distributions and the modelled failure dynamics (which may differ in the real system).

Of these, the scenario-based approaches align best with the framework of using dynamic simulations to determine the system's post-fault recovery performance (e.g., in the resilience triangle [6]). One open research question for the scenario-based methods is how to best generate the set of scenarios to analyze the system over. While prior work in this area has demonstrated the incorporation of resilience with respect to events that originate outside the system (e.g., changing planetary conditions [19], adversarial attacks [20–22]), when designing the contingency management of a system, there has been little to translate this perspective into the traditional engineering process, where the hazards of interest arise from internal failure mechanisms which lead to system failure. Conversely, while traditional risk-based design approaches (e.g., FMEA, FFDM) enable one to consider the system's ability to handle identified fault modes, they are limited in their ability to discover new modes. As a result, these methods are most effective during the redesign of an existing system with operational hazard data [23, 24]. When designing a new system, however, there is much less data that can be used to inform design. Hence, to aide resilient design, there is an opportunity to use scenario-based approaches to help the designer discover new modes which they might not otherwise identify.

The main contribution of this work is the development of a synthetic mode generation approach which leverages fault simulation to enable the designer to not just consider the consequences of discrete identified fault modes, but a faulty state-space resulting from the underlying failure mechanisms of the system. Because this assessment happens in simulation, a large number of potential fault modes can be evaluated rapidly, making it possible to exhaustively evaluate system resilience over the space of hazard-producing parameters. We further demonstrate how to evaluate the performance of these approaches in a model of an autonomous rover whose task is to follow line markings by using cluster analysis to compare the space of consequences revealed by this approach and manual identification. The next sections present background about risk-based design and resilience simulation (Section 2), the mode generation approach (Section 3), the evaluation of different mode generation approaches in the rover model (Section 4), and the discussion of and conclusions from this evaluation (Sections 5 and 6).

2. BACKGROUND

To contextualize how this approach fits within the risk-based design process and previous approaches for resilience simulation, this section covers how fault risk is considered in engineering design and how existing fault simulation approaches have been leveraged in this process.

2.1 Risk-based Design

Removing risk is an inherent part of the engineering process. Traditional mechanical engineering analysis is often oriented around preventing defined modes of failure like stress and fatigue [25]. General consideration of risk is traditionally taken into account in the early engineering process using a Failure Modes and Effects Analysis (FMEA) where the failure modes are identified, effects are determined, and the overall risk evaluated by estimating the likelihood and severity [5, Chp. 10]. The FMEA is a tabular analysis across the entire system, and can be developed for the functions, components, and manufacturing processes for the system as the design process advances through the conceptual, embodiment, and construction design stages [5, Chp. 10]. While the FMEA is a helpful risk identification tool, it can be time-consuming, difficult to iterate on, and prone to designer biases–especially the designer's lack of knowledge in the fault mode brainstorming process [26].

Simulation-based design approaches can help resolve these limitations by automating the evaluation of failure consequences with a model that determines how a fault causes hazardous states given the component/functional dependencies and/or system behavior. With a fault simulation, one can thus automatically generate the FMEA iteratively over time as the system changes without having to go through a detailed expert-driven analysis each time [27]. Fault simulation approaches also enable one to evaluate a larger set of fault scenarios (such as joint fault modes [28] or faults injected over a range of model parameters e.g. injection times [29]) to reveal more information about how and in what circumstances hazardous scenarios unfold. In this paper, we further build on these simulation-based hazard identification approaches to generate new modes and further reveal the *faulty state-space* inherent to a given system's behavior and parameters.

2.2 Fault Simulation for Resilience

Simulation is used in the resilience-based design process to evaluate the system response to hazardous scenarios. A number of prior formalisms have been developed for fault simulation [30], including network topology models [31], failure logic/graph-based error propagation [32], dynamic behavioral simulation [33, 34], and stochastic simulation [35]. In early design, it is often necessary to consider the high-level hazards inherent to the conceptual design of the system. To accomplish this, frameworks such as Functional-Failure Identification and Propagation [36], Inherent Behavior of Functional Models [37], and Function-based Failure Propagation [38] have been developed which propagate faults through the high-level function structure of the system to understand the risks [39]. This work uses the fmdtools package and framework to evaluate faults, which builds on this work and is capable of embodying a number of simulation formalisms by using an object-oriented dynamic representation of the system structure and behavior [40].

One of the more common applications of fault application in practice is software design, where it is used to ensure that the software will continue to perform well to hazards that may inevitably occur in operations [41, 42]. Since software can run on a number of different machines and faults can additionally come from external attacks, software flaws, and operator errors [42],

one may not be able to accurately predict the distributions of the underlying faults [43]. Thus, while a number of software fault injection tools use fault distributions based on the underlying physics of failure for the underlying hardware, random sampling possible faults [43–46] and orchestrated attack approaches [47] are also used. In some situations (where the relative probability of faults is within the same order of magnitude), the results from fault mode sampling are quite similar to the results one would get if the underlying distribution is known [43]. These same processes are used in the field of Chaos Engineering, the main difference being that Chaos Engineering approaches test the software in production, rather than in a controlled test setup [48].

The difference between using fault injection to incorporate resilience in software systems and a general engineered system is the physical constraints which shape the underlying design trade-space and hazard-space. In software system fault injection, the faulty state-space is often defined by potential hardware modes (e.g. bit-flips) which creates a large faulty state-space (because of the large numbers of bits), the distribution of which is relatively uniform. In the early design context, on the other hand, the modes are not fully identified and the underlying distribution is not necessarily known very well. The physics of failure of the system will make it such that some modes are very prevalent while others are not. Additionally, the space of potential design solutions available in early design is quite large and often does not come "for free" as it would in software design–fail-safes and redundancies could come at the cost of significant loss of efficiency. Thus, to best leverage the adaptation presented in this work, it is important to consider the costs associated with possible failure mitigations–if a mitigation comes with significant costs, it may be necessary to balance this cost against the cost of unknown failures (for which more data would be preferred). If the mitigations come at no cost, precise valuation does not have to be considered. For example, designing the contingency management in the control system in Fig. 1, requires no consideration of trade-offs because each individual fault state is unique and new fault information only helps the rover decide what to do in that particular instance.

3. METHOD

Synthetic mode generation augments the traditional failure mode identification process sampling the *faulty state-space*, which is composed of performance-effecting parameters which can lead to failures if perturbed, to generate a set of potential fault modes. That is, instead of identifying the specific, discrete modes which lead to failure, the designer identifies behavior-affecting fault states. These fault states are given domains for nominal and off-nominal behaviors. The synthetic modes are then constructed by systematically permuting the fault states within these ranges, and simulated to reveal the corresponding set of consequences. This information can then be iterated over in the design process as shown in Fig. 3, with the designer adding features or changing variables to reduce the severity of the modelled consequences. The next sections explain the underlying health-state formalism, mode elaboration approach, and its use in the design process in more detail.

3.1 Formalism

To understand the fault mode elaboration approach, consider the notional model shown in Fig. 1, which illustrates the underlying formalism. As shown, a function may be composed of a control system and a component. The component comprises the physical behaviors (`f1`, `f2`, and `f3`) of the system, which translate the inputs of the function `X` into outputs `Y` at each time in the simulation. The high-level control system controls the component behavior by switching between modes depending on the current variable states. In Fig. 1, for example, the system alternates between mode `m1` and `m2` depending on the value of input variable `X3`, which in turns controls the action variable `a` and thus the behaviors in equations `f1` and `f2`. These attributes effectively define the nominal operations of the system. Hazardous operations are defined using fault states, which are variables (`h1`, `h2`, and `h3`) that modify the behaviors of the component by taking on different values than in the nominal state. Depending on the behaviors, these states can be given continuous (e.g., range (0…2)) or discrete (e.g., $-1, 0, 1, 10$) domains. A hazardous mode (e.g., H_1, H_2 ... H_n) in this instance is thus a tuple of fault states (`h1,h2,h3`) where at least one state is off-nominal. These hazards may further be controlled by the control system hazard management, which may switch the modes of the system (e.g., `m1` to `m2`) or cause it to enter specialized hazard management modes depending on states (e.g., `S1` or `S2`) detected by the system.

To illustrate this formalism, consider the case of a household lamp. In this system, the function of providing light is embodied by a component (light bulb) which translates input electrical potential into visible light, and is controlled by a control system (light switch), which determines whether electricity flows through the bulb based on states input by the user. Fault modes then arise through physical modification of the component behavior (e.g., burning out, damaged contacts, excess heat, etc.), which would correspond to changes in the underlying equations (e.g., setting circuit inverse resistance, and thus energy flow, to zero). Typically, hazard management (in the instance of a burned out mode) would be accomplished by the user detecting that the bulb is burned and replacing it.

3.2 Mode Elaboration

Hazards in the system are thus represented as modifications of the component fault states away from the nominal state, as shown in Fig. 1. The set of possible fault modes is thus the set of possible permutations of the fault states in the function. This set can be defined as having the form of H:

$$H = \{[h_1, ...h_j, ...h_m] \neq \vec{1}, h_j \in D_j\} \quad (1)$$

where D_j is the range of possible states for the heath state h_j. This set can become quite large, since it includes not only the single permutations of fault states, but every possible joint-permutation of fault states as shown in Fig. 1. Thus, as the number of fault states m (and possible values D) increases, the number of potential fault modes increases by $\mathcal{O}(D^m)$. As a result, as the faulty state-space increases in size, it may become necessary to reduce model computational costs and lower the number of states needed to query by (1) constraining the set (e.g., specifying exclusivity

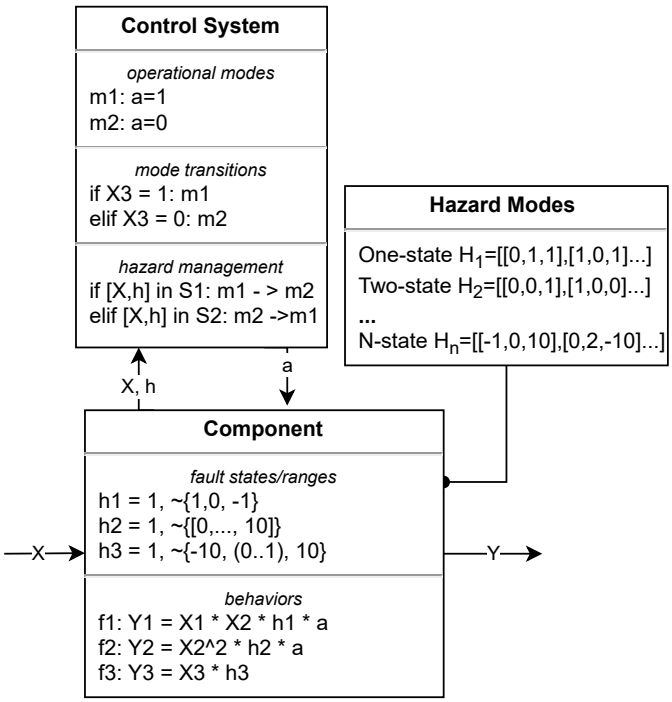

FIGURE 1: EXAMPLE FUNCTION BEHAVIOR AND STATE REPRESENTATION

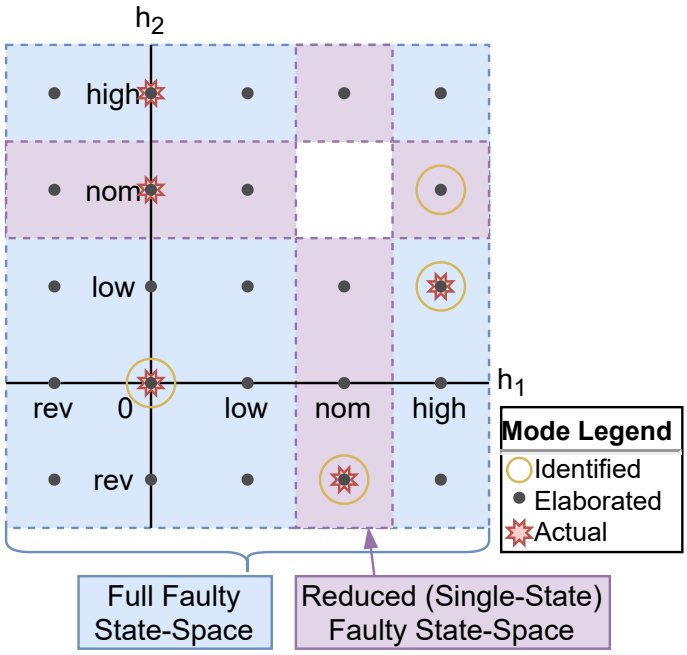

FIGURE 2: GENERIC FAULTY STATE-SPACE REVEALED BY MODE ELABORATION APPROACHES

between perturbations of identified fault states), (2) only considering a subset of n joint perturbations at a time, or (3) sampling from the set in a way that admissibly represents the set of hazards for the purpose of the analysis. In particular, existing theory and/or lab data establishing the relationships between fault states should be used to narrow the space of modes to physically realizable combinations of fault states.

3.3 Process

Designing a system using this approach follows the process shown in Fig. 3. In this process, one first develops the fault model and then iterates over the model features to produce a resilient design–a typical approach for a risk-based design process. It begins by first creating a nominal model with its respective modes and behaviors. These fault states (and their domains) are first identified (using the form of the equation, rather than past experience) and added to the system's behavioral equations. While this identification is within the designer's judgement, there are several rules which may be employed to identify these states systematically. Fault states are factors which modify the behavioral equations of the system through multiplication/division (representing amplification/reduction) and addition/subtraction (representing a constant drift). Defining the domain of fault states can further result from identifying the feasible and meaningful limits of the underlying equations. That is, the fault state domain should be restricted such that the resulting range of output values is physically realizable (e.g., if a factor value would lead to undefined or infinite output variables from the function, that value should be removed from the range) and the results in meaningful output (e.g., inputs for periodic functions need only be defined over the period). Additionally, depending on the desired mode sampling approach, it may be helpful to limit the set of possible fault state values to discrete quantities of interest (e.g., low, nominal, high, etc) rather than treating the domain as continuous. While this restricts the possible set of modes, it also gives the designer more ability to selectively consider specific health-state values and value combinations, and can help manage the computational expense by reducing the size of the domain. After these modes are generated, the designer proceeds to simulate the model over these modes to assess the resilience of the system. Since the faulty state-space generated is large, the modes (and their effects) must be assessed in aggregate. Based on this analysis, the designer changes the behavior, operations, and/or hazard management controls to improve the distribution of system resilience metrics across the responses. This continues until the design is satisfactorily resilient.

The main difference between this approach and the conventional FMEA approach is that instead of identifying fault modes–defined, known patterns characterizing the states of the system as it fails–one instead identifies fault states which could impact the performance of the system and lead to hazards and associates a set or range of perturbations with these states. The resulting hazard assessment thus includes a much larger set of modes than can be manually identified, as illustrated in Fig. 2. As shown, in the future (designed) system, there will be a set of modes which will arise in operations. While a number of modes can be identified by the designer, some of them will not, and others will be misidentified as possible modes which will not be realized in the actual system. In the approach presented here, the entire faulty state-space (or some defined subset or sample) is instead elaborated. This increases the set of modes considered in the analysis–catching more modes than would otherwise be identified, but also including a much larger set of modes which may

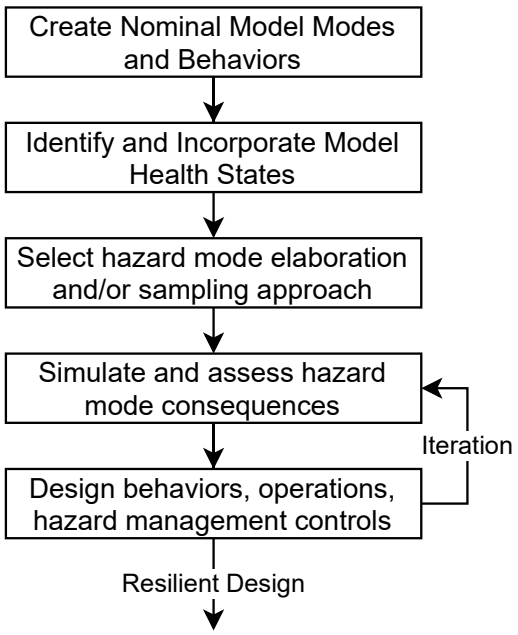

FIGURE 3: RESILIENCE ASSESSMENT AND DESIGN PROCESS

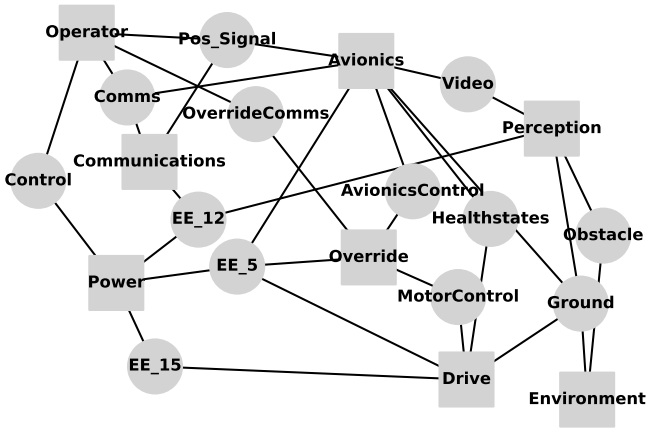

FIGURE 4: ROVER MODEL STRUCTURE

not (or may not be likely to) arise in the true system. To limit the space of potential modes to realizable hazards, it is thus helpful to identify constraints on both health-state combinations and ranges from the physical limitations on and known/tested relationships between these fault states.

4. DEMONSTRATION

To demonstrate the mode generation approach, we use a model of an autonomous rover (Sec 4.1) to show how a model can be set up, the resulting faulty state-space uncovered when the possible modes are elaborated and simulated, and a comparison of this analysis with a more traditional mode identification-based approach.

4.1 Rover Model

To demonstrate this framework, this paper presents the design of an autonomous rover. This rover was modelled at a high level

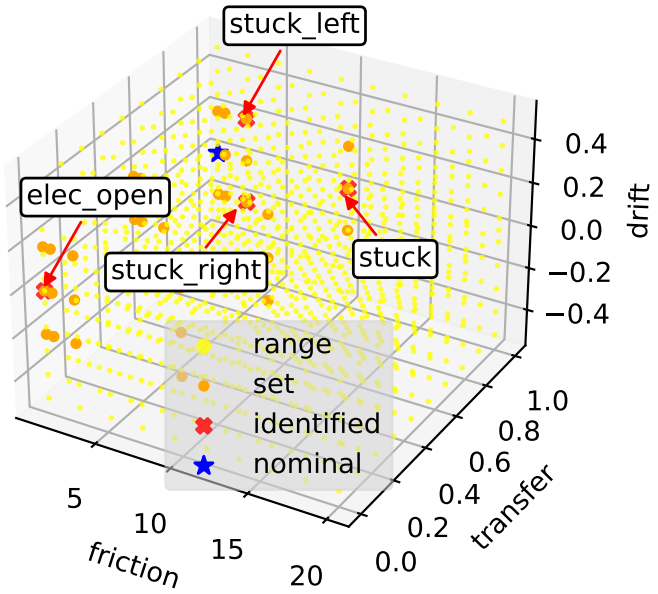

FIGURE 5: FAULTY STATE-SPACE UNCOVERED BY THE DIFFERENT FAULT MODE GENERATING METHODS

to perform a basic autonomous driving task, with the functional model of the model shown in Fig. 4. As shown, this model encompasses the rover power and control systems, as well as its drive system, avionics, and interactions with its environment (i.e., movement and position with respect to a map). The task is to follow a given line from a given starting location to a given end location. While many different input lines can be used for different routes, the route used in this paper for demonstration purposes follows a simple L-Curve as shown in Fig. 7. If the rover deviates from the center line, it may go off course and crash into its surroundings. When the distance from the center line is greater than 1m, the rover can no longer see the center line and stops moving, because the rover has crashed. The major fault effect considered here is thus how far the rover deviates from the center line in fault scenarios–with the priority being fault scenarios which directly lead to a crash.

4.2 Approach Setup

To address these faults (and show the value of synthetic mode sampling), this work uses three different approaches for generating fault modes:

* a manual identification approach, where a few select modes were manually identified by the user
* a set elaboration approach where a few discrete perturbations were identified for each fault state of the function, which were combined to form the space of synthetic modes
* an range elaboration approach where ranges were identified for each fault state of the function, which were then elaborated and combined at a high resolution to form the space of synthetic modes

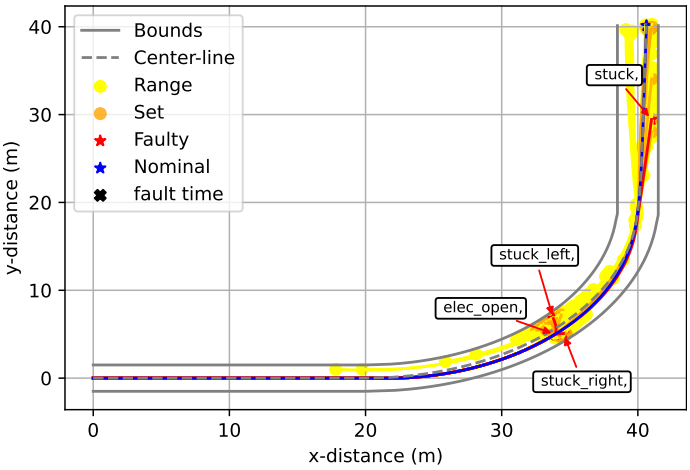

FIGURE 6: FAULTY TRAJECTORIES OVER FAULTY STATE-SPACE

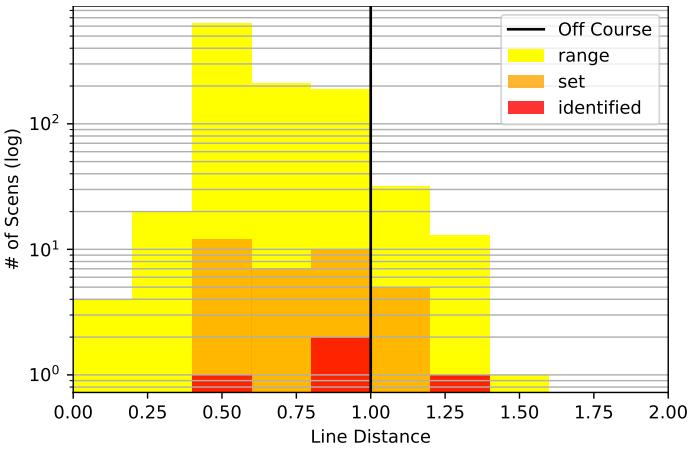

FIGURE 7: DISTANCE OF ROVER FROM CENTER LINES IN SCENARIOS

For simplicity of demonstration, this analysis focused solely on faults originating from the Drive function, the goal of which is to move and turn the rover given external power and control inputs for velocity and heading. It also focused on a single fault time where the rover is in the middle of turning (rather than sampling several times). In the drive system, the following fault states were defined to define the faulty state-space:

* friction, which is resistance that makes the rover require more power to move a given distance
* transfer, which is the ability of the rover to move forward in a given time-step at a level of input power
* drift, which is the misalignment of the rover trajectory from its intended heading.

The resulting fault state combinations from the mode sampling approaches are shown in Fig. 5. As shown, the three identified modes are:

* `stuck`, where the rover is blocked from moving forward,
* `stuck_left` and `stuck_right`, where the left and right side of the rover is blocked from moving, respectively, and
* `elec_open`, where the power is disconnected from the rover motors.

The set and range elaboration approach cover a much larger portion of the faulty state-space, with many more combinations of fault state variables. However, this increased coverage comes at the cost of simulation expense and tractability. Thus, it is important to understand what is gained by elaborating these additional states. For the purposes of comparison, the next subsection will discuss the simulation of these scenarios to determine if these approaches can represent the faulty state-space better.

4.3 Analysis Comparison

Simulating each of these fault modes in the model at a given time results in the fault trajectories shown in Fig. 6. As shown, the manually-identified modes mainly result in the rover stopping at the fault injection location, with one (`stuck`) resulting in the rover advancing further but not reaching the goal. In the set elaboration approach, several more trajectories are uncovered in which the rover makes it closer to the goal. In the range elaboration approach, a much larger space of trajectories are uncovered–including ones where the rover turns around and travels along the line backwards. The corresponding distribution of line distances (the metric for identifying when crashes occur) for each approach is shown in Fig. 7. As shown, the raw number of scenarios in the range elaboration approach is orders of magnitude higher than the set elaboration and manual identification approaches, resulting in many more scenarios which result in a crash. Additionally, the coverage of the space of the severity of failures (i.e., the distance from the line) was much larger for the range elaboration approach compared to the other approaches, with a range of 0.0m to 1.6m instead of 0.3m to 1.3m for the set approach and 0.3m to 1.4m for manual identification. As a result, relying on manual identification or the set elaboration approach in this instance would have resulted in missing the possible severity ranges of less than 0.3m and greater than 1.3m entirely. This ability to identify a broader range of scenarios is important because (1) it helps one better identify the true worst-case scenario possible and (2) low-severity failures can often play an important role throughout the product lifecycle if they occur more often than expected by driving up maintenance cost and making more severe joint-fault modes more likely.

While the range elaboration approach uncovers a wider range of fault severities, it should be noted that the vast majority (note the log scale in Fig. 6) are still within the range expected from the set elaboration approach and manual identification–only a few additional severities are caught in the tails of the distribution. However, given the range elaboration approach has such a high resolution, one may wonder if this has actually uncovered more meaningful hazard information. That is, simulating a large number of fault state combinations at high resolution may result in a large number of scenarios that essentially play out in the same way and do not add anything more to the analysis. To evaluate the extent to which the range approach is affected by this, the next section uses clustering to identify and group similar simulation results.

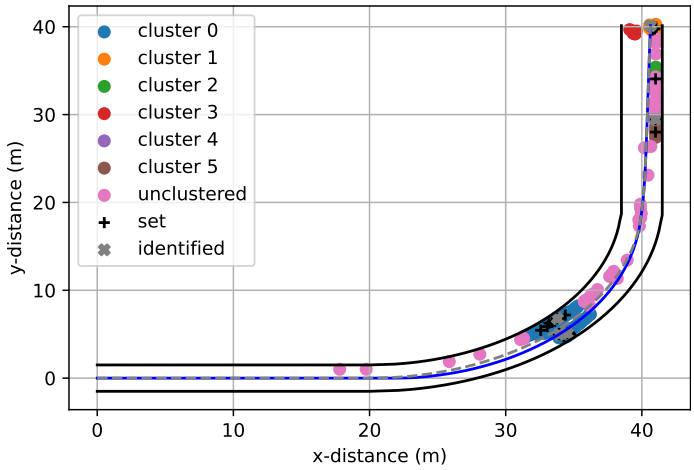

FIGURE 8: CLUSTERED ROVER END LOCATIONS OVER THE SET OF FAULT SCENARIOS

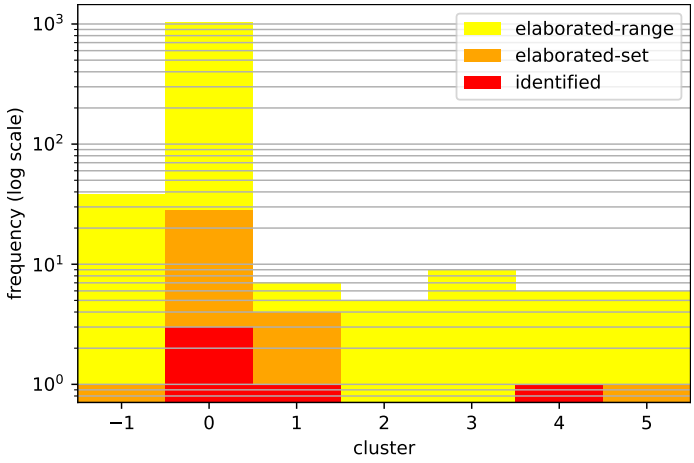

FIGURE 9: FREQUENCY OF CLUSTER TYPES IN EACH APPROACH.

4.3.1 Cluster Analysis. Clustering was used to identify similar sets of results where the fault state combinations caused a similar set of results. To perform this evaluation, the DBSCAN algorithm [49] in scikit-learn [50] was used, which grouped the final locations of the simulations in terms of x-y coordinates. The DBSCAN algorithm was used to identify duplicates because it (as a density-based algorithm) identifies densely-packed clusters of points and neglects outliers (as opposed to other clustering methods, which group all points spatially into a set number of categories). As shown in Fig. 8, the algorithm identified five clusters (and the -1 cluster, which is made of data points which do not fit with any of the others). Figure 9 shows the distribution of scenarios in the different fault generation approaches across these clusters. As shown, the vast majority of the scenarios in the range elaboration approach occur in the 0 cluster, which represents scenarios where the rover stops in the immediate vicinity of the location where the fault was injected. However, it also better covers the space of clusters than the set elaboration and manual identification approaches, since every cluster is represented and many more scenarios still are included in the -1 cluster, which by definition are non-duplicate scenarios. From the designer's perspective, they would not have learned about 3 of these clusters (due to the lack of scenarios) if they used manual identification, and 2 clusters if they used the set elaboration approach, leaving a significant gap in terms of their understanding of potential risk. While spanning the space of clusters is important (since it shows that unique fault trajectories were identified), identifying multiple scenarios within each cluster is not necessarily a waste if it helps to identify the worst-case consequences that could result from that cluster. Table 1 shows the number of scenarios found in each cluster for each approach, the spatial coverage of this cluster, and the worst-case line distance identified. Coverage loss is measured using the metric in equation 2, which is the proportion of the x-y range which is lost from the approach a compared to the overall ranges identified.

$$\frac{(x_{max} - x_{max,a}) - (x_{min,a} - x_{min})}{x_{max} - x_{min}} \frac{(y_{max} - y_{max,a}) - (y_{min,a} - y_{min})}{y_{max} - y_{min}} \quad (2)$$

As shown in Table 1, the identified and set approaches perform poorly on this metric, except in Cluster 0 and 1 because these approaches have only a few scenarios in the other clusters compared to the range approach. Additionally, the worst cases identified by the set elaboration and manual identification approaches are often less hazardous than the worst cases identified by the range elaboration approach, with a severity of less than one. So, if one was using manual identification, they would have only considered Cluster 4 to have a high severity (distance from the line greater than 1m) scenario, missing the worst-cases faults in the rest of the clusters (and as a result, would not even be aware of the existence of the rest of the clusters). While the set elaboration approach performs better than manual identification for this, the range elaboration approach performs best. Thus, even though there are many duplicates in the range elaboration approach for each cluster, these duplicates enable it to better represent the hazards within the cluster in terms of the representing the variation within the cluster, and thus identifying the worst cases that could result from a mode of a given type.

4.3.2 Summary. A summary of the results provided by each approach is shown in Table 2. As shown, the set elaboration approach simulated a much larger number of scenarios than manual mode identification, while the range elaboration approach simulated over an order of magnitude more scenarios than the set elaboration approach. This results in a much larger computational cost. While this is not a significant burden for this model, it could result in significantly more costs if more parameters were included or if the simulation itself were more computationally expensive. However, the use of these approaches uncovered substantially more hazardous trajectories, which was reflected both in the number of clusters represented in these approaches as well as the number of scenarios which did not fit in any cluster. In general, while the elaborated-set approach represented an improvement on manual fault identification, many more were identified by the elaborated-range approach. Of the failure clusters that were elaborated in the range elaboration approach, only a small proportion of them were identified beforehand, and even the set elaboration approach was not able to represent the full percent. Thus, even though the range elaboration approach requires

TABLE 1: CLUSTER COVERAGE

Cluster		Identified	Set	Range
-1	# Scenarios	0	1	38
	Coverage Loss	1.00	1.00	0.00
	Worst-Case	0.00	1.03	1.25
0	# Scenarios	3	17	919
	Coverage Loss	0.43	0.18	0.00
	Worst-Case	0.90	1.03	1.08
1	# Scenarios	1	4	7
	Coverage Loss	1.00	0.10	0.00
	Worst-Case	0.63	1.05	1.15
2	# Scenarios	0	0	5
	Coverage Loss	1.00	1.00	0.00
	Worst-Case	0.00	0.00	1.38
3	# Scenarios	0	0	9
	Coverage Loss	1.00	1.00	0.00
	Worst-Case	0.00	0.00	1.07
4	# Scenarios	1	1	6
	Coverage Loss	1.00	1.00	0.00
	Worst-Case	1.29	1.29	1.40
5	# Scenarios	0	1	6
	Coverage Loss	1.00	1.00	0.00
	Worst-Case	0.00	1.05	1.37

TABLE 2: OVERALL COMPARISON OF APPROACHES

	Identified	Set	Range
Scenarios	5	35	1099
Comp. Time (s)	0.24	0.96	35.10
% Clusters	42.86	71.43	100.00
Unclustered	0	1	38

a much larger number of scenarios–many of which are essentially duplicates–it uncovers a much larger portion of the faulty state-space than would be possible to view otherwise. Even though there are many approximately duplicate scenarios, more information is also uncovered.

This is likely an artefact of the fault states explored–while we would expect some fault states to produce relatively consistent results at different levels (e.g., friction should just slow the rover down in different ways), a specific level of drift can cause a great variety of possible results, since it additionally modifies the direction of the rover. As a result, some level of drift can, for example, make the rover turn around and travel backwards, as shown in Fig. 6. However, this type of result is very sensitive to the precise value of the fault state parameter and is thus only detectable by elaborating (or exploring) the full space. This shows how full faulty state-space elaboration can identify more modes which would not be considered otherwise–by uncovering specific values of fault state parameters which change the faulty behavior of the system.

5. DISCUSSION

The synthetic mode generation approach presented here has a number of implications to resilience-informed design. The design of resilience is often taken from the perspective that the designer does not necessarily know the probability distribution of the hazards that the system may be subjected to. Thus, to minimize risk, a very broad and generic tolerance and adaptiveness needs to be built into the system to enable it to respond well to all hazards it could encounter. In this sense, from the perspective of resilience, it is very important to uncover new knowledge of the faulty state-space, even if those faults seem unrealistic or improbable. As shown in the previous section, synthetic mode generation can encourage this by elaborating an entire range of fault state combinations for hazard simulation and assessment, rather than the small list that can be directly identified by a designer. This approach can essentially be seen as the translation of resilience simulation and chaos engineering approaches–where a large number of externally-driven hazardous scenarios are presented to a system–to the traditional engineering design scenario where the hazards to prevent are internal to the system (i.e., component failures). While this method would be infeasible to perform manually due to the large space of modes, performing this sort of analysis in a computational environment is entirely practicable because the determining of fault mode consequences is performed by a computationally-inexpensive simulation.

However, from a risk-based design perspective, there are assumptions embedded in this approach which are important to consider. Given a probability distribution has not been given or assumed for fault state combinations, it may be difficult to understand how to weigh individual worst-case failure trajectories that only arise at particular fault state parameter value combinations: they could be highly improbable because of the narrow range under which they are realized (if the multivariate distribution is uniform), or they could be highly likely (if the range of the probability distribution where they are realised is very dense). Thus, to consider the risk of these events, an appropriate probability model should be set up and sampled with the modes that appropriately consider parameter ranges and range combinations.

The previous section shows that this approach can uncover meaningful information about the faulty state-space, which can help the designer understand how to most comprehensively mitigate potential failure events. For example, when viewing all of the different possible trajectories from the manually-identified modes, a designer may set a requirement that the system shut down if it deviates too far from the line, under the assumption that this would prevent it from crashing. However, this would still leave open the hazardous faulty behavior uncovered by the mode sampling approach in which the rover follows the line backwards. A more comprehensive and safer requirement might instead be for the rover to not deviate too far from its intended trajectory (position over time) while staying acceptably close to the line.

A major challenge for applying this method is how best to specify and sample the faulty state-space. First, it may not be clear how to determine the limits of the domain for each fault state, or, more broadly, determine what constraints limit the domain of possible joint fault state values. This can lead to difficulties–on one hand, if the domain for health-states are defined too narrowly, the approach loses its ability to capture faults that are outliers. For example, in the design of a car, the designer may have an imagined range of hazardous operating temperatures that could prove incorrect if the car ended up being operated during an extreme

whether event. As a result, they would leave out potential resulting fault modes in this process. On the other hand, if the domain is specified too broadly, the analysis may be dominated by fault modes which are impossible to realize (e.g., operating temperatures below 100K or above 4000K). As a result, it is important for the designer to appropriately exercise judgement over ranges (as they would manually identifying modes). Second, it may not be clear how best to sample the set of faults modes from the faulty state-space. From the previous section, we know that using the range elaboration approach with high resolution increases the amount of risk-related information that can be extracted, but these increases come at significant computational cost. When the fault sampling approach elaborates the entire set of fault modes from ranges, designers need to balance computational cost with knowledge increase. While this can be done on the basis of cost-benefit assessment (i.e., quantifying the value of hazard information), it also highlights the need for more efficient sampling methods. For example, in the analysis process, one may choose the resolution of fault states first by performing a sensitivity analysis to identify the influence of each fault state on the outcome (putting lower resolution on lower-impact states) and then successively increasing the resolution step by step until there is not much change in the hazard-related information to gain.

6. CONCLUSIONS

In conclusion, this work presented a method to synthetically generate fault modes for resilience analysis and resilience-based design. This approach works by elaborating a faulty state-space–a set of fault modes made up of parameters which affect the behavior of the system. The resulting space is much larger than a list of fault modes which the designer might uncover, but it also has a greater potential to reveal discrete faulty behaviors which can arise in the system. This is shown in the rover example in Section 4, where synthetic mode elaboration approaches are shown to find a larger quantity of potential hazardous outcomes, revealing previously-unknown hazardous behaviors, and better identifying worst-case scenarios for each behavior. Furthermore, among these approaches, it was shown that elaborating the full space within a range can better identify these scenarios than manually specifying specific values for the states up-front. However, both of these increases in hazard information come at the expense of simulation time, which increases substantially with every increase in resolution–an important consideration when this approach is used throughout a complex or computationally-expensive model, or as a part of an iterative design process.

6.1 Limitations and Future Work

This approach opens a number of potential directions for future work. Because of the computational expense of synthetic mode generation, and the potential for essentially identical duplicate modes to be generated and evaluated, future work needs to extract the high-level modes uncovered in an approach like this so that they can (1) be tractably understood and accounted for by the designer and (2) be more readily simulated at a lower computational cost. In this work, clustering was used to identify groups of modes which were essentially representative of each other. However, given there is a distribution of modes within each cluster, it may not be clear how to represent the cluster health-states and parameter values as a whole. Future work should thus determine how to best represent these groups for tractable analysis and design.

Reducing computational cost is also an important consideration for optimization. Since sampling modes generates more hazard information, it could lead to more optimal designs if used in an optimization loop. On the other hand, it could slow the optimization process down while not meaningfully changing the optimal variables. Future works should show how much value can be gained through increasing the faulty state-space to optimize over and help designers understand when to include increased faulty state-space resolution in the resilience optimization. Finally, to mitigate the trade-off between faulty state-space information and computational cost, future work should develop strategies which efficiently map out the faulty state-space by searching for parameters which lead to different results. Approaches such as monte carlo sampling, latin hypercube sampling, bayesian optimization, adaptive stress testing [21], etc., should all be explored to find how to most efficiently represent the faulty state-space without wasting computation on essentially duplicate fault modes.

ACKNOWLEDGMENTS

This research was partially conducted at NASA Ames Research Center. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

REFERENCES

[1] Hajikazemi, Sara, Ekambaram, Anandasivakumar, Andersen, Bjørn and Zidane, Youcef JT. "The Black Swan–Knowing the unknown in projects." *Procedia-Social and Behavioral Sciences* Vol. 226 (2016): pp. 184–192. DOI 10.1016/j.sbspro.2016.06.178.

[2] Aven, Terje. "On the meaning of a black swan in a risk context." *Safety science* Vol. 57 (2013): pp. 44–51. DOI 10.1016/j.ssci.2013.01.016.

[3] Aven, Terje. "Implications of black swans to the foundations and practice of risk assessment and management." *Reliability Engineering & System Safety* Vol. 134 (2015): pp. 83–91. DOI 10.1016/j.ress.2014.10.004.

[4] Woods, David D. "Four concepts for resilience and the implications for the future of resilience engineering." *Reliability Engineering & System Safety* Vol. 141 (2015): pp. 5–9. DOI 10.1016/j.ress.2015.03.018.

[5] Pahl, Gerhard and Beitz, Wolfgang. *Engineering design: a systematic approach*. Springer Science & Business Media (2013).

[6] Yodo, Nita and Wang, Pingfeng. "Engineering resilience quantification and system design implications: A literature survey." *Journal of Mechanical Design* Vol. 138 No. 11. DOI 10.1115/1.4034223.

[7] MacKenzie, Cameron A and Hu, Chao. "Decision making under uncertainty for design of resilient engineered systems." *Reliability Engineering & System Safety* Vol. 192 (2019): p. 106171. DOI 10.1016/j.ress.2018.05.020.

[8] Keshavarzi, Elham. "Resilient Design for Complex Engineered Systems in the Early Design Phase." Master's Thesis, Oregon State University. 2018. URL https://ir.library.oregonstate.edu/concern/graduate_thesis_or_dissertations/6969z576w.

[9] Falco, Gregory J. "City resilience through data analytics: A human-centric approach." *Procedia engineering* Vol. 118 (2015): pp. 1008–1014. DOI 10.1016/j.proeng.2015.08.542.

[10] Garnier, Emmanuel. "Lessons learned from the past for a better resilience to contemporary risks." *Disaster Prevention and Management: An International Journal* DOI 10.1108/DPM-09-2019-0303.

[11] Wilhelm, Bruno, Ballesteros Cánovas, Juan Antonio, Macdonald, Neil, Toonen, Willem HJ, Baker, Victor, Barriendos, Mariano, Benito, Gerardo, Brauer, Achim, Corella, Juan Pablo, Denniston, Rhawn et al. "Interpreting historical, botanical, and geological evidence to aid preparations for future floods." *Wiley Interdisciplinary Reviews: Water* Vol. 6 No. 1 (2019): p. e1318. DOI 10.1002/wat2.1318.

[12] Walsh, Hannah S, Dong, Andy, Tumer, Irem Y and Brat, Guillaume. "Detecting and Characterizing Archetypes of Unintended Consequences in Engineered Systems." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 83976: p. V008T08A021. 2020. American Society of Mechanical Engineers. DOI 10.1115/DETC2020-22108.

[13] Walsh, Hannah S. "The Structural Characteristics of Robustness in Large-Scale Complex Engineered Systems." Ph.D. Thesis, Oregon State University. 2020. URL https://ir.library.oregonstate.edu/concern/graduate_thesis_or_dissertations/cj82kf537.

[14] Paparistodimou, Giota, Duffy, Alex, Whitfield, Robert Ian, Knight, Philip and Robb, Malcolm. "A network tool to analyse and improve robustness of system architectures." *Design Science* Vol. 6. DOI 10.1017/dsj.2020.6.

[15] Markina-Khusid, Aleksandra, Jacobs, Ryan B, Antul, Laura, Cho, Lance and Tran, Huy T. "A Complex Network Framework for Validated Assessments of Systems of Systems Robustness." *IEEE Systems Journal* DOI 10.1109/JSYST.2021.3064817.

[16] Nafday, Avinash M. "Consequence-based structural design approach for black swan events." *Structural Safety* Vol. 33 No. 1 (2011): pp. 108–114. DOI 10.1016/j.strusafe.2010.09.003.

[17] Chopra, Shauhrat S, Dillon, Trent, Bilec, Melissa M and Khanna, Vikas. "A network-based framework for assessing infrastructure resilience: a case study of the London metro system." *Journal of The Royal Society Interface* Vol. 13 No. 118 (2016): p. 20160113. URL 10.1098/rsif.2016.0113.

[18] Zhang, Dong-ming, Du, Fei, Huang, Hongwei, Zhang, Fan, Ayyub, Bilal M and Beer, Michael. "Resiliency assessment of urban rail transit networks: Shanghai metro as an example." *Safety Science* Vol. 106 (2018): pp. 230–243. DOI 10.1016/j.ssci.2018.03.023.

[19] Short, Ada-Rhodes and DuPont, Bryony L. "Computational Cognition for Mission Command and Control Decisions Facing Risk in Unknown Environments." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 59193: p. V02BT03A020. 2019. American Society of Mechanical Engineers. DOI 10.1115/DETC2019-98483.

[20] Nguyen, Tien, Wang, Shiyuan, Alhazmi, Mohannad, Nazemi, Mostafa, Estebsari, Abouzar and Dehghanian, Payman. "Electric power grid resilience to cyber adversaries: State of the art." *IEEE Access* Vol. 8 (2020): pp. 87592–87608. DOI 10.1109/ACCESS.2020.2993233.

[21] Lee, Ritchie, Mengshoel, Ole J and Kochenderfer, Mykel J. "Adaptive stress testing of safety-critical systems." *Safe, Autonomous and Intelligent Vehicles*. Springer (2019): pp. 77–95. DOI 10.1007/978-3-319-97301-2_5.

[22] Kong, Zelun, Guo, Junfeng, Li, Ang and Liu, Cong. "Physgan: Generating physical-world-resilient adversarial examples for autonomous driving." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*: pp. 14254–14263. 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Kong_PhysGAN_Generating_Physical-World-Resilient_Adversarial_Examples_for_Autonomous_Driving_CVPR_2020_paper.html.

[23] Stone, Robert B, Tumer, Irem Y and Stock, Michael E. "Linking product functionality to historic failures to improve failure analysis in design." *Research in Engineering design* Vol. 16 No. 1 (2005): pp. 96–108. DOI 10.1007/s00163-005-0005-z.

[24] Oman, Sarah, Koch, Michael, Tumer, Irem Y and Bohm, Matt. "Verifying the usability of failure-based computational design methods." *ASME International Mechanical Engineering Congress and Exposition*, Vol. 44489: pp. 329–337. 2010. DOI 10.1115/IMECE2010-39259.

[25] Budynas, Richard Gordon, Nisbett, J Keith et al. *Shigley's mechanical engineering design*. Vol. 9. McGraw-Hill New York (2011).

[26] Bluvband, Zigmund and Grabov, Pavel. "Failure analysis of FMEA." *2009 Annual Reliability and Maintainability Symposium*: pp. 344–347. 2009. IEEE. DOI 10.1109/RAMS.2009.4914700.

[27] Montgomery, Thomas A, Pugh, David Richard, Leedham, Steve T and Twitchett, Steve R. "FMEA automation for the complete design process." *Proceedings of 1996 Annual Reliability and Maintainability Symposium*: pp. 30–36. 1996. IEEE. DOI 10.1109/RAMS.1996.500638.

[28] Price, Chris J and Taylor, Neil S. "Automated multiple failure FMEA." *Reliability Engineering & System Safety* Vol. 76 No. 1 (2002): pp. 1–10. DOI 10.1016/S0951-8320(01)00136-3.

[29] Hulse, Daniel, Hoyle, Christopher, Tumer, Irem Y, Goebel, Kai and Kulkarni, Chetan. "Temporal Fault Injection Considerations in Resilience Quantification." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 84003: p. V11AT11A040. 2020. American Society of Mechanical Engineers. DOI 10.1115/DETC2020-22154.

[30] Wang, Jing, Zuo, Wangda, Rhode-Barbarigos, Landolf, Lu, Xing, Wang, Jianhui and Lin, Yanling. "Literature review on modeling and simulation of energy infrastructures from a resilience perspective." *Reliability Engineering & System Safety* Vol. 183 (2019): pp. 360–373. DOI 10.1016/j.ress.2018.11.029.

[31] Zhang, Xiaodong, Miller-Hooks, Elise and Denny, Kevin. "Assessing the role of network topology in transportation network resilience." *Journal of transport geography* Vol. 46 (2015): pp. 35–45. DOI 10.1016/j.jtrangeo.2015.05.006.

[32] Morozov, Andrey, Ding, Kai, Steurer, Mikael and Janschek, Klaus. "Openerrorpro: A new tool for stochastic model-based reliability and resilience analysis." *2019 IEEE 30th international symposium on software reliability engineering (issre)*: pp. 303–312. 2019. IEEE. DOI 10.1109/ISSRE.2019.00038.

[33] Leveson, Nancy, Dulac, Nicolas, Zipkin, David, Cutcher-Gershenfeld, Joel, Carroll, John and Barrett, Betty. "Engineering resilience into safety-critical systems." *Resilience engineering*. CRC Press (2017): pp. 95–123. DOI 10.1201/9781315605685-12.

[34] Links, Jonathan M, Schwartz, Brian S, Lin, Sen, Kanarek, Norma, Mitrani-Reiser, Judith, Sell, Tara Kirk, Watson, Crystal R, Ward, Doug, Slemp, Cathy, Burhans, Robert et al. "COPEWELL: a conceptual framework and system dynamics model for predicting community functioning and resilience after disasters." *Disaster medicine and public health preparedness* Vol. 12 No. 1 (2018): pp. 127–137. DOI 10.1017/dmp.2017.39.

[35] Miller-Hooks, Elise, Zhang, Xiaodong and Faturechi, Reza. "Measuring and maximizing resilience of freight transportation networks." *Computers & Operations Research* Vol. 39 No. 7 (2012): pp. 1633–1643. DOI 10.1016/j.cor.2011.09.017.

[36] Kurtoglu, Tolga and Tumer, Irem Y. "A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems." *Journal of Mechanical Design* Vol. 130 No. 5. DOI 10.1115/1.2885181. 051401.

[37] McIntire, Matthew G, Keshavarzi, Elham, Tumer, Irem Y and Hoyle, Christopher. "Functional models with inherent behavior: Towards a framework for safety analysis early in the design of complex systems." *ASME International Mechanical Engineering Congress and Exposition*, Vol. 50657: p. V011T15A035. 2016. American Society of Mechanical Engineers. DOI 10.1115/IMECE2016-67040.

[38] Krus, Daniel and Lough, Katie Grantham. "Function-based failure propagation for conceptual design." *AI EDAM* Vol. 23 No. 4 (2009): pp. 409–426. DOI 10.1017/S0890060409000158.

[39] Hughes, Nigel, Chou, Enxi, Price, Chris J, Lee, Mark H et al. "Automating Mechanical FMEA Using Functional Models." *FLAIRS Conference*: pp. 394–398. 1999. URL https://www.aaai.org/Papers/FLAIRS/1999/FLAIRS99-071.pdf.

[40] Hulse, Daniel, Walsh, Hannah, Dong, Andy, Hoyle, Christopher, Tumer, Irem, Kulkarni, Chetan and Goebel, Kai. "fmdtools: A Fault Propagation Toolkit for Resilience Assessment in Early Design." *International Journal of Prognostics and Health Management* Vol. 12 No. 3. DOI 10.36001/ijphm.2021.v12i3.2954.

[41] Allspaw, John. "Fault injection in production." *Communications of the ACM* Vol. 55 No. 10 (2012): pp. 48–52. DOI 10.1145/2346916.2353017.

[42] Natella, Roberto, Cotroneo, Domenico and Madeira, Henrique S. "Assessing dependability with software fault injection: A survey." *ACM Computing Surveys (CSUR)* Vol. 48 No. 3 (2016): pp. 1–55. DOI 10.1145/2841425.

[43] Soyturk, Muhammet Abdullah, Parasyris, Konstantinos, Salami, Behzad, Unsal, Osman, Yalcin, Gulay and Gomez, Leonardo Bautista. "Hardware Versus Software Fault Injection of Modern Undervolted SRAMs." *arXiv preprint arXiv:1912.00154* DOI 10.48550/arXiv.1912.00154.

[44] Goldstein, Brunno, Srinivasan, Sudarshan, Mellempudi, Naveen K, Das, Dipankar, Santiago, Leandro, Ferreira, Victor C., Solon, N., Kundu, Sandip and França, Felipe M. G. "Reliability Evaluation of Compressed Deep Learning Models." *2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS)*: pp. 1–5. 2020. DOI 10.1109/LASCAS45839.2020.9069026.

[45] Georgakoudis, Giorgis, Laguna, Ignacio, Vandierendonck, Hans, Nikolopoulos, Dimitrios S and Schulz, Martin. "Safire: Scalable and accurate fault injection for parallel multithreaded applications." *2019 ieee international parallel and distributed processing symposium (ipdps)*: pp. 890–899. 2019. IEEE. DOI 10.1109/IPDPS.2019.00097.

[46] Engelmann, Christian and Naughton, Thomas. "Toward a performance/resilience tool for hardware/software co-design of high-performance computing systems." *2013 42nd International Conference on Parallel Processing*: pp. 960–969. 2013. IEEE. DOI 10.1109/ICPP.2013.114.

[47] Martins, Rolando, Gandhi, Rajeev, Narasimhan, Priya, Pertet, Soila, Casimiro, António, Kreutz, Diego and Veríssimo, Paulo. "Experiences with fault-injection in a Byzantine fault-tolerant protocol." *Acm/ifip/usenix international conference on distributed systems platforms and open distributed processing*: pp. 41–61. 2013. Springer. DOI 10.1007/978-3-642-45065-5_3.

[48] Zhang, Long, Morin, Brice, Haller, Philipp, Baudry, Benoit and Monperrus, Martin. "A chaos engineering system for live analysis and falsification of exception-handling in the JVM." *IEEE Transactions on Software Engineering* DOI 10.1109/TSE.2019.2954871.

[49] Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, Xu, Xiaowei et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." *kdd*, Vol. 96. 34: pp. 226–231. 1996. URL https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

[50] Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent et al. "Scikit-learn: Machine learning in Python." *the Journal of machine Learning research* Vol. 12 (2011): pp. 2825–2830. URL https://www.jmlr.org/papers/v12/pedregosa11a.html.