ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC 2018
August 26-29, 2018, Quebec City, Quebec

DETC2018-86049

EXPLORING SYSTEM ARCHITECTURE ATTRIBUTES AND DYNAMIC CHANGE PROPAGATION

Daniel Long
Graduate Research Assistant
North Carolina State University
Raleigh, NC, USA
delong@ncsu.edu

Dr. Scott Ferguson[1]
Associate Professor
North Carolina State University
Raleigh, NC, USA
scott_ferguson@ncsu.edu

ABSTRACT

Long-lived systems will experience many successive changes during their lifecycle as they are adapted to meet new system requirements. Existing change propagation tools predict how changes to a system's design at a fixed point in its life are likely to spread, but have not been extended to consider a series of successive modifications where the change probabilities update. This change in propagation probabilities in response to successive changes is introduced as Dynamic Change Propagation (DCP). This paper integrates research from change propagation, network theory, and excess to achieve the following objectives: 1) describe how a DCP model predicts system propagation change trajectories, 2) use a new synthetic test case generator to correlate network parameters like degree distribution with DCP, and 3) determine the correlations between a measure of DCP and a selection of existing change propagation metrics. Results indicate that DCP is limited by reducing the number of dependencies between components (affirming the usefulness of adding modularity to a system) and including high degree component 'hubs' between components.

1.0 INTRODUCTION

The choice to modify a system after it has been put into service depends on the amount of effort required to meet new requirements when they arise [1]. If a system cannot be changed with acceptable effort, the system is used as-is (with diminished value) or the system is retired. One of the phenomena that makes changing a system challenging is the tendency for one change to propagate, leading to many other changes. Existing change propagation research has generally focused on propagation with respect to the initial system design. Rarely is the impact of successive changes considered. If each change alters the change propagation pathways within the system, as research into excess and evolvability suggests [2], then it would be beneficial to model alterations in change propagation. This research explores Dynamic Change Propagation (DCP) and quantifies its consequences. Predictions made by existing change propagation techniques are then compared to the results of the DCP analysis.

A variety of strategies have been employed to understand how changes impact a system. Architecture based methods include the Change Propagation Method (CPM) [8], Design for Variety (DfV) [9], and network theory approaches [10] are some examples. An implicit assumption of these approaches is that the change propagation probabilities and connections are static. Changes are analyzed in relative isolation and the collective impact of many changes over time is not addressed. This assumption is reasonable for systems where context changes slowly relative to the in-service period (like roads, homes) or that are relatively inexpensive to replace (like keyboards or phones). It may be inappropriate for expensive systems and long-lived systems.

Limitations of the static system assumption have been highlighted by existing work. It has been acknowledged that "Avalanches occur when unexpected change multipliers are encountered or when change margins of known multipliers are used up." [11] and "Loops could be included in the analysis to allow the prediction of additional changes to the initiating systems." [8] The types of changes considered in this research are those driven by external sources. Fricke and Schulz [3] identified three primary categories of change drivers. These are: technology evolution, a dynamic environment (changes in customer preferences), and variety of environments (the system-of-systems ecosphere in which the system operates). An initial system design is performed with a known set of technologies, customer preferences, and interactions with other systems (referred to collectively as the initial context). This context is built into the system's requirements.

As time passes the drivers identified by Fricke and Schulz change, altering the system's value. This phenomenon is increasingly impactful because the pace of context change has accelerated. This has occurred in tandem with extended system lifecycles due to increased cost and development effort [4].

[1]Corresponding Author

Nuclear power plants with 30 year operating licenses are being renewed for 20 and 40 additional years of operation [5] and aircraft like the B-52 and C-130 have expected service periods of 80-100 years [6,7].

A poignant example of this is the design of the original F/A-18 Hornet. As discussed in Long and Ferguson [12], the Hornet was designed with features like a multiplex bus [13] that were intended to reduce the cost of future changes. The initial design did not provide sufficient system resources to accommodate the addition of multiple systems over time. Boeing and the Department of Defense quickly recognized that the existing airframe would soon be unable to support future desired modifications after a comparably short operational period, resulting in a costly design refresh [14].

This example demonstrates how changes made to a system consume excess which in turn erodes a system's ability to be changed again. This is an issue that has not been sufficiently addressed in existing literature and can have severe unanticipated consequences. This research is an initial step towards exploring and characterizing the impact that multiple changes over time have on a system. It begins with the assumption that excess consumption increases change probabilities. This research then uses existing change propagation methods to determine what impact those modified probabilities have on change propagation dynamics. The result is a preliminary understanding of the effect of excess without explicitly modeling it.

In addition, this research introduces a method found in other fields, but not yet used for engineering propagation research. The technique used for testing hypothesis in this research involves generating synthetic test cases using network properties found in real systems. Generation of synthetic cases allows researchers to test the impact that properties like clustering or degree-distribution have on DCP. This allows for rapid modeling and characterization of these properties with goodness measures for correlation testing.

2.0 BACKGROUND

This research falls under the scope of engineering change literature. Engineering change is a broad category of research that examines all aspects of system and design change from need identification, through project management, to documentation and tracking. An excellent overview of the entire scope of engineering change literature was recently compiled by Jarratt et al. [16]. By focusing on a system after it has been put into service, this work more specifically aligns with recent efforts to understand changeability; though the meanings of flexibility, adaptability, and changeability have yet to be settled [17–20]. Most pertinent is the definition provided by Mak and Clarkson [21] where flexibility is considered to be a change made by an agent external to the system whose degree "may be assessed by the ability of the system to be changed easily."

Several approaches have been proposed to understand what design decisions and system characteristics are favorable for flexibility. One category is general frameworks like Beesemyer et al. [22] that categorized "-ilities" and how they might be created in a system, or Olewnik and Lewis [23] who discussed the application of multi-objective optimization and utility theory as being central to flexibility. Another category is more heuristic based approaches like that of Tilstra et al. [24] where a large collection of flexible systems were studied and 24 guidelines were identified to which many of the systems adhered.

Ulrich [25] provides one of the earliest, and clearest, arguments for an architecture-based approach to engineering design. A system architecture is described as the assignment of functions to components, and the assignment of components to each other. System architectures were identified as a key to achieving product variety, product change, and competitive product performance.

A focus on system architecture leads to the exploration of system change. The following discussion highlights a small collection of change propagation-based research, and a broader review can be found in [16].

Change Modes and Effects Analysis [26] is a novel take on the Failure Modes and Effects Analysis used to identify deficiencies in systems that may lead to failure. Fig. 1 shows the column headings of a table to be filled out on each component of a decomposed system. The table is populated by an expert or experts. It encourages discussion about how likely changes are to happen, what source they might come from, how difficult it would be to change the system, and what actions might be taken. The tool serves as a way to compare thoughts about flexibility, but lacks sufficient mathematical rigor to provide a solid quantitative footing [27].

Modules / Parts	Potential change modes	Potential effects of change	Design flexibility	Potential cause(s) of change	Occurrence	Readiness	Change of potential number	Recommended actions / Guidelines

Fig. 1 Partial CMEA table headings

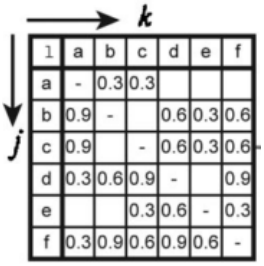
Design for Variety [9] uses two indices to help designers identify which components make good candidates for reuse across a family of products, and which components should be decoupled from the system as much as is feasible via modularization. The first index is the Coupling Index. The Coupling Index captures the strength of a connection between components to identify design dependencies. The second index is Generational Variety Index which identifies the extent to which engineering metrics, based on customer requirements, depend on specific components. Again, a numerical score is used to indicate the strength of the dependency. These scores are then compared with the various redesign costs to identify components for standardization (limit changes across generations) and for modularization (limiting the change propagation form the component).

This approach does have limitations. The Coupling Index, for example, only considers 1 step change propagation. In real systems change can propagate along several design dependencies. This causes change to components without a direct connection to the initiating component. This method also

does not explicitly consider the impact of successive changes over time and how those successive changes might result in a more severe impact than expected.

This research primarily builds upon the Change Propagation Method (CPM) as outlined by Clarkson et al. [8]. CPM was developed while studying the effects of change on Westland Helicopters. The approach begins by modeling component connections. Connections are design dependencies along which changes can propagate. They do not necessarily need to be in direct physical contact.

Direct change likelihoods (the direct probability that a change in one component will cause a change in another) and direct change impacts (the direct amount of rework required if change to one component necessitates a change in another) are then assessed for each connection by expert opinion in a similar manner as was done in CMEA. Each dependency is rated on a scale from 0 to 1 with 0 being no probability of propagation and 1 being a certainty. The resulting is shown in Fig. 2. Where the k components are those initiating change and the j components are being changed.



1	a	b	c	d	e	f
a	-	0.3	0.3			
b	0.9	-		0.6	0.3	0.6
c	0.9		-	0.6	0.3	0.6
d	0.3	0.6	0.9	-		0.9
e			0.3	0.6	-	0.3
f	0.3	0.9	0.6	0.9	0.6	-

Fig. 2 Direct likelihood matrix [8]

Once this direct likelihood matrix has been generated it is used to calculate a combined likelihood score. The examples and figures below are adapted directly from [8]. The combined likelihood is the probability that a change will propagate to other components when considering propagation beyond the direct interaction. To calculate this, two components are selected. The various pathways that exist between the two components (a and b in this example) are enumerated as shown in Fig. 3.

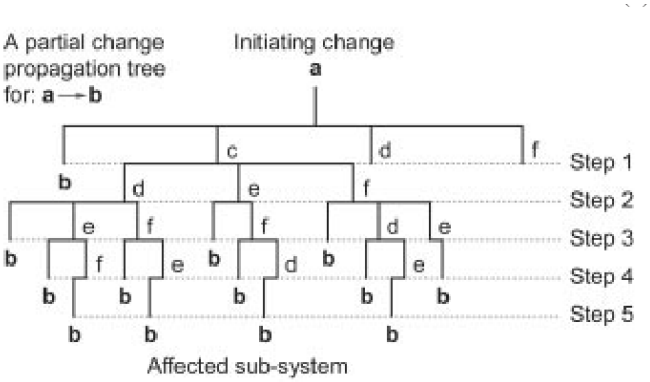
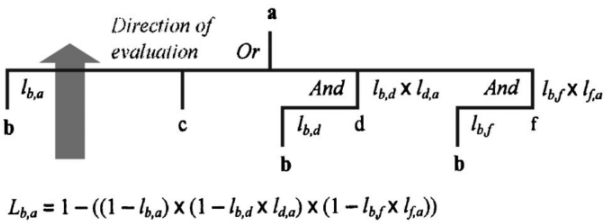


Fig. 3 Change propagation pathway tree [8]

Following pathway enumeration, the probability tree is evaluated from bottom to top by adding and multiplying propagation probabilities until a single probability is calculated at the top level as shown in Fig. 4. This value is the combined likelihood for the two components. The process is repeated for each cell in the matrix except diagonals and the resulting matrix is shown in Fig. 5.



$$L_{b,a} = 1 - ((1 - l_{b,a}) \times (1 - l_{b,d} \times l_{d,a}) \times (1 - l_{b,f} \times l_{f,a}))$$

Fig. 4 Pathway probability And/Or summation [8]

CPM resolves a limitation from GVI by allowing change to propagate across several components. A refinement to CPM as introduced by Koh et al. [28] is reachability. Reachability is the decay in the probability that change will propagate between components as a function of the number of steps away from the original change initiator the change gets. This captures behavior validated by system experts and a case study involving observations of real change propagation processes [29]. However, like GVI, it is unable to model successive changes.

L	a	b	c	d	e	f
a	-	0.6	0.1	1.0	0.1	
b	0.8	-	0.8	0.3	0.9	0.1
c	0.9	0.4	-	0.1	0.7	0.8
d		0.9	0.5	-	0.7	0.6
e	0.4		0.3	0.4	-	0.3
f	0.1	0.3	0.9	0.3	0.2	-

Fig. 5 Combined likelihood matrix [8]

A more recent approach to system modeling leverages advancements in graph and network theory [30,31]. For system design it is more often used when modeling network resilience to failures [32,33]. However, there has been some crossover into the domain of modularity [10]. Graph and network modeling is useful because it provides a less subjective metric for evaluating system flexibility and allows for tools that have been developed for other domains to be used in engineering system design. Knowledge from existing research on system design can be used to generate realistic synthetic test cases. Testing these realistic synthetic cases will allow for a statistically significant sample to discern what relationships exist between network properties (like clustering) and change propagation.

An additional approach to modeling change is that of excess and evolvability. This research [34] examines how system attributes could be intentionally over-designed (excess) in such a way as to enhance the system's ability to be changed over time (evolvability). Modifications consume the available excess but

the system gains additional value from the modification. Watson et el. [35] demonstrated a method for optimizing the trade between the cost of adding evolvability and the value added by it on a military truck design. Most recently work from Cansler [36] and White [37] have started to explore how the concepts of excess can be applied on a component level. These advancements provide a stronger link to existing change propagation research.

This work continues the integration of excess and evolvability research with change propagation research. We draw on the insights from network and graph theory to generate test cases. These cases are then subjected to increases in change probability likelihoods, mimicking the consequences of excess consumption, so the impact the modification has on a system's continued flexibility can be explored.

3.0 DYNAMIC CHANGE PROPAGATION MODELING

The goal of DCP is to measure the impact that accumulated changes have on a system over time. This section outlines the general framework for modeling DCP. A graphical overview of one iteration of the procedure is presented in Fig. 6.

We create a direct likelihood DSM in the first step. A random component is then selected as the change initiator for this iteration. A sample of dependent components modified by direct propagation are then identified. These components are added to the modified list (these are components B and E in Fig. 6). Change is then further propagated from B and E until no further change pathways are available. This is similar to the CPM method shown in Figs. 3 and 4. Finally, the updated probabilities obtained from the CPM process are used as the DSM value for the next iteration. After the change probabilities are updated, a system cascade score is calculated and stored for later analysis.

In more detail the four steps necessary for simulation are:

1) **System Definition –** Set up a system representation. DCP modeling adopts the existing direct likelihood matrix from CPM.

Steps two, three, and four are performed iteratively until the simulation converges (generally when all change probabilities in the DSM are equal to 1).

2) **Propagation Simulation –** This step stochastically models the effects of change propagation in a system. This is completed by randomly selecting a component as the change initiator and probabilistically determining the other components affected by the change.

3) **Propagation Probability Modification –** The probabilities of change for all design dependencies of components that are modified in Step 2 are increased by a predefined amount.

4) **Cascade Score Calculation –** The metric used to score DCP impact is how many changes, on average, are caused by propagation from a single change. This is referred to as a "cascade score" as it measures how much a change cascades throughout the system. After each iteration, the cascade score is calculated for the new system and convergence is assessed. If the simulation has not converged, another iteration is completed.

These four steps are discussed in the proceeding sections.

3.1 System Definition

This representation uses Design Structure Matrices to store the relationships between components. The DSM is compact, mature, and widely adopted in literature. An example of a direct

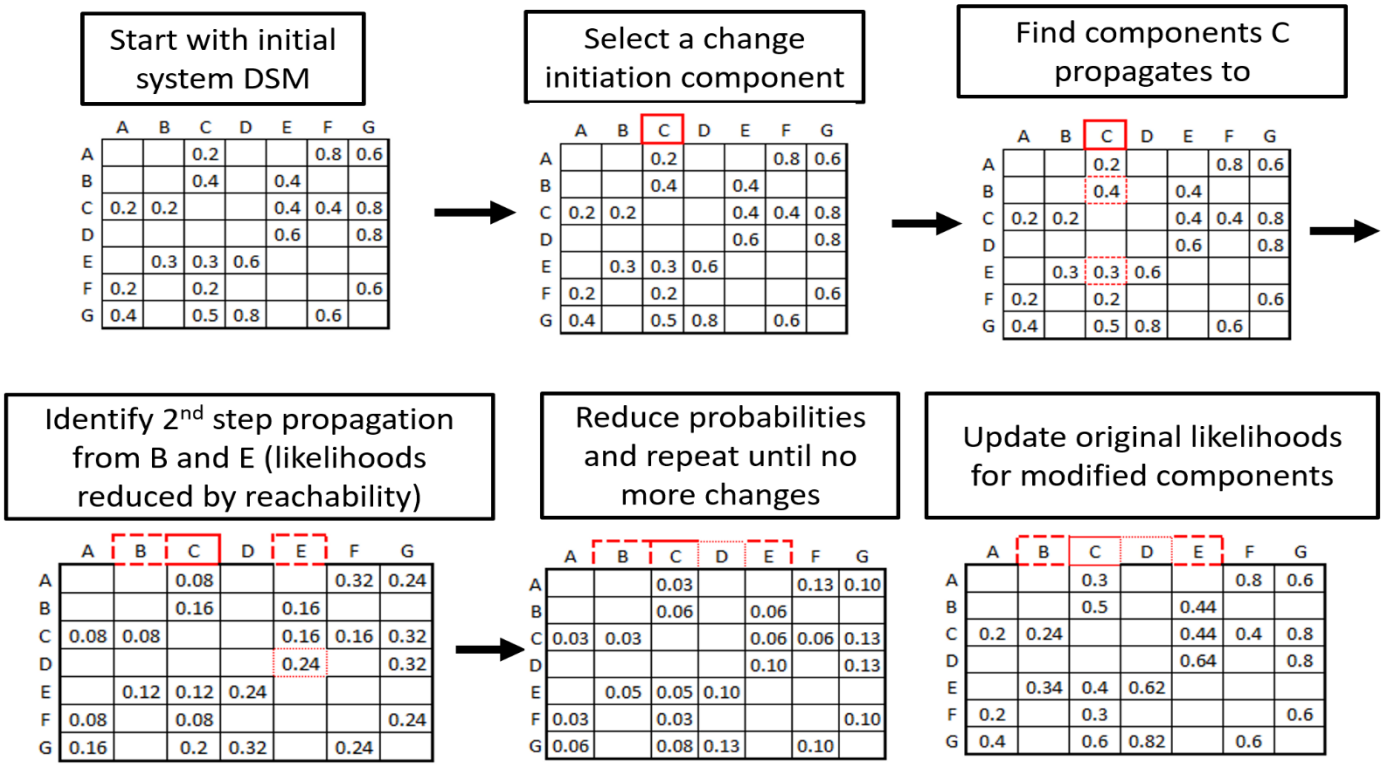


Fig. 6 A graphical illustration of the DCP Modeling

likelihood matrix from CPM is shown in Fig. 2. Along the top are the components initiating change, along the left are components that are at risk of being changed in response. Each entry in the matrix is the probability of that change occurring. In addition to the benefits of the DSM structure, placing probabilities in the matrix is both intuitive and convenient for probabilistic methods without requiring normalization.

In CPM the probability values are generated from expert opinion, but other techniques have been proposed for deriving dependencies including: the number and strength of connections [9], the number and type of connections [38], or historical data [29].

3.2 Propagation Simulation

Once a DSM has been constructed, the second step involves simulating change propagation in the system. This is done by sampling DCP trajectories.

The first step of sampling a DCP trajectories is selecting a component from which change will propagate (the change initiator). In this work, the change initiator is selected uniformly from all components. Other approaches could use knowledge about planned changes, as in Koh et al. [28], or predictions about the rate of change for constitutive components to generate a more representative distribution for a particular system.

Once a change initiator is selected the components that depend on the change initiator are assessed to see if they too change. A uniform distribution is sampled for each probability in the change initiator column. If the results are smaller than the value in the cell, the component in the associated row (the j component from Fig. 2) is marked as changed. After each dependent component has been tested, all propagation probabilities on the DSM are multiplied by a reachability factor of 0.4. The reachability factor represents the diminishing likelihood of change to propagate through components as discussed in Koh et al [28]. A value of 0.4 limits the reachability of change propagation beyond four steps to less than 1%.

Since multiple components may have been affected by the first change, this procedure is performed on each affected component. A simplification made in this step is that a component may not be marked as changed more than once. This assumption precludes the existence of design loops. This procedure repeats until there are no more components marked as changed.

After the procedure is completed the change probabilities are modified as described in the next section. The newly modified network is scored using the cascade score described in Section 3.4. The propagation simulation is then repeated beginning with the selection of a new change initiator.

3.3 Propagation Probability Modification

Modifying propagation probabilities starts with a list of components that were changed in the last propagation simulation. The step at which the component was modified is also recorded, and the probability for change propagation is increased.

The algorithm increases change propagation probabilities for each affected dependent component by a fixed amount. The components impacted by the first propagation step are increased by the fixed amount times the reachability factor (0.4 by default). Components changed in subsequent steps are changed in a similar manner. This reduction is in recognition that the further away from change a component is, the smaller the impact is likely to be.

This algorithm is a reasonable starting point for an initial exploration into dynamic change propagation because the small increase in propagation probabilities have the same results as larger steps but with more iterations. The extra iterations help to fill out cascade trajectories for full exploration. When used on physical systems, the step size should be calibrated to the specific system.

3.4 Cascade Score Calculation

Change propagation metrics in literature generally focus on component level measures of change. This allows designers to understand which components, or component interactions, should be targeted for improvement. However, this obfuscates the response of the system in its entirety. This research examines the system-wide impact of change and therefore requires a global measure. The metric derived is a cascade score because it is a measure of how many changes to the system would occur, on average, if a change is initiated (including the initiating component). This provides a straightforward aggregate for the entire system that can be tracked as change probabilities are modified throughout the simulation.

The cascade score is calculated by approximating the mean number of changed components when the initiating components are drawn from a uniform probability. The same method is used as in Step 2, except that it is reset after each trial. The number of components that change in each trial is stored. This process continues until the mean value of the distribution is estimated within a specified tolerance of the sample with a 95% confidence interval.

It was discovered during analysis that the cascade score can also be derived from the combined likelihood matrix as developed in Clarkson et al. [8]. In essence, the combined likelihood table is a conditional probability table where the columns are the conditional probability of the dependent component changing if the initiating component changes. The sum of the column is then the expected number of changes that would occur. Averaging expected changes for each column provides the average number of changes if a random component initiates change (plus one for the initiating component).

After the cascade score is calculated, a check to determine if the DCP model has converged is performed. If it has converged, then the simulation is terminated. If convergence has not been achieved, the algorithm performs another iteration.

The cascade score is tracked through the entirety of the simulation. This score can be used to generate sample trajectory plots of *cascade score vs iteration*, as shown in Fig. 7.

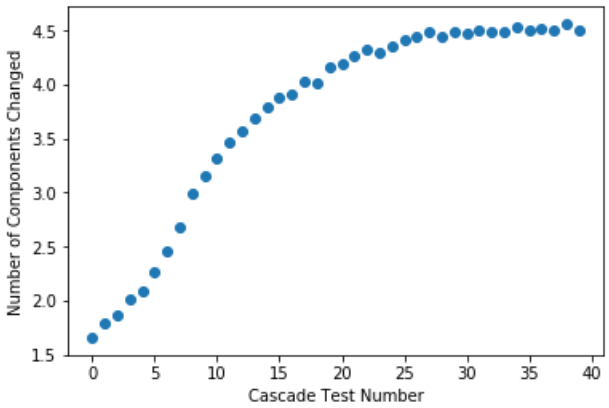


Fig. 7 A sample plot of changes vs. test iteration

Since the model is stochastic in nature, each result varies depending on the order in which components are selected each iteration. For this reason, the average of a sample of trajectories is taken to smooth out fluctuations.

To compare the averaged results from the DCP model against other system configurations it is necessary to devise a measure of overall goodness for a cascade trajectory. The approach taken is adapted from Smith [39]. Smith suggests the quantity of interest be integrated over the time period of interest to capture any dependency. To ensure the trajectory is scaled correctly for comparison the number of components should be normalized if the systems do not have the same number components. The propagation step axis should also be normalized to unity.

This provides a result in the range 0-1, where 0 is a system that exhibits no increase in cascade score. A 1 is a system in which the first change propagates to every component in the system. This measure is defined as the Area Under Curve (AUC). Calculating this metric is the final step in the DCP model.

4.0 SYNTHETIC TEST SYSTEM GENERATION

For this study, we choose to model a battery of synthetically generated test cases rather than testing individual systems represented in literature. This allows us to study how DCP is impacted by chosen network properties, such as degree distribution. This section covers the generation procedure for the test cases.

Test cases generated were desired to approximate real world systems. Complex network research provides a framework for generating edges that connect components. In network terminology, the nodes (components) are connected by directed edges (design dependencies), and the degree (number of connections a component has) distribution is the probability distribution for number of edges per node. Engineered systems have in-edges (information flows in from another component and out-edges (information flows from the component to a different component).

Strogatz [30] provides a thorough overview of different network topologies of interest in the sciences. One category of topology is the regular network. This type of network is generated with a repeating pattern of connection between nodes. An example would be a network in which each node is connected to its two closest neighbors and no others. The result is a ring.

Of more relevance to engineering design are complex network architectures that incorporate some randomness. The simplest such network is a random network in which nodes are randomly connected by *n* number of edges. As *n* grows the network begins to coalesce from isolated clusters to become ever larger.

Strogatz asserts that most real-world networks fall between the extremes of regular and completely random. One simple model generating networks in this middle ground is the small-world network [40]. This network is created by starting with a regular lattice network and rewire edges with a probability of ϕ. With a small number of rewired edges, the graphs drastically reduce the average shortest path length (a property of random graphs) but also retains the high degree of clustering (nodes tend to exist in well-connected clusters). This type of graph begins to approximate the behavior of real-world networks like social groups.

The final class of networks highlighted here are the scale-free networks. These networks have a few nodes with very high degree (many connecting edges) while most have low degree. When plotted the degree distribution takes on the shape of a power law or exponential distribution. Strogatz also asserts that many real-world systems are well modeled by scale-free networks.

For engineered systems specifically, Sosa et al. [32] have examined a number of systems from a quality perspective and discovered that their sample was consistent with a power-law distribution with cut-offs (decay at the tail of the probability distribution). Though their research only defined degree as the "number of other components (within the system) the component connects to" it is reasonable to assume that design dependencies would follow similar patterns.

$$p \leftarrow list[\alpha_1 ... \alpha_n] \text{ for } n = number\ of\ components$$

$$while\ E =< \max\_edges:$$

$$draw\ n\ from\ P = \left[\frac{\alpha_1}{\sum p}, \frac{\alpha_2}{\sum p}, ... \frac{\alpha_n}{\sum p} \right]$$

$$add\ edge\ to\ node\ n$$

$$\alpha_n = \alpha_n + 1$$

$$E = E + 1$$

Fig. 8 Algorithm for network generation

With this assumption, the artificial systems were generated by a modified form of the preferential attachment algorithm adapted from Barabasi and Albert [31]. Fig. 8 shows an outline of the algorithm used where α is the node weight, p is a list of all weights, and E is the total number of edges present. It is also enforced that no edge can be duplicated and that no component may have a self-edge.

	A	B	C	D	E	F	G	H	I	J
A			0.1		0.1	0.1	0.1			0.1
B			0.1					0.1	0.1	
C		0.1				0.1	0.1			0.1
D										
E						0.1		0.1		0.1
F			0.1		0.1					0.1
G				0.1	0.1			0.1		0.1
H			0.1		0.1				0.1	0.1
I		0.1	0.1			0.1		0.1		0.1
J			0.1		0.1		0.1	0.1		

Fig 9 Example network (n=10, E=35, α_{in}=2, α_{out}=2)

This procedure creates a random network like the one in Fig. 9. The parameter α can be modified to tune the degree of clustering desired. A low value of α results in a system where there are a few very high degree nodes and as $\alpha \rightarrow \infty$ the graph approaches a random graph. This allows for a variety of in and out degree distributions to be tested.

The final step in generating an artificial system is selecting the probabilities associated with each of the edges. For these studies all edges were set to a fixed value of 0.1. A fixed value allows the results to show primarily the impact that the configuration of edges has without needing to make additional assumptions, or perform additional analysis, to account for a non-uniform probability distribution. The low value allows the entire cascade trajectory for the system to be seen. Propagation probabilities will theoretically transit the entire region between 0.1 and 1 as they are increased with each change. A higher value would truncate the first part of the cascade trajectory and potentially hide interesting behavior.

5.0 RESEARCH METHODOLOGY

The objectives for the analysis portion of the research are as follows:

1) Describe how DCP models predict system response to increased change probability values.

This phenomenon has only been examined qualitatively in literature, so questions of interest include: To what level does change propagation increase in a system? At what rate does the increase happen? What shape does the curve take?

2) Test tunable parameters in the generative model to see what correlation exists between them and DCP.

Since a goal of modularity is to reduce edges in a system, it is hypothesized that this should have significant correlation with low AUC scores. It is also hypothesized that since systems tend to be approximately scale free, the degree distribution, tuned by the α_{in} and α_{out} generative parameters discussed in Section 5, should have a significant correlation with low AUC scores. A wider range of α values were tested because while systems tend towards a scale-free degree distribution, there is likely to be a great deal of variation. Testing a variety of distributions is more inclusive of the expected variation.

3) Test for correlation with selected existing change propagation metrics.

It is hypothesized that AUC scores should have good correlation with existing metrics since the two are testing related system properties.

To test these hypotheses a set of test cases were simulated with varying generative algorithm inputs.

- The number of components was set to 25 for each graph. The network is large enough system for reasonable differentiation of degree distributions. The network is also small enough that many samples could be simulated with available computational resources.
- The number of edges, in-degree α, and out-degree α were varied over a range of values shown in Table 1.
- Each combination of levels was simulated 15 times and the average of those 15 was output as the result for the combination.
- The number of simulations was truncated at 30 for simulations and 35 for initial testing. The truncation reduced computational effort and captured most of the behavior of interest.

Variable	Values
Edges	[25, 125, 250, 375]
α_{in}	[0.1, 0.5, 1, 10, 100]
α_{out}	[0.1, 0.5, 1, 10, 100]

Table 1 Levels used for generating test systems

6.0 RESULTS

The results from this analysis reveal several insights regarding the impact that system architecture properties have on the way change propagation behaves as probabilities are increased. The results are broken into the three categories outlined in the methods section.

6.1 SYSTEM RESPONSE TO DCP

The cascade score of a system increases with propagation probabilities, but levels off at a value that is generally less than the total number of components, as shown in Fig. 10 (a random selection of cascade trajectories for various parameters selected to illustrate the shapes and terminal values of cascade trajectories). This is a result of the interplay between the system architecture and the reachability of change propagation. Even if all probabilities are set to one, the odds of a change propagating beyond four steps is very low. This highlights the importance of the mean degree and average distance modularity [10] for the system.

Intuitively, more edges per node, i.e. a higher average degree, increases the number of components a change can potentially reach. This relationship is influenced by how the architecture is arranged. Propagation can by stymied in a high

degree system if that system has a higher out distance modularity. This occurs when a sub-group of components is tightly clustered together, but is a larger number of edges away from other clusters.

The rate of change follows an S-shaped curve. Some curves rise quickly and approach the maximum number of components while others rise much more slowly. The tuned parameters that are responsible for different observed behaviors are discussed in the following section.

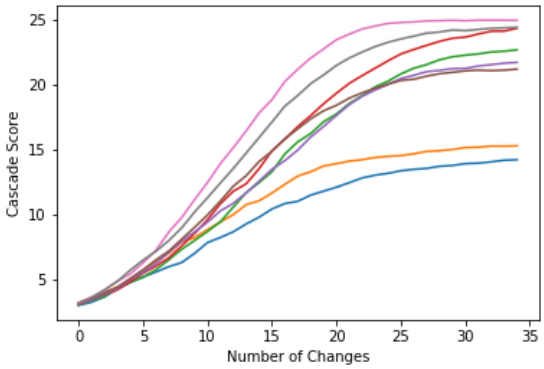


Fig. 10 A sample subset of cascade scores

6.2 Influence of Tuned Parameters

Results of the influence of the tuned parameters on the cascade scores are shown in this section. The simulations that were run with 25 edges (with an average degree of one) increased so slowly that the simulation met convergence criteria well before other simulations. These simulations have been left in the charts and appear in the bottom section of each figure.

The number of edges has the most pronounced effect on the shape of cascade trajectory. Fig. 11 shows that as the number of edges increases, the rate of increase in the cascade trajectory is larger and tends towards the maximum of 25.

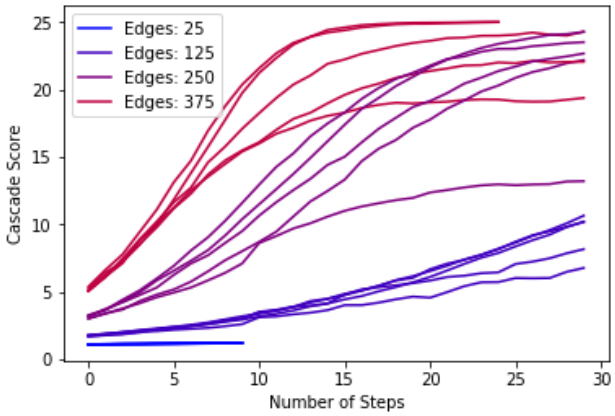


Fig. 11 Effect of edges on cascade scores: α_{in}= 1, α_{out} = [0.1-100]

The α_{out} parameter provides a measure of the degree distribution for the out-degree of components. A low value results in a few components with most of the outward-directed edges. As α_{out} increases the graph approaches a random distribution. Fig. 12 shows that this parameter has significant influence. The clusters of different edge values can still be seen, but within each cluster α_{out} correlates with where in the cluster the specific simulation falls. The higher α_{out}, the less skewed the distribution and the more quickly the cascade score increases and the higher its maximum value.

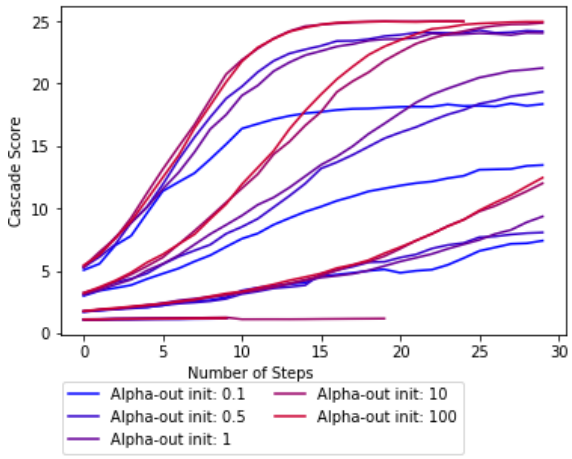


Fig. 12 α_{out} effect on Cascade Scores: Edges=[25-375], α_{in}=1

This implies that holding number of edges constant, a network with more outgoing connections from a single component, experiences less increase in the cascade score than a network with a more even degree distribution.

This is intuitively reasonable. The few components with a high-out degree are not likely to have more in-degree connections than any other component. When selecting for initiating components at random it is therefore less likely that the few nodes with many connections will be selected.

A notable exception to this would be the case in which a component or subset of components has both high out-degree and high in-degree. In this case any change made to the system is likely to travel to the high degree component and propagate out to its large number of dependent components.

The final parameter, α_{in}, was found to have less influence when compared to number of edges or out degree distribution.

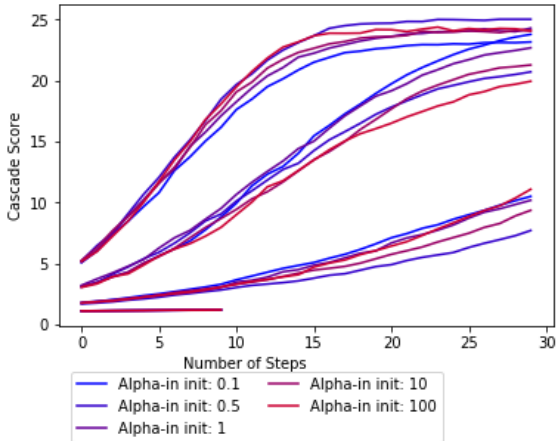


Fig. 13 α_{in} effect on Cascade Scores: Edges = [25-375], α_{out}=1

When α_{out} is low its influence dominates the position of cascade trajectories as seen Fig. 13. α_{in} has no discernable pattern.

Only when the α_{out} parameter is high (having little impact) does the effect of α_{in} appear. A more skewed distribution for in-degree correlates with the property of being more resistant to DCP, while a more uniform assignment of in-degree increases DCP.

The correlation coefficients for the three tuned parameters are reported in Table 2. These and all correlations provided are the Pearson correlation coefficient [39]. As hypothesized, the number of edges has a large effect on the AUC. Less expected is the weaker correlation between degree distribution and AUC. The out-degree distribution has a small effect using the Pearson correlation coefficient while the in-degree is small enough to be classified as no discernable effect.

	AUC
In Degree Initial Value	0.03
Out Degree Initial Value	0.12
Number of Edges	0.96

Table 2 AUC vs Network Generative Parameters

6.3 Correlation with existing metrics

The third objective was to determine how well existing metrics correlate with dynamic change propagation. Four existing metrics were tested: the initial system average CPM score [8], a system average weighted CPI (adapted from [41]), and two metrics from Sosa et al. [10].

The initial CPM score used for this correlation is the average of the sum for each column in the combined likelihood matrix. The correlation between CPM and AUC is large with a coefficient of 0.96. Fig. 14 shows a scatter plot that reveals a distinct trend where greater scatter is seen with larger CPM values. The clusters of different numbers of edges can be distinctly observed.

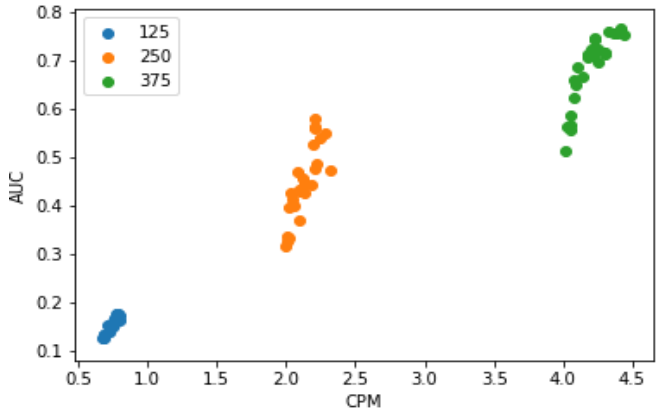


Fig. 14 CPM vs AUC with edge colors

The CPI measure used for comparison is weighted by the weight of the edge instead of the sum of number of edges as derived in [29]. The CPI for each component is calculate using:

$$CPI_i = \frac{W_{i,ji\neq j} - W_{j,ii\neq j}}{W_{i,ji\neq j} + W_{j,ii\neq j}} \quad (1)$$

where i and j are components and W is the weight of the edge of connecting i and j. The system score is then taken as the average of all components. The result is a small correlation with a coefficient of 0.20. Fig. 15 shows that at smaller edge numbers the CPI has little effect, but as the number of edges grows average CPI begins to correlate with AUC.

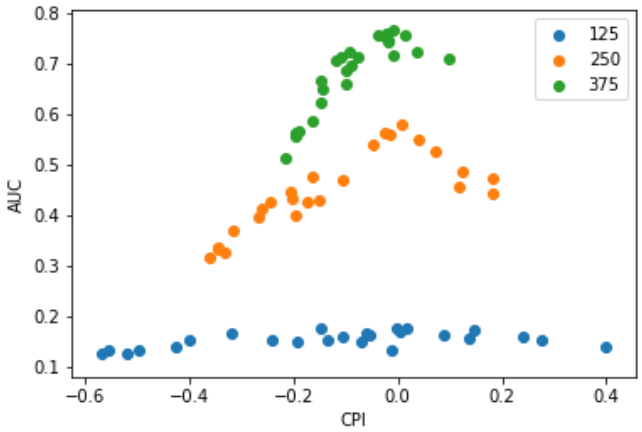


Fig. 15 CPI vs AUC with edge colors

Two metrics from Sosa et al. used for comparison are in-distance modularity and out-distance modularity. The distance metrics give insight into how closely connected the system is. The out-distance modularity and in-distance modularity both have large inverse effects on the AUC with correlation coefficients of -0.56 and -0.63 respectively These values are inversely correlated because the modularity score goes up as distance increases.

Fig. 16 shows the effect of out-distance modularity on AUC and reveals again that with a lower number of edges the score isn't impactful, but as number of edges increases a higher out-distance modularity score is correlated with a decline in AUC.

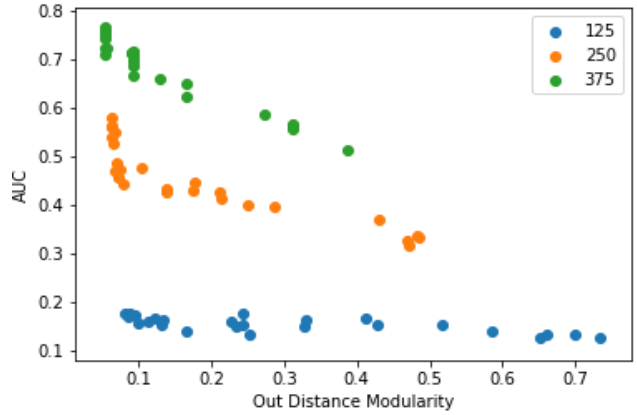


Fig. 16 Out-Distance Modularity vs. AUC with edge colors

7.0 DISCUSSION

There are two principle contributions to the field of engineering design in this research. These are:

1. Using artificial network representations of systems to test hypotheses about the impact of network structure on properties important to designers.

2. A comparison of existing flexibility metrics and system properties for predicting the evolution of change propagation within a system over time

Regarding the first contribution, the use of generated networks for testing system properties holds promise for learning about the influence those properties have on change propagation. Generated networks have already been used to test network structure for resilience to malicious attack, to see how failures might propagate in a system, and for system models in many other fields. Change propagation knowledge has reached a sufficient level of maturity synthetic test cases can now be used to supplement existing research and provide initial hypothesis test before spending resources for testing on real world systems.

The most significant finding is that keeping the number of edges low is the most effective way to prevent change cascades from worsening when the number of components is fixed. This finding supports existing research by affirming the value of modularity in minimizing connecting edges and thereby change propagation. A second finding is that the reachability and average component degree have competing effects on change propagation. Reachability limits how far change propagation travels, but a high average degree will mitigate this effect.

Of note is the observation that network hubs appear to help mitigate change propagation. This is true under the assumption that change starts with uniform probability at any component. It is theorized that if more edges are connected to a single component then change is more likely to propagate through that component to reach the rest of the system. Each step reduces the probability of propagation, so the extra step is likely to diminish the impact of change on the system. This assumption is supported by the observation that as the out-distance modularity increases, the AUC decreases indicating less change in the system. This is likely a useful technique for limiting change propagation, especially if the cost of modifying the hub component can be kept low.

The final finding is that existing change propagation metrics are correlated with improved DCP outcomes. The CPM score is the best correlated with outcomes, but also involves more effort than either CPI or network-based modularity metrics.

8.0 CONCLUSION AND FUTURE WORK

There is significant research to be completed, especially aimed at refining assumptions made during this preliminary study. Most significant would be the inclusion of cost as variable. This would allow for the modeling of observed change propagation phenomena that are dependent on the expense of a proposed change.

Also important is understanding the relationship between component excess and change probabilities. This research simplified the relationship to a fixed step in change propagation per change made. A more sophisticated model would link the use of excess by a change to the modification of change probability.

Finally, it has been observed that new change propagation pathways are created as system excess is consumed. For example, research into the F/A-18 revealed that internal system volume was depleted by component additions. It became necessary to miniaturize existing components. The miniaturized components were therefore altered only because of a lack of excess and not through any direct design dependency. Modeling this phenomenon could be possible with a more detailed accounting of system excess and would provide significant insight into how much excess should be included in the preliminary system.

In conclusion this research has established a foundation upon which to reduce the cost and unexpected redesign for long-lived systems. This is done by modeling how changing propagation pathways increase the number of components that must be modified when a change is made to the system. Incorporating model enhancements like component costs and system specific ties between change probabilities will allow designers to better understand how their system architecture impacts future lifecycle costs.

REFERENCES

[1] Schulz, A. P., Fricke, E., and Igenbergs, E., 2000, "Enabling Changes in Systems throughout the Entire Life-Cycle – Key to Success ?"

[2] Long, D., and Ferguson, S., "A CASE STUDY OF EVOLVABILITY AND EXCESS ON THE B-52 STRATOFORTRESS AND F/A-18 HORNET."

[3] Fricke, E., and Schulz, A. P., 2005, "Design for Changeability (DfC): Principles to Enable Changes in Systems throughout Their Entire Lifecycle," Syst. Eng., **8**(4).

[4] Bloebaum, C. L., and McGowan, A.-M. R., 2012, "The Design of Large-Scale Complex Engineered Systems: Present Challenges and Future Promise," AIAA Aviat. Technol. Integr. Oper. Conf. 14th AIAA/ISSM, (September), pp. 1–19.

[5] Voosen, P., 2009, "How Long Can a Nuclear Reactor Last? - Scientific American," Sci. Am. [Online]. Available: https://www.scientificamerican.com/article/nuclear-power-plant-aging-reactor-replacement-/. [Accessed: 03-Jan-2017].

[6] Swarts, P., 2016, "Air Force Prolongs the Life of the Venerable B-52," Air Force Times.

[7] Heisler, T., 2014, *C-130 Hercules : Background , Sustainment , Modernization , Issues for Congress*.

[8] Clarkson, P. J., Simons, C., and Eckert, C., 2004, "Predicting Change Propagation in Complex Design,"

ASME Des. Eng. Tech. Conf., **136**(August 2004), pp. 788–797.

[9] Martin, M. V, and Ishii, K., 2002, "Design for Variety: Developing Standardized and Modularized Product Platform Architectures," Res. Eng. Des., **13**, pp. 213–235.

[10] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2007, "A Network Approach to Define Modularity of Components in Complex Products," J. Mech. Des., **129**(11), p. 1118.

[11] Eckert, C., Clarkson, P. J., and Zanker, W., 2004, "Change and Customisation in Complex Engineering Domains," Res. Eng. Des., **15**(1), pp. 1–21.

[12] Long, D., and Ferguson, S., 2017, "A Case Study of Evolvability and Excess on the B-52 Stratofortress and F/A-18 Hornet," *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 1–11.

[13] Wieringa, J. A., 2003, "Spiral Development and the F/A-18: Parallels from the Past Emerge in Spiral Development of the F/A-18A through F Varients," Progr. Manag., pp. 50–52.

[14] Elward, B., 2012, *The Boeing F/A-18E/F Super Hornet & EA-18G Growler: A Develompental and Operational History*, Atglen: Schiffer Publishing Ltd.

[15] Hazelrigg, G. a., 1998, "A Framework for Decision-Based Engineering Design," J. Mech. Des., **120**(4), p. 653.

[16] Jarratt, T. A. W., Eckert, C. M., Caldwell, N. H. M., and Clarkson, P. J., 2011, "Engineering Change: An Overview and Perspective on the Literature," Res. Eng. Des., **22**(2), pp. 103–124.

[17] Saleh, J. H., Mark, G., and Jordan, N. C., 2009, "Flexibility: A Multi-Disciplinary Literature Review and a Research Agenda for Designing Flexible Engineering Systems," J. Eng. Des., **20**(3), pp. 307–323.

[18] Ferguson, S., Siddiqi, A., Lewis, K., and de Weck, O. L., 2007, "Flexible and Reconfigurable Systems: Nomenclature and Review," *Volume 6: 33rd Design Automation Conference, Parts A and B*, ASME, pp. 249–263.

[19] Ross, A. M., Rhodes, D. H., and Hastings, D. E., 2008, "Defining Changeability : Reconciling Flexibility , Adaptability , Scalability , Modifiability , and Robustness for Maintaining System Lifecycle Value," **11**(3), pp. 246–262.

[20] Ryan, E. T., Jacques, D. R., and Colombi, J. M., 2013, "An Ontological Framework for Clarifying Flexibility-Related Terminology via Literature Survey," Syst. Eng., **16**(1), pp. 99–110.

[21] Jonathan Mak, W. H., and John Clarkson, P., 2017, "Towards the Design of Resilient Large-Scale Engineering Systems," Procedia CIRP, **60**, pp. 536–541.

[22] Beesemyer, J., Fulcoly, D., Ross, A. M., and Rhodes, D. H., 2011, "Developing Methods to Design for Evolvability: Research Approach and Preliminary Design Principles," 9th Conf. Syst. Eng. Res., (April), pp. 1–12.

[23] Olewnik, A., and Lewis, K., 2006, "A Decision Support Framework for Flexible System Design," J. Eng. Des., **17**(1), pp. 75–97.

[24] Tilstra, A. H., Backlund, P. B., Seepersad, C. C., and Wood, K. L., 2015, "Principles for Designing Products with Flexibility for Future Evolution," Int. J. Mass Cust., **5**(1), pp. 22–54.

[25] Ulrich, K., 1995, "The Role of Product Architecture in the Manufacturing Firm," Res. Policy, **24**(3), pp. 419–440.

[26] Rajan, P. K. P., Van Wie, M., Campbell, M. I., Wood, K. L., and Otto, K. N., 2005, "An Empirical Foundation for Product Flexibility," Des. Stud., **26**(4), pp. 405–438.

[27] Olewnik, A., and Lewis, K., 2008, "Limitations of the House of Quality to Provide Quantitative Design Information," Int. J. Qual. Reliab. Manag., **25**(2), pp. 125–146.

[28] Koh, E. C. Y., Caldwell, N. H. M., and Clarkson, P. J., 2013, "A Technique to Assess the Changeability of Complex Engineering Systems," J. Eng. Des., **24**(7), pp. 477–498.

[29] Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., and Clarkson, P. J., 2009, "Change Propagation Analysis in Complex Technical Systems," ASME J Mech Des., **131**(August), pp. 1–14.

[30] Strogatz, S. H., 2001, "Exploring Complex Networks," Nature, **410**(6825), pp. 268–276.

[31] Barabási, A.-L., and Albert, R., 1999, "Emergence of Scaling in Random Networks," Science (80-.)., **286**, pp. 509–512.

[32] Sosa, M., Mihm, J., and Browning, T. R., 2011, "Degree Distribution and Quality in Complex Engineered Systems," J. Mech. Des., **133**(10), p. 101008.

[33] Mehrpouyan, H., Haley, B., Dong, A., Tumer, I. Y., and Hoyle, C., 2015, "Resiliency Analysis for Complex Engineered System Design," Artifcial Intell. Eng. Des. Anal. Manuf., **29**(1), pp. 93–108.

[34] Tackett, M. W. P., Mattson, C. A., and Ferguson, S. M., 2014, "A Model for Quantifying System Evolvability Based on Excess and Capacity," J. Mech. Des., **136**(5), p. 51002.

[35] Watson, J. D., Allen, J. D., Mattson, C. A., and Ferguson, S. M., 2016, "Optimization of Excess System Capability for Increased Evolvability," Struct. Multidiscip. Optim., **53**(6), pp. 1277–1294.

[36] Cansler, E. Z., White, S. B., Ferguson, S. M., and Mattson, C. A., 2016, "Excess Identification and Mapping in Engineered Systems," J. Mech. Des., **138**(8), p. 81103.

[37] White, S., and Ferguson, S., 2017, "Exploring Architecture Selection and System Evolvability," *Proceedings of the ASME 2017 International Design Engineering Technical Conferences and Computers and*

Information in Engineering Conference, pp. 1–16.
[38] Asikoglu, O., and Simpson, T., 2012, "A New Method for Evaluating Design Dependencies in Product Architectures," 12th AIAA Aviat. Technol. Integr. Oper. Conf. 14th AIAA/ISSMO Multidiscip. Anal. Optim. Conf., (September), pp. 1–11.
[39] Smith, R. C., 2013, *Uncertainty Quantification: Theory, Implementation, and Applications*, Siam.
[40] Watts, D. J., and Strogatz, S. H., 1998, "Collective Dynamics of 'small-World' Networks," Nature, **393**(6684), pp. 440–442.
[41] Suh, E. S., De Weck, O. L., and Chang, D., 2007, "Flexible Product Platforms: Framework and Case Study," Res. Eng. Des., **18**(2), pp. 67–89.