SYM-AM-18-109

PROCEEDINGS
OF THE
FIFTEENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

THURSDAY SESSIONS
VOLUME II

Acquisition Research:
Creating Synergy for Informed Change

May 9–10, 2018

March 30, 2018

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Assessing Vulnerabilities in Model-Centric Acquisition Programs Using Cause-Effect Mapping

Jack Reid—is a graduate student with the Systems Engineering Advancement Research Initiative (SEAri) at the Massachusetts Institute of Technology. Reid is earning master's degrees in both Aeronautics & Astronautics and Technology & Policy. His research interests concern the design and management of complex sociotechnical systems, particularly with regard to the anticipation of emergent and cascading behavior. He received a BS in Mechanical Engineering and a BA in Philosophy from Texas A&M University and has experience with RAND Corporation and Sandia National Laboratories. [jackreid@mit.edu]

Donna H. Rhodes—is a principal research scientist at the Massachusetts Institute of Technology, and director of the Systems Engineering Advancement Research Initiative (SEAri). Dr. Rhodes conducts research on innovative approaches and methods for architecting complex systems and enterprises, designing for uncertain futures, and human-model interaction. Previously, she held senior management positions at IBM, Lockheed Martin, and Lucent. Dr. Rhodes is a Past President and Fellow of the International Council on Systems Engineering (INCOSE), and INCOSE Founders Award recipient. She received her PhD in Systems Science from T. J. Watson School of Engineering at Binghamton University. [rhodes@mit.edu]

Abstract

Acquisition programs increasingly use model-centric approaches, generating and using digital assets throughout the lifecycle. Model-centric practices have matured, yet in spite of sound practices there are uncertainties that may impact programs over time. The emergent uncertainties (policy change, budget cuts, disruptive technologies, threats, changing demographics, etc.) and related programmatic decisions (e.g., staff cuts, reduced training hours) may lead to cascading vulnerabilities within model-centric acquisition programs, potentially jeopardizing program success. This paper presents ongoing research that seeks to provide program managers with the means to identify model-centric program vulnerabilities and determine where interventions can most effectively be taken. Cause-Effect Mapping (CEM), a technique developed at MIT, is employed to examine cascading effects between emerging perturbations and terminal outcomes. Research begins with literature investigation and gathering results of past studies of relevance, including studies of model-centric environments and transformations from a traditional to model-centric engineering paradigm (sometimes referred to as the digital engineering paradigm), recent workshop findings, and related work on vulnerability assessment that may have implications for this work. The results are used to refine the CEM and analytic approach to develop a reference model for vulnerability assessment of model-centric programs. Usability of the resulting model is tested with selected research stakeholders.

Introduction

In a world where engineered systems are rapidly increasing in complexity, scale, and interoperability, there is an urgency to transform traditional practices. Digital transformation changes how systems are acquired and developed through the use of model-centric engineering practices and toolsets. While offering great benefit, new challenges arise from both technological and socio-cultural dimensions. This drives the need to examine and address vulnerabilities not only for products and systems, but also for the model-centric environments necessary for their acquisition and development. Ongoing research investigates the use of Cause-Effect Mapping (CEM) as a mechanism for better enabling program managers and system engineers to anticipate and respond to programmatic vulnerabilities as related to model-centric environments. A Reference CEM for model-centric enterprises is generated based on gathered research findings and used for discussion with subject matter experts. Information on uncertainties and leading indicators is collected. Analysis is performed to consider the cascading vulnerabilities and potential intervention options.

Motivation

Acquisition program management is grounded in management science and a sound set of practices evolved over decades; however, new challenges arise as acquisition becomes increasingly model-centric. Increasing availability and use of model-centric approaches and enabling technologies is transforming engineering from documentation-centric to model-centric. While good practices have emerged to support the shift to model-centric program acquisition, such programs experience perturbations over their lifecycles that introduce new vulnerabilities that may lead to cascading failures. For instance, perturbations may be caused by policy change (leading to IP disagreements), economic factors (leading to training cuts), or disruptive technology (leading to outdated infrastructure). Early detection and intervention of vulnerabilities can mitigate disruptions and failures. The research seeks to contribute to the vulnerability assessment state of practice for acquisition programs, both public and private, that increasingly depend on digital assets and model-centric environments.

Background

The following subsections describe model-centric engineering; vulnerability, hazard and risk analysis; cause-effect mapping; and programmatic vulnerabilities.

Model-Centric Engineering

Acquisition program management is grounded in management science and a sound set of practices evolved over decades; however, new challenges arise as acquisition becomes increasingly model-centric. Baldwin and Lucero (2016) state, “The DoD sees value in adopting digital engineering design and model-centric practices, enabling a shift from the linear, document centric acquisition and engineering process toward a dynamic digital, model-centric ecosystem. ”

The systems engineering field is going through a period of significant transformation (Peterson, 2017). Advances in computing, digital workflows, and multidomain-multiscale models are leading to new concepts and approaches for model-centric engineering (Piaszczyk, 2011; Reid & Rhodes, 2016; Glaessgen & Stargel, 2012; Puckek et al., 2017; West & Pyster, 2017).

Model-Centric Engineering (MCE) has been defined as “an overarching digital engineering approach that integrates different model types with simulations, surrogates, systems and components at different levels of abstraction and fidelity across disciplines

throughout the lifecycle" (Blackburn et al., 2017). MCE involves using integrated models across disciplines, subsystems, lifecycle stages, and analyst groups. It uses models as the "source of truth" to reduce document handoff and allow for more continuous evaluation. This reduces communication time and rework in response to requirement changes. While many engineering organizations are applying various aspects of MCE (Glaessgen & Stargel, 2012; Kellner, 2015; Lockheed Martin, 2015), implementation is not without its difficulties. Enhanced infrastructure and new leadership capabilities are needed. Increased connectivity means the danger of improper access is heightened. Even with sound MCE practices in use, there are still many challenges that remain. Efforts are also ongoing to identify inconsistent policies in an organization using model-based tools (e.g., Krishnan, Virani, & Gasoto, 2017; Virani & Rust, 2016).

Most discussions of MCE focus on engineering practices and methods to overcome implementation difficulties. In any system, however, engineering is only a piece of the problem. Numerous human factors, business concerns, and organizational issues exist. The design and development of a system exists itself inside a sociotechnical system. Program managers and system engineers must learn how to identify and address programmatic vulnerabilities that pose threats to schedule and budget. Current program managers have significant experience with modern engineering processes. They can use this experience to identify and mitigate such vulnerabilities. Minimal experience exists specific to MCE and model-centric environments, however. This fact, coupled with the increased integration of models, means that emergent uncertainties (policy change, budget cuts, disruptive technologies, threats, changing demographics, etc.) and related programmatic decisions (e.g., staff cuts, reduced training hours) may lead to cascading vulnerabilities within MCE programs, potentially jeopardizing program success. New tools are needed to enable program managers to readily identify model-centric program vulnerabilities and determine where interventions can most effectively be taken.

The Defense Acquisition Glossary defines a *program* as "a directed, funded effort that provides a new, improved, or continuing materiel, weapon or information system, or service capability in response to an approved need." In this paper, this definition of *program* will be used. A *project*, on the other hand is acknowledged as being synonymous with program in general usage, but more specifically defined as "a planned undertaking having a finite beginning and ending, involving definition, development, production, and logistics support (LS) of a major weapon or weapon support system or systems. A project may be the whole or a part of a program" (Defense Acquisition University, n.d.).

Vulnerability, Risk, and Hazard Analysis

Numerous methods for analyzing vulnerabilities, risks, and hazards exist. These three interrelated terms have different definitions depending on the field and on the method of analysis. In this paper, a *hazard* refers to a system or environmental state that has the potential to disrupt the system. Examples include the existence of an iceberg at sea and tired operators. Hazards may not result in system failure, partly depending on the design of the system.

A *vulnerability* is the means by which the hazard might disrupt the system, thus it is through the vulnerability that the system is susceptible to the hazard. Vulnerabilities are best expressed as the causal series of events connecting a hazard to system failure. This is a generalization of common, field-specific usages of the term. MITRE's Common Vulnerabilities and Exposures (CVE) database defines a vulnerability as "a weakness in the computational logic (e.g., code) found in software and some hardware components (e.g., firmware) that, when exploited, results in a negative impact to confidentiality, integrity, OR availability" (The MITRE Corporation, 2015). In this definition, the same components can be

seen: some structural means or “weakness” that can result in system disruption or “negative impact” if a hazard is present or the vulnerability is “exploited.” For example, the infamous Spectre security vulnerability is described by CVE as “systems with microprocessors utilizing speculative execution and branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis” (The MITRE Corporation, 2017). This is a neat summary of the hazard (an attacker), the means (side-channel analysis using speculative execution and branch prediction), and the disruption (unauthorized disclosure of information).

Risk is a measure of the probability of a system disruption and the consequences of that disruption. It is commonly expressed with just a statement of those two components (e.g., 1.25 deaths per 100 million vehicle miles). Risk can also be expressed as a multiplication of likelihood and consequence and can include other components such as detectability.

Common means of analysis include Fault-Tree Analysis (FTA); Failure Modes, Effects, and Criticality Analysis (FMECA, though sometimes reduced to FMEA); Systems Theoretic Process Analysis (STPA); and Event Tree Analysis (ETA).

FTA is a deductive, top-down analysis method where a failure mode is identified and all the possible causes of that event are laid out in sequences until the exogenous hazards are reached. Logic gates are used to connect the various hazards and intermediary events. An example FTA may be seen in Figure 1. Probabilities may be assigned to each hazard and thus a cumulative probability of the failure calculated. FTA is thus quite proficient in investigating the cause of failures afterwards, but is limited in its ability to identify all possible hazards. Additionally, it is somewhat limited by its arbitrary stopping point (i.e., where one chooses to define an event as an exogenous hazard).

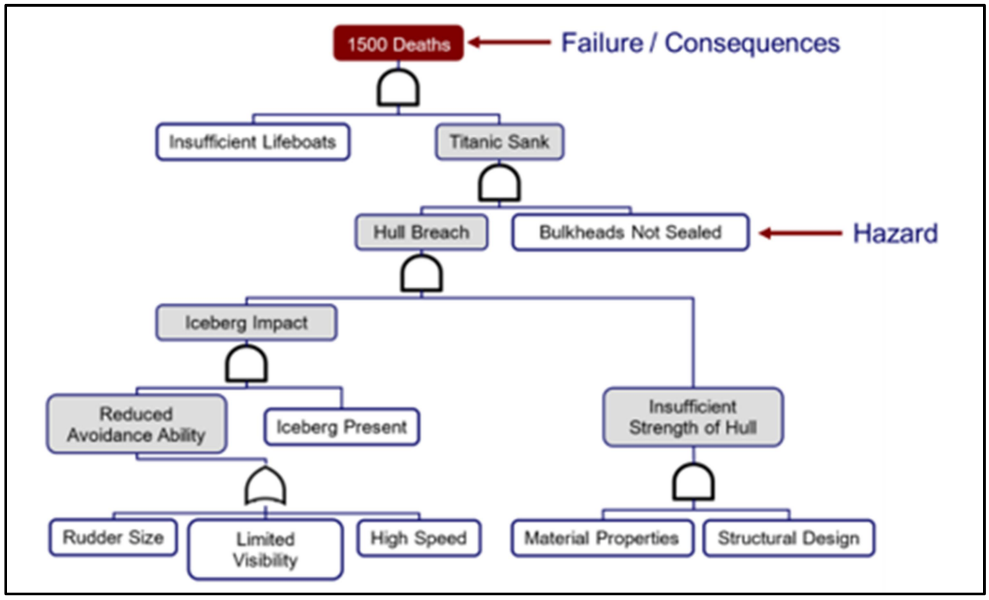

Figure 1. Simplified Fault-Tree Analysis of the Sinking of the Titanic

ETA is essentially an inverted FTA. Instead of starting from a failure and working backwards to a hazard, a hazard is selected and logic gates are used to assess potential consequences. This method is useful for predicting potential failures rather than determining the cause of an existing failure. It suffers from some of the same limitation as FTA. Additionally, it fails to examine the consequences of multiple concurrent hazards.

FMECA is an inductive method, similar to ETA, that seeks to tabulate all possible failures and then assess their severity, probability, detectability, and criticality. It excels at thoroughness but suffers from an inability to easily access multiple failures simultaneously. Additionally, its tabular format can be difficult to read. An example FMECA can be seen in Figure 2.

Function	Dispense amount of cash requested by customer				
Potential Failure Mode	Does not dispense cash			Dispenses too much cash	
Potential Effect(s) of Failure	Customer dissatisfied	Incorrect entry to demand deposit system	Discrepancy in cash balancing	Bank loses money	Discrepancy in cash balancing
Severity Rating	8			6	
Potential Cause(s) of Failure	Out of Cash	Machine jams	Power failure during transaction	Bills stuck together	Denominations in wrong trays
Occurrence Rating	5	3	2	2	3
Current Process Controls	Internal low-cash alert	Internal jam alert	None	Loading procedure (riffle ends of stack)	Two-person visual inspection
Detectability Rating	5	10	10	7	4
Risk Priority Number	200	240	160	84	72
Criticality	40	24	16	12	18

Figure 2. Portion of an FMECA
(Tague, 2004)

STPA takes a rather different approach and conceptualizes systems as control loops, as can be seen in Figure 3. The goal of STPA is to avoid focusing on exhaustively tabulating all vulnerabilities and attempting to quantitatively calculate probabilities. These can be difficult to do accurately for a system of any significant size. Instead STPA attempts to ensure that appropriate monitors and controls are in place for each component of the system (including its operators) so that *any* hazard is detected and addressed before it can cause a failure. In this way, it seeks to eliminate vulnerabilities while relying primarily on a qualitative, rather than a quantitative, assessment.

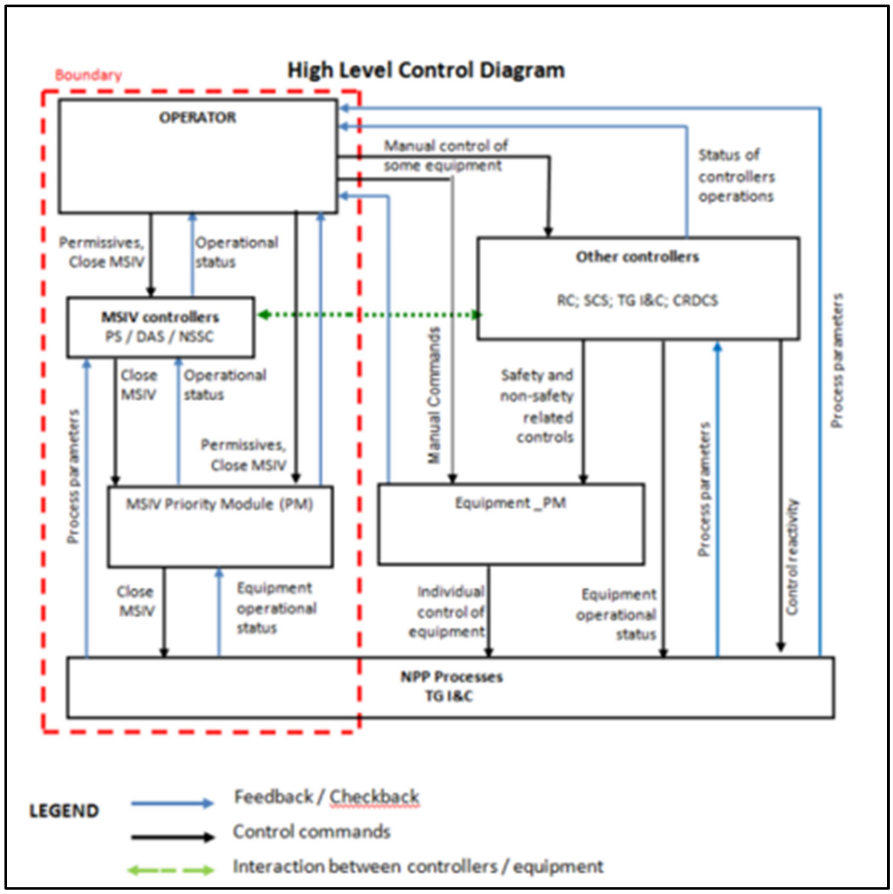

Figure 3. Example STPA Diagram
(Leveson, 2013)

Most vulnerability analysis methods fail to directly grapple with the problem of blame (though STPA does). Humans, engineers and program managers included, have a tendency to assign blame for a failure to someone or something other than themselves. FTA, ETA, and FMECA can enable this by allowing for an arbitrary "stopping point" (i.e., where the previous step in the causal chain is deemed the initiating hazard). In the Titanic FTA presented in Figure 1, for instance, why did we stop deconstructing the causes there? Were the designers of the rudder actually at fault? Or were the engineering standards poorly written? Were the owners of the boat at fault for installing too few lifeboats or should the government set a minimum required number of lifeboats? By adjusting the bounds of the analysis, it is easy to place blame on whomever the analyst desires.

STPA avoids this by (a) not assigning a specific "cause" of a failure and (b) by having every part of the system responsible for monitoring the other parts. Despite this, as the creators of STPA themselves acknowledge, the method has been criticized for its lack of a neat, one-page explanation of the causes of an accident (Leveson, 2013).

Cause-Effect Mapping

Cause-Effect Mapping (CEM) captures some of the benefits of STPA while still presenting distinct cause-effect paths. CEM has previously been applied to a case study of a Maritime Security System of Systems (Mekdeci et al., 2012) and to a supply chain case (Rovito & Rhodes, 2016). It consists of a mapping of causal chains that connect an exogenous hazard to a system degradation or failure, termed a *terminal event.* Each chain represents a vulnerability, sometimes called a *vulnerability chain* in order to emphasize that vulnerabilities are not discrete events. Terminal events are broadly defined and include any form of value loss. An example CEM (that lacks intervention points) can be seen in Figure 4. Similar to FTA, CEM is easily read in either direction, but it also allows for the simultaneous consideration of multiple failures and multiple hazards.

The hazards are external to the perspective of the defined user, and are thus sometimes called *external triggers.* An *intermediary event* is any unintended state change of a system's form or operations which could jeopardize value delivery of the program.

A CEM is not created for a system, but for a specific class of decision-maker. The hazards (referred to as "spontaneous events"; Figure 2) are exogenous from the point of view of the decision-maker that the CEM was made for. In this way, CEM avoids the aforementioned "blaming someone else" problem by making *all* hazards exogenous. The decision-maker only has control over the intermediary events. While she may not be at fault for any of the vulnerabilities, it is still her responsibility to address them.

CEM is fundamentally a qualitative analysis method, though it can be readily adapted into a quantitative form, by adding probabilities of transition to each intermediary. CEM provides immediate insight into which parts of the system warrant more detailed modeling. For instance, it may be useful to determine the likely time required for a specific vulnerability to complete. CEM enables classification of different vulnerability chains (by terminal event, by triggering event or type of triggering event, or by intermediary event). Additionally, it allows immediate identification of potential intervention points at intermediary events where multiple vulnerability chains intersect.

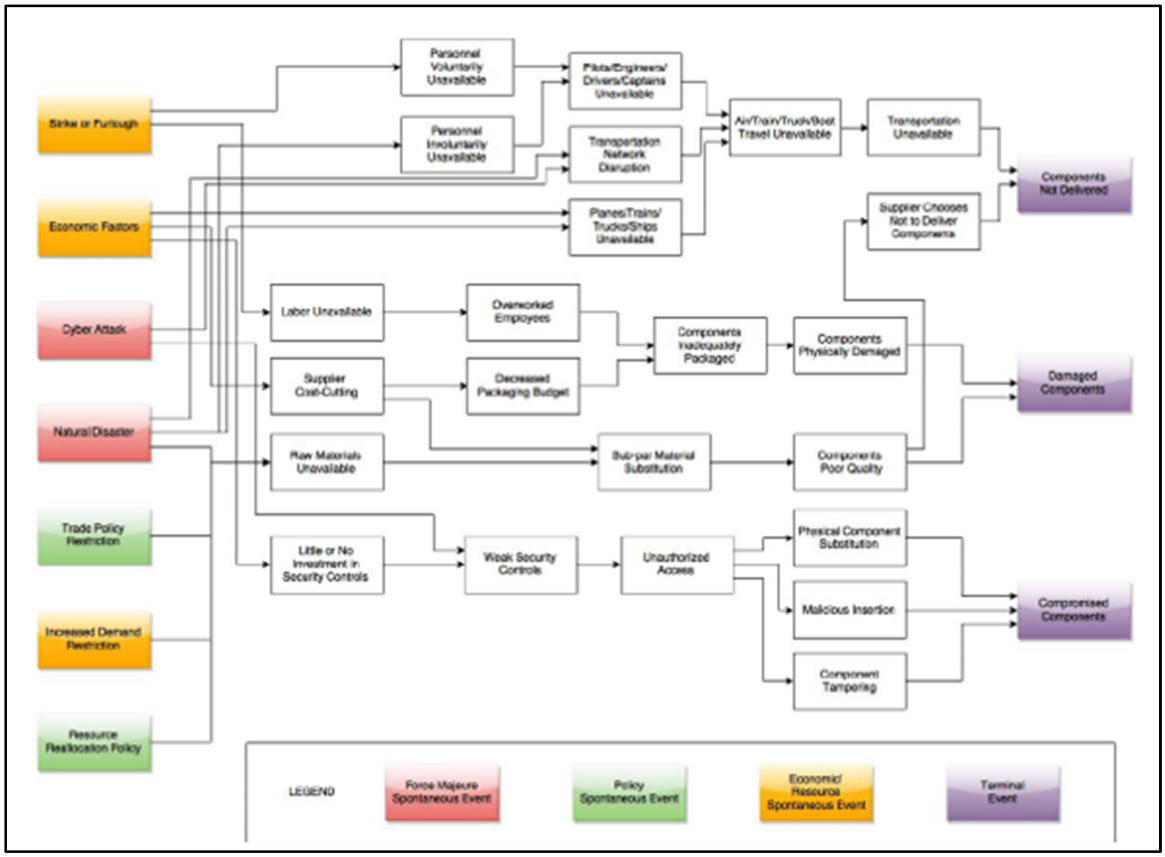

Figure 4. Example CEM of a Supply Chain
(Rovito & Rhodes, 2016)

Programmatic Vulnerabilities

Programmatic vulnerabilities differ from technological system vulnerabilities in a number of ways. Programmatic vulnerabilities tend to be more people-oriented, involving politics, economics, incentives, social interactions, and the like. They tend to be much less thoroughly studied, assessed, and understood, both in academia and in practice. Over the course of this study, a series of interviews was conducted with system engineers and program managers from a variety of fields, including defense, aerospace, manufacturing, and semiconductors. These interviews sought to provide insight into the following questions, in the context of a model-centric enterprise:

1. To what extent are program managers aware of programmatic vulnerabilities?
2. How do program managers conceptualize these vulnerabilities?
3. How do program managers respond to these vulnerabilities?
4. What vulnerabilities are present in MCE programs?
5. What cybersecurity vulnerabilities does MCE pose?

The first three questions provided some useful information regarding the status quo. Across all these industries, several facts were clear. First, many, if not most, programmatic vulnerabilities appear to be triggered by exogenous hazards beyond the control of the program manager. Some, such as poor scope or inadequate budget, are sometimes present before the first program manager joins a program. In general, program managers are at

least aware of the potential for these hazards and in some cases can even see them coming. There was some variance in responses to these hazards, however. Some program managers attempt to do things like preemptive padding of a schedule using a multiplicative factor based on experience. Others used their own spreadsheets and tools for estimating the "real" schedule or cost (that is, the schedule or cost that would result from a potential hazard becoming real). Little to no formal risk or vulnerability assessment would take place, however, and responses tended to be reactive rather than proactive, contributing to the "program management via crisis management" paradigm. In general, the perception by interviewees was that program managers rely heavily on expertise, rather than on formal education. While some, such as the INCOSE PM-SE Integration Working Group's Strategic Initiative, seek to directly address the exogenous hazards and others seek to provide risk registries that contain programmatic risks (Hall, 2018), there is a real need for easy to use tools for program managers to improve their ability to assess programmatic vulnerabilities and respond to them. The fourth question was intended to supplement and corroborate a Reference CEM, as discussed in the following section.

CEM for Model-Centric Programs

In this research, an objective was to develop a high-level cause-effect map for model-centric programs to serve as a reference for use by program managers. The intent is for this CEM to serve both as a standalone resource for such program managers, as well as a basis for organizations to construct their own, program-specific CEM with added detail. Additionally, this research sought to document the general steps to create and use CEM in general, as well as to conduct some initial usability testing of the usefulness of the Reference CEM.

Generating the CEM

Generating a CEM can be done in different ways and to different levels of granularity, depending on the need of the stakeholder. This process can be done with groups, such as project teams, as well as individually. The general process is as follows:

1. The stakeholder herself lists potential hazards posed to the program.
2. She then traces the consequences of each of these hazards through the intermediary events to the final terminal events.
3. The process is then done in reverse: She looks at the terminal events, adds in any that are still missing, and works backwards on how they might come about.
4. She then examines the causal connections between each intermediary event to see if there are any additional connections not previously noticed.
5. Finally, she consults lessons learned databases, case studies, and other experts to generate additional hazards, intermediary events, causal connections, and interventions, as well as to verify existing ones.

The Reference CEM shown in Figure 5 was generated through a combination of methods. At this time, there is little literature on programmatic vulnerabilities posed by MCE. Most negative case studies, that is, those that depict failures (Software Engineering Institute, 2007), and lessons learned databases (NASA Office of the Chief Engineer, n.d.), are from prior to the rise of MCE and thus deal with general vulnerabilities. Existing case studies that directly deal with MCE tend to be more positive, likely due to the rising popularity of the paradigm (Conigliaro, Kerzhner, & Paredis, 2009; Maley & Long, 2005; Martz & Neu, 2008). As a result of these, extrapolations from extant vulnerabilities had to be made, along with hypothetical inversions of the positive instances of MCE. Additional

vulnerabilities were contributed by group brainstorming during the in-class activity discussed further in the section titled Usability Testing. All of these were supplemented and confirmed by the same interviews discussed previously.

The higher level of detail in the upper portion of the CEM, which includes issues such as training, misunderstood model assumptions, and level of trust in the models, represents the increased degree of concern that interviewees had about these issues. In general, the domain of aligning culture and expertise with well-designed and well-documented toolsets was of high priority. Failure to accomplish this had led to significant problems in past projects, but is viewed as a surmountable difficulty moving forward.

Using the CEM

Vulnerability analysis methods are most commonly applied either to the design or the operation of an engineered system. This is usually done to improve its design or investigate a failure. However, these methods can also be applied to the program itself. Instead of hazards such as “relief valve failure” and “solar flare” instead we have “hiring freeze” and “unexpected technological hurtle.” It can be difficult to assess likelihoods for such hazards, but even qualitative analysis can be useful. Similarly, terminal events are not “nuclear meltdown” or “loss of communications” but instead “schedule delay” or “failure during verification/validation.”

CEM in particular can be used to assess vulnerabilities in multiple ways and by different individuals. Four uses are described as follows:

(A) By a Program Manager: Assessing potential future vulnerabilities and planning possible interventions

(B) By a Program Manager: Determining specific vulnerabilities to address in response to the presence of a specific hazard

(C) By the Program Organization: Changing program processes to mitigate or eliminate vulnerabilities

(D) By Researchers: Organizing and classifying vulnerabilities into various categories or types

All of these start with the creation of a CEM for the organization’s standard program process or for a particular program. Once this is completed, additional steps can be taken, including

1. Identifying notable intermediary events and potential intervention points
2. onducting more detailed modeling of specific vulnerability chains
3. Classifying vulnerability chains to enable future study and potential mitigation.

While a program manager would be well-served by the creation of a CEM specific to their own program, there is some benefit in using a Reference CEM for model-centric programs in general. Such a Reference CEM can be seen in Figure 5.

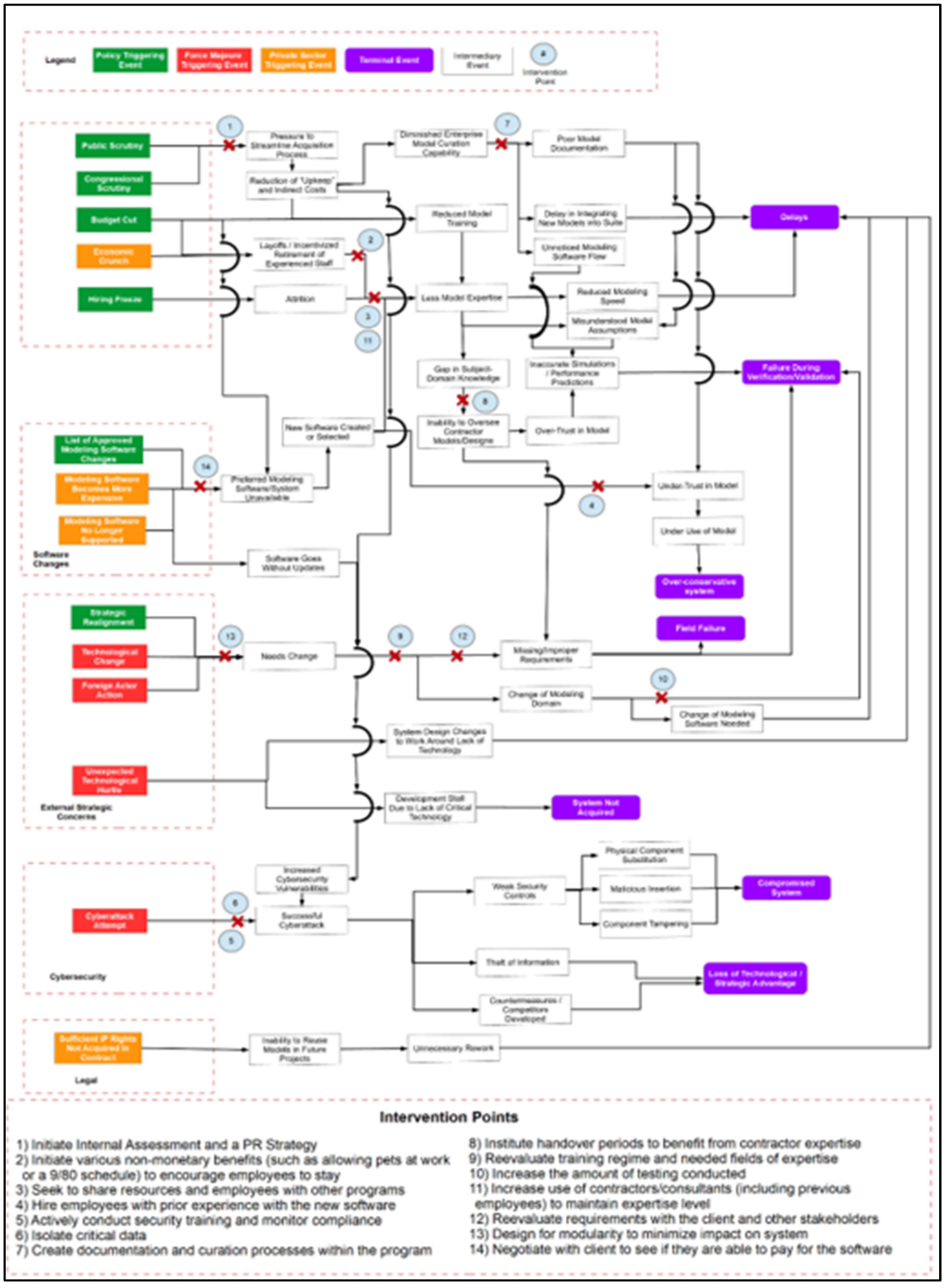

Figure 5. Reference CEM for Model-Centric Vulnerabilities (Preliminary)

Use (A) is most relevant for novice program managers or program managers using MCE for the first time. A senior program manager or team of program managers creates a CEM for their organization's program process. This CEM can then be provided to the novice for study and reference. The program manager can then learn what can go wrong and how to intervene. In this case, the CEM could be tied to a Lesson's Learned database, such as NASA's Lessons Learned Information System (NASA Office of the Chief Engineer, n.d.). This enables concrete examples and consequences to be linked to each vulnerability. One of the important factors here is that the CEM does not just present potential interventions, but it also places them in the appropriate part of the causal sequence. This enables the program manager to not only know how to intervene, but at what point.

Use (B) is relevant to all program managers, regardless of level of experience. Once a hazard manifests, the program manager examines the CEM to assess potential consequences and options. He can then respond quickly to head off any cascading effects. This may require additional analysis of a specific vulnerability chain or an individual intermediary event. System dynamics is a method particularly useful for this due to the preexisting models of many organizational phenomena (Rouwette & Ghaffarzadegan, 2013). For instance, *Attrition, Reduced Model Training*, and *Less Model Expertise* can be modeled by adapting the rookie fraction model shown in Figure 6 into the more MCE-relevant model shown in Figure 7. In this model, it is apparent that a hiring freeze (which would set the "Growth Rate" variable to zero) has no immediate impact, as rookies will continue to develop into experienced employees and model expertise will continue to accumulate. Overtime, however, the dearth of new rookies will result in fewer experienced employees, increasing the error rate. These kinds of long-term, indirect impacts are likely to become more common with increased use of MCE.

Use (C) is the traditional use of vulnerability assessment methods: to improve a design or investigate a failure. The program organization can change policies or create infrastructure to either mitigate or wholly eliminate certain vulnerability chains. For example, if the organization elects to only use modeling software produced in-house, the three hazards in the "Software Changes" grouping of Figure 5 are no longer relevant. Such a change could be costly though or even introduce new vulnerabilities, so careful analysis is necessary. In this use, the CEM is a visual representation of a risk registry, tabulating all possible hazards to the program and mitigation choices made (Hall, 2018).

In Use (D), CEM is used to organize and classify vulnerability chains. Two obvious classifiers are terminal events and hazards. Which is used to organize a CEM depends on whether the user wants to examine the causal chains forwards or backwards. Beyond this, however, more complicated classifiers are possible. As can be seen in Figure 5, external triggers that result in similar vulnerability chains are grouped together. By "similar," we mean that these vulnerability chains either involve many of the same intermediary events or that they involve the same part of the program. For instance, most of the intermediary events involving model curation and trust are located close to one another in the center-top of the figure. Once these groupings have been identified, they can be considered together, such as the "Belt-tightening" grouping, and common means of intervention considered.

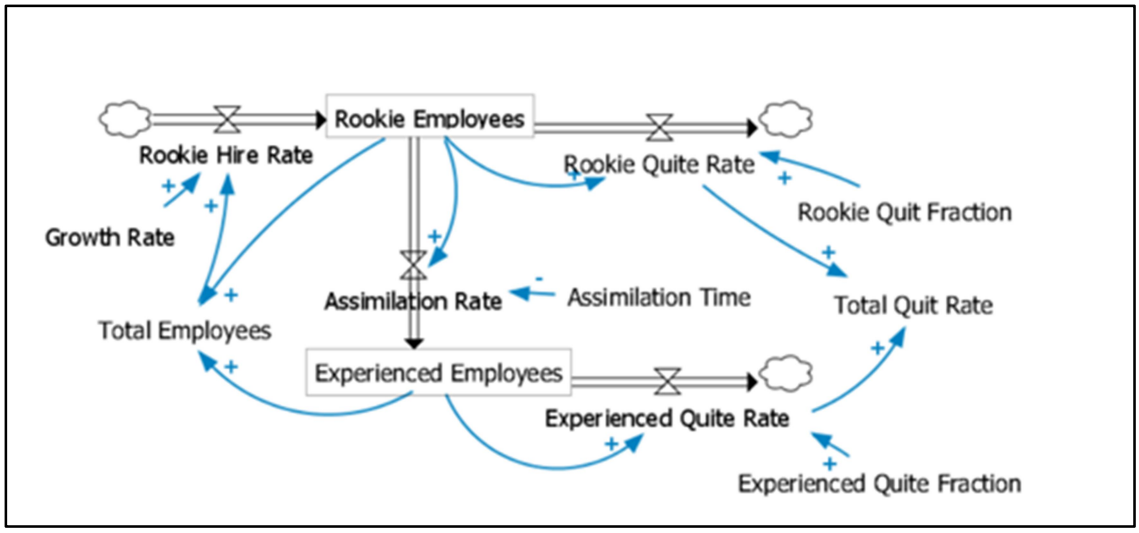

Figure 6. System Dynamics Model of Employee Training Rate
(Adapted from Sterman, 2000)

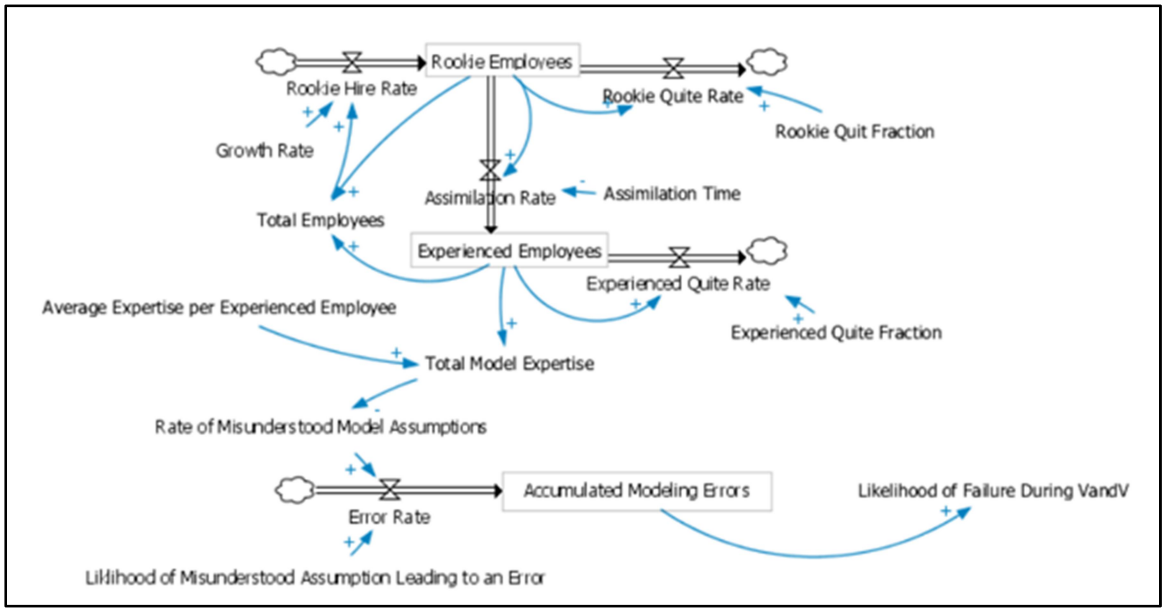

Figure 7. System Dynamics Model of Accumulated Modeling Errors

Usability Testing

While several potential use cases were proposed in the previous section, due to the scale, duration, cost, and uniqueness of major MCE programs, it is difficult to systematically test the utility of either the Reference CEM or of using CEM in general. Some simple usability testing was explored in the interviews with experts. Usability was also considered through analyzing the results of a scenario-based exercise on vulnerability analysis that had been generated in a graduate class activity involving techniques for investigating enterprises.

The classroom activity involved approximately 40 students from various backgrounds, most having prior experience in either industry or the military as systems engineers and/or program managers. The students were randomly divided into six groups of 5 to 7 students each, and each group was provided with the same "context" for the activity, as follows:

> You are a project manager for a vehicle manufacturer. Your current project is designing a lightweight troop transport vehicle for the U.S. military. It has a variety of high-tech components, including encrypted radio and satellite communication systems, an explosives detector, and night vision cameras. The design and testing process will take multiple years. Your company considers this a major project in terms of the resources put into it, the revenue received for it, and the potential for future military contracts. The military, as part of the contract, specified that the design and production process should predominately rely on models (sometimes called model-centric engineering) rather than written specifications.

Each group was provided with one or two selected external triggers or hazards to respond to. They were asked to discuss and record the potential negative impacts these hazards may have on the engineering environment and how they might act to mitigate these consequences. The hazards provided were as follows:

1. An unrelated military project (at another company) to design a next-generation missile defense system has ended up in the national news. That system has gone significantly over budget, has been repeatedly delayed, and still looks like it is a long way off from being completed. Public accusations of mismanagement and waste are being made, including frivolous travel and lavish company events. Congress and the Department of Defense are now carefully scrutinizing all other major projects for potential mismanagement or waste, including your project.
2. After recent elections, there is significant political pressure on Congress to reduce federal spending. As a result of this, they are making significant cuts to many agencies and programs, including the military. The decrease in government spending is likely to impact your company's other projects and may impact yours as well.
3. Government intelligence officials inform you that your company, and perhaps even your project, is likely to be the target of cybersecurity attacks based out of another nation.
4. This is your first project of this type (your prior experience was in designing civilian vehicles). You now have to choose whether your project will use the set of modeling software that you are accustomed to from the civilian projects or use another set that is more commonly used on military projects, but that you are unfamiliar with.

5. A recent economic downturn, coupled with a government that is cutting back on its military spending, has resulted in your company declaring a hiring freeze for an indeterminate amount of time.
6. Another project at your company is threatening to miss its deadlines. In order to get it back in line, its project manager is requesting that personnel from other projects, such as yours, be reallocated to hers.
7. The design requirements from the military that you are currently working with were put together with the idea of using this vehicle in a currently ongoing conflict. However, U.S. involvement in this conflict is winding down and the military is currently unsure where this vehicle will be used in the future. The context (and thus the requirements) may change during the design process. Furthermore, your models were created with the current context in mind.
8. A recent economic downturn, coupled with a government that is cutting back on its military spending, has resulted in your company providing incentives for early retirement to the more experienced, higher paid employees. You have no intention of retiring early yourself, but some of those working on your project might accept the offer.
9. In order to minimize rework and redundancy, your company has recently started an initiative pushing for increased reuse of components, designs, and models from one project to another. Your previous project also involved a night vision camera, but in a very different application context.
10. A particular piece of simulation software that your company has used on similar projects in the past is licensed from another company. The license contract is up for renewal soon and the price might go up significantly. You are uncertain if your company's executives will approve the license renewal.

After a period of 20 minutes, the students were taught about causal chains and use of the CEM reference model as a technique for investigating enterprise vulnerabilities. Each group was instructed to re-write the previously identified vulnerabilities and interventions as causal chains and map them on the provided CEM (Figure 8), as well as coming up with new ones. After another period of 20 minutes, their results were collected, a group debrief was given, and students shared general feedback on the class activity. [Note: The CEM presented in Figure 8 is similar to that in Figure 5 but not identical, as knowledge gained in interviews and usability testing has since been used to further develop the CEM.]

Several useful pieces of information could be garnered through analyzing the documented results of the class activity that had been conducted. In the out-briefing material, the participants expressed unanimous agreement that using CEM and conceptualizing programmatic vulnerabilities as causal chains was helpful, though the perceived degree of usefulness varied from "slightly" to "extremely." Additionally, team out-briefs reported on four primary forms of how the CEM helped in their assigned scenario in the class exercise:

1. Identifying high priority intervention points: (70%)
2. Identifying new vulnerabilities: (55%)
3. Understanding the causal path / Reframing the concept of vulnerabilities: (45%)
4. Understanding interrelationships between vulnerabilities: (40%)

The relative importance of this first point was corroborated by the group outputs that were generated. It was clear that in this instance, when provided with a Reference CEM, the

groups tended to focus on identifying where and how to intervene in the vulnerability chains. During the first round (without the CEM), most of the groups had presented their vulnerabilities and interventions as unordered lists of short phrases, typically unpaired (i.e., vulnerabilities were not matched with interventions). These short phrases were typically isolated events such as “team feeling more cautious” and “reputation damages.” The ultimate outcomes of these were assumed rather than explicitly stated. Once the CEM was introduced in the second round, the matching of interventions to vulnerabilities appeared to become much more clear, and most groups also identified additional interventions.

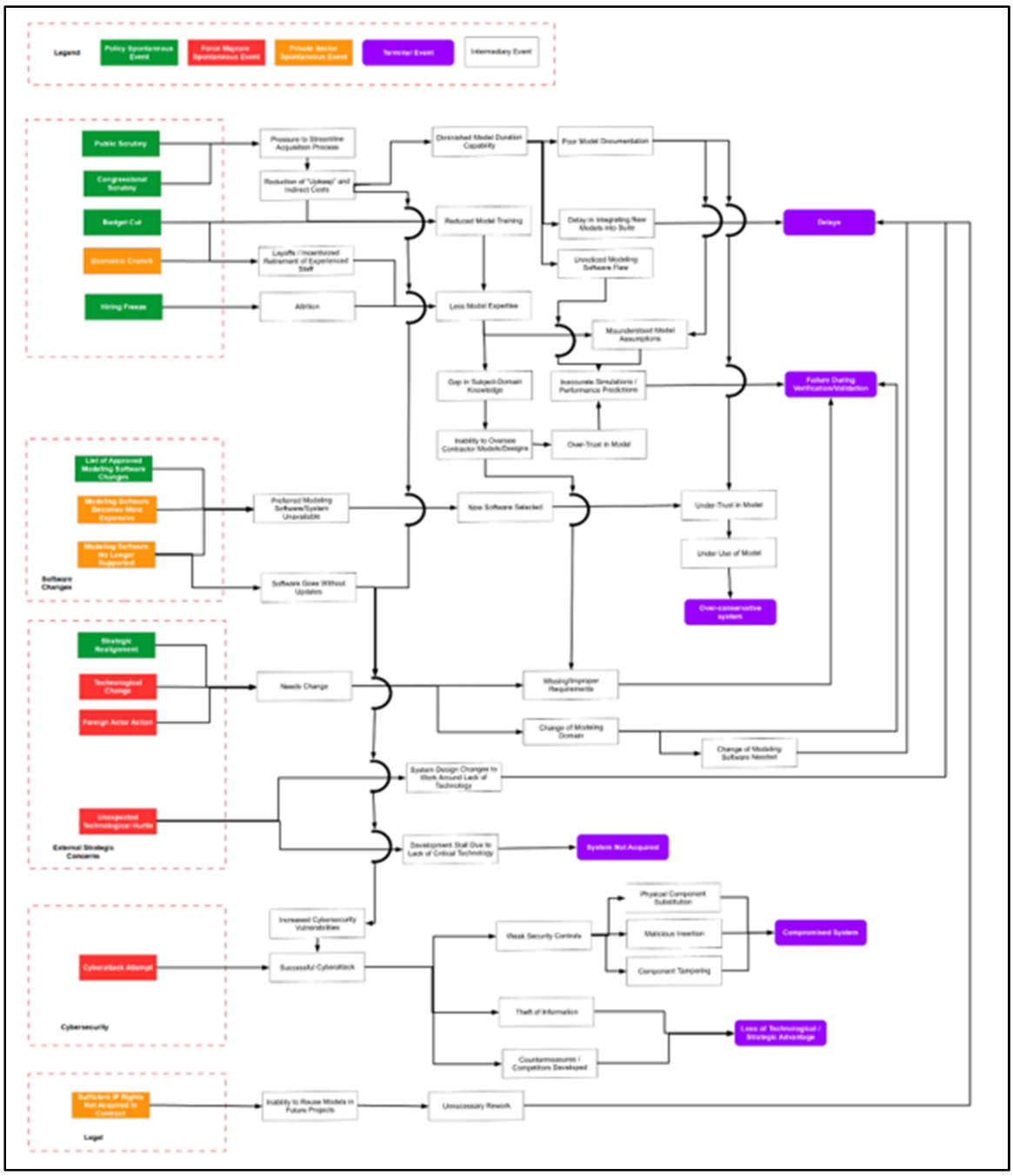

Figure 8. Reference CEM Used as a Basis for Usability Testing

Future Directions

Digital engineering is transforming the systems acquisition process (Zimmerman, Gilbert, & Salvatore, 2017), including the model-centric techniques and toolsets. Enterprises face new challenges in this transformation, including potential for new vulnerabilities within model-centric enterprises. While vulnerability analysis of products and systems is performed, examining vulnerabilities within an enterprise is less common. Vulnerability analysis of the enterprise becomes increasingly urgent given increasing complexity and interconnectivity in model-centric environments used to make system decisions. The interim outcomes of this research, including expert interview results, show the potential benefit of cause-effect mapping techniques and availability of a reference map for model-centric program vulnerability analysis. Additional expert interviews are planned with an expanded set of stakeholders.

Insights into usability were also gained through analyzing a data set that had been generated in a classroom setting. Accordingly, the results should be viewed as purely exploratory, but there appears to be good justification for future research to include conducting a controlled human-subjects research experiment (similar to the scenario-based exercise used in a classroom setting). Additionally, an important future research activity is to evaluate the Reference CEM on a pilot project in a real world model-centric engineering program. Further development of the Reference CEM is planned for next phase research, including more extensive investigation of cybersecurity vulnerabilities resulting from model-centric practices and infrastructure.

Conclusion

Acquisition programs increasingly use model-centric approaches, generating and using digital assets throughout the lifecycle. Model-centric practices have matured, yet in spite of sound practices, there are uncertainties that may impact programs over time. The emergent uncertainties (policy change, budget cuts, disruptive technologies, threats, changing demographics, etc.) and related programmatic decisions (e.g., staff cuts, reduced training hours) may lead to cascading vulnerabilities within model-centric acquisition programs, potentially jeopardizing program success. Ongoing research has led to a preliminary CEM Reference Model that aims to provide program managers with a means to assess, prioritize, and mitigate model-centric vulnerabilities. Usability testing of the reference model has shown positive benefits for practical use in assessing vulnerabilities of model-centric programs. Anticipated results are empirically-grounded vulnerabilities of model-centric programs and a cause-effect mapping reference model for identifying vulnerabilities and interventions.

References

Baldwin, K. J., & Lucero, S. D. (2016). Defense system complexity: Engineering challenges and opportunities. *The ITEA Journal of Test and Evaluation, 37*(1), 10–16.

Blackburn, M., Verma, D., Dillon-Merrill, R., Blake, R., Bone, M., Chell, B., … Evangelista, E. (2017). *Transforming systems engineering through model-centric engineering.* Hoboken, NJ: Systems Engineering Research Center. Retrieved from http://www.sercuarc.org/wp-content/uploads/2014/05/A013_SERC-RT-168_Technical-Report-SERC-2017-TR-110.pdf

Conigliaro, R. A., Kerzhner, A. A., & Paredis, C. J. J. (2009). Model-based optimization of a hydraulic backhoe using multi-attribute utility theory. *SAE International Journal of Materials and Manufacturing, 2*(1), 298–309. Retrieved from http://www.sae.org

Defense Acquisition University. (n.d.). *Glossary of defense acquisition acronyms and terms.* Retrieved from https://dap.dau.mil/glossary/Pages/Default.aspx

Glaessgen, E. H., & Stargel, D. (2012). *The digital twin paradigm for future NASA and U.S. Air Force vehicles*. Paper for the 53rd Structures, Structural Dynamics, and Materials Conference (pp. 1–14). Retrieved from https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20120008178.pdf

Hall, D. (2018). Risk identification challenge. INCOSE 2018 International Workshop, Jacksonville, FL.

Kellner, T. (2015). Wind in the cloud? How the digital wind farm will make wind power 20 percent more efficient. GE Reports. Retrieved from http://www.gereports.com/post/119300678660/wind-in-the-cloud-how-the-digital-wind-farm-will/

Krishnan, R., Virani, S., & Gasoto, R. (2017). Discovering toxic policies using MBSE constructs. In A. M. Madni & B. Boehm (Eds.), *Conference on Systems Engineering Research*. Redondo Beach, CA.

Leveson, N. (2013). *An STPA primer*. Cambridge, MA.

Lockheed Martin. (2015). Digital tapestry. Retrieved from http://www.lockheedmartin.com/us/what-we-do/emerging/advanced-manufacturing/digital-tapestry.html

Maley, J., & Long, J. (2005). *A natural approach to DoDAF*. Blacksburg, VA.

Martz, M., & Neu, W. L. (2008). Multi-objective optimization of an autonomous underwater vehicle. *Oceans* (Vols. 1–4), 1042–1050\r2248.

Mekdeci, B., Ross, A. M., Rhodes, D. H., & Hastings, D. E. (2012). A taxonomy of perturbations: Determining the ways that systems lose value. In *Proceedings of the 2012 IEEE International Systems Conference*, *Proceedings* (pp. 507–512). Vancouver, British Columbia: IEEE. https://doi.org/10.1109/SysCon.2012.6189487

The MITRE Corporation. (2015). Terminology. Retrieved February 20, 2018, from https://cve.mitre.org/about/terminology.html

The MITRE Corporation. (2017). CVE-2017-5753. Retrieved February 20, 2018, from https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5753

NASA Office of the Chief Engineer. (n.d.). NASA public lessons learned system. Retrieved July 13, 2017, from https://llis.nasa.gov/

Peterson, T. A. (2017). INCOSE transformation strategic objective. In *INCOSE Webinar 106. INCOSE*.

Piaszczyk, C. (2011). Model based systems engineering with Department of Defense architectural framework. *Systems Engineering, 14*(3), 305–326. https://doi.org/10.1002/sys.20180

Rouwette, E., & Ghaffarzadegan, N. (2013). The system dynamics case repository project. *System Dynamics Review, 29*(1). https://doi.org/10.1002/sdr.1491

Rovito, S. M., & Rhodes, D. H. (2016). Enabling better supply chain decisions through a generic model utilizing cause-effect mapping. In *Proceedings of the 2016 Annual IEEE Sytems Conference.* Orlando, FL: IEEE.

Software Engineering Institute. (2007). Acquisition archetypes: Firefighting. Pittsburgh, PA.

Sterman, J. D. (2000). Coflows and aging chains. In *Business dynamics: Systems thinking and modeling for a complex world* (pp. 469–512). Boston, MA: Irwin McGraw-Hill.

Tague, N. (2004). Failure mode effects analysis (FMEA). Retrieved January 17, 2017, from http://asq.org/learn-about-quality/process-analysis-tools/overview/fmea.html

Virani, S., & Rust, T. (2016). Using model based systems engineering in policy development : A thought paper. In *Conference on Systems Engineering Research.* Huntsville, AL.

Zimmerman, P., Gilbert, T., & Salvatore, F. (2017). Digital engineering transformation across the Department of Defense. The *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*. https://doi.org/10.1177/1548512917747050

Acknowledgments

This material is based upon work by the Naval Postgraduate School Acquisition Research Programs under Grant No. N00244-17-1-0011.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net