MIT-PM-18-222

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

3 August 2018

Dr. Donna H. Rhodes

Mr. Jack Reid

Massachusetts Institute of Technology

Abstract

Acquisition programs increasingly use model-centric approaches, generating and using digital assets throughout the lifecycle. Recent research supports new model-centric practices, 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. The research objectives included: (1) investigate uncertainties and related decisions that may lead to potential vulnerabilities in model-centric acquisition programs; (2) generate a CEM model for aiding program managers in detecting and assessing vulnerabilities related to the program's model-centric practices and environment; and (3) define a step-wise process for applying the generic model in assessing and mitigating model-centric vulnerabilities.

This research seeks to provide program managers with the means to identify model-centric program vulnerabilities and determine where interventions can most effectively be taken. The technical approach for the research began with literature review and gathering results of past research studies of relevance, including studies of model-centric environments and transformations from 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. Cause-Effect Mapping, a technique developed at MIT, was employed to examine cascading effects between emerging uncertainties and terminal outcomes. Using the results, a CEM was generated and used for discussion with subject matter experts, and information on uncertainties and leading indicators was collected. Analysis was performed to consider the cascading vulnerabilities and potential intervention options. The results were used to refine the CEM and analytic approach to develop a generic model for vulnerability assessment of model-centric programs. Usability of the resulting model was tested with selected research stakeholders. Several analytic approaches were investigated.

This research aims to contribute a useful approach for assessing, detecting and mitigating vulnerabilities in acquisition programs, specifically related to the use of model-centric practices and environments. The approach is compatible with existing DoD vulnerability assessment practices and frameworks. Research results are empirically-grounded vulnerabilities of model-centric programs, a Reference CEM for identifying vulnerabilities and interventions, and a step-wise process that program managers can apply on their programs using the CEM to assess, prioritize, and mitigate model-centric vulnerabilities.

Keywords: model-centric program, vulnerabilities, cause-effect mapping, acquisition, interventions

About the Author

Dr. Donna H. Rhodes – 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 Ph.D. in Systems Science from T.J. Watson School of Engineering at Binghamton University.

Massachusetts Institute of Technology
77 Massachusetts Avenue, E17-361
Cambridge, MA 02139
Tel: (617)-324-0473
rhodes@mit.edu

Mr. Jack Reid – Jack Reid is a graduate student with the Systems Engineering Advancement Research Initiative (SEAri) at the Massachusetts Institute of Technology. Reid received a master's degrees in both Aeronautics & Astronautics and Technology & Policy. He is presently a doctoral student in MIT Media Arts & Sciences. 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.

Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
Tel: 512-350-5261
jackreid@mit.edu

THIS PAGE INTENTIONALLY LEFT BLANK

MIT-PM-18-222

ACQUISITION RESEARCH PROGRAM SPONSORED REPORT SERIES

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

3 August 2018

Dr. Donna H. Rhodes

Mr. Jack Reid

Massachusetts Institute of Technology

Disclaimer: This material is based upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. N00244-17-1-0011. The views expressed in written materials or publications, and/or made by speakers, moderators, and presenters, do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

THIS PAGE LEFT INTENTIONALLY BLANK

Table of Contents

THIS PAGE LEFT INTENTIONALLY BLANK

List of Figures

THIS PAGE LEFT INTENTIONALLY BLANK

Executive Summary

This report describes recent research in support of acquisition programs facing challenges in transforming to model-centric engineering. Acquisition programs increasingly use model-centric approaches, generating and using digital assets throughout the lifecycle. Recent research supports new model-centric practices, 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. The research objectives included: (1) investigate uncertainties and related decisions that may lead to potential vulnerabilities in model-centric acquisition programs; (2) generate a generic model for aiding program managers in detecting and assessing vulnerabilities as related to the program's model-centric practices and environment; and (3) define a step-wise process for applying the generic model in assessing and mitigating model-centric vulnerabilities.

This research seeks to provide program managers with the means to identify model-centric program vulnerabilities and determine where interventions can most effectively be taken.   The technical approach for the research began with literature review and gathering results of past research studies of relevance, including studies of model-centric environments and transformations from 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. Cause-Effect Mapping, a technique developed at MIT, was employed to examine cascading effects between emerging uncertainties and terminal outcomes. Using the results, a CEM was generated and used for discussion with subject matter experts, and information on uncertainties and leading indicators was collected. Analysis was performed to consider the cascading vulnerabilities and potential intervention options. The results were used to refine the CEM and analytic approach to develop a generic model for vulnerability assessment

of model-centric programs. Usability of the resulting model was tested with selected research stakeholders. Several analytic approaches were investigated.

This research aims to contribute a useful approach for assessing, detecting and mitigating vulnerabilities in acquisition programs, specifically related to the use of model-centric practices and environments. The approach is compatible with existing DoD vulnerability assessment practices and frameworks. Anticipated results are empirically-grounded vulnerabilities of model-centric programs and a CEM generic model for identifying vulnerabilities and interventions. The research outcome is a step-wise process that program managers can apply on their programs using the generic model to assess, prioritize and mitigate model-centric vulnerabilities.

The research resulted in two published papers for 2018 Naval Postgraduate School Acquisition Research Symposium, one presented in a panel session and the other as a poster paper.

Background

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. The research discussed in this report has investigated the use of cause-effect mapping as a mechanism for better enabling program managers and system engineers to anticipate and respond to programmatic vulnerabilities as related to model-centric environments.

Prior research on cause-effect mapping for vulnerability analysis has included commercial and defense sectors. The research in the project is primarily focused on the defense sector. The following terms are defined as used in this research project.

acquisition	The conceptualization, initiation, design, development, test, contracting, production, deployment, integrated product support (IPS), modification, and disposal of weapons and other systems, supplies, or services (including construction) to satisfy DoD needs, intended for use in, or in support of, military missions.
analysis	An evaluation, either quantitative or qualitative; synonymous with assessment.
assessment	Synonymous with analysis.
causal chain	A series of events, with each event causing or being an integral part of the cause, or the next “link” in the chain.
causal factor	Any aspect of the system which, when removed or changed, is likely to reduce the occurrence of emergent behavior, or, when induced, is likely to increase the occurrence of emergent behavior.
classification	A generic term for sorting a set by some defined metric, either quantitative or qualitative. Includes both taxonomies and typologies.
Digital Systems Model	A digital representation of a defense system, generated by all stakeholders that integrates the authoritative technical data and

	associated artifacts which define all aspects of the system for the specific activities throughout the system lifecycle.
external trigger	A hazard external from the perspective of a defined user; sometimes referred to as a spontaneous event.
framework (in vulnerability assessment)	A method of assessing and analyzing hazards, vulnerabilities, and risks in a comprehensive manner, as well as determining appropriate mitigations and countermeasures.
hazard	A system or environmental state that has the potential to disrupt the system.
intermediary event	Any unintended state change of a system’s form or operations which could jeopardize value delivery of the program and is situated along a causal chain connecting an external trigger to a terminal event.
intervention point	A means of disrupting or mitigating a vulnerability chain.
model-centric engineering	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.
program	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.
risk	A measure of the probability of a system disruption, the consequences of that disruption, and sometimes other factors such as detectability or criticality.
technique (in vulnerability assessment)	A method for analyzing a particular set of hazards, vulnerabilities, or risks.
terminal event	Any form of system degradation or failure, or of value loss.
vulnerability	The means by which the hazard might disrupt the system.
vulnerability chain	A conceptualization and representation of a vulnerability as a causal chain.

Literature Investigation

Several areas of research were investigated during the project, model-centric engineering; vulnerability, hazard and risk analysis; and cause-effect mapping.

Model-Centric Engineering

Acquisition program management is grounded in management science and a sound set of practices that have evolved over decades. New challenges arise as acquisition becomes increasingly model-centric and traditional engineering transforms to digital model-based engineering. The systems engineering field is going through a period of significant transformation (Peterson, 2017). Advances in computing, digital workflows, and multi-domain-multi-scale models are leading to new concepts and approaches for model-centric engineering (Piaszczyk, 2011; Reid & Rhodes, 2016; Glaessen & Sargel, 2015; 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 ("Digital Tapestry," 2015; Glaessgen & Stargel, 2012; Kellner, 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, and organizational issues exist. Program managers and system engineers must learn how to identify and address programmatic vulnerabilities that are associated with model-centric transformation, as these may pose threats to programmatic schedule, budget and performance outcomes.

Current program managers and engineering leaders have significant experience with modern engineering processes, which they use 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.

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 research, 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 usage 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: 1.25 deaths per 100 million vehicle miles. Sometimes risk is instead expressed as a multiplication of likelihood and consequence or includes 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).

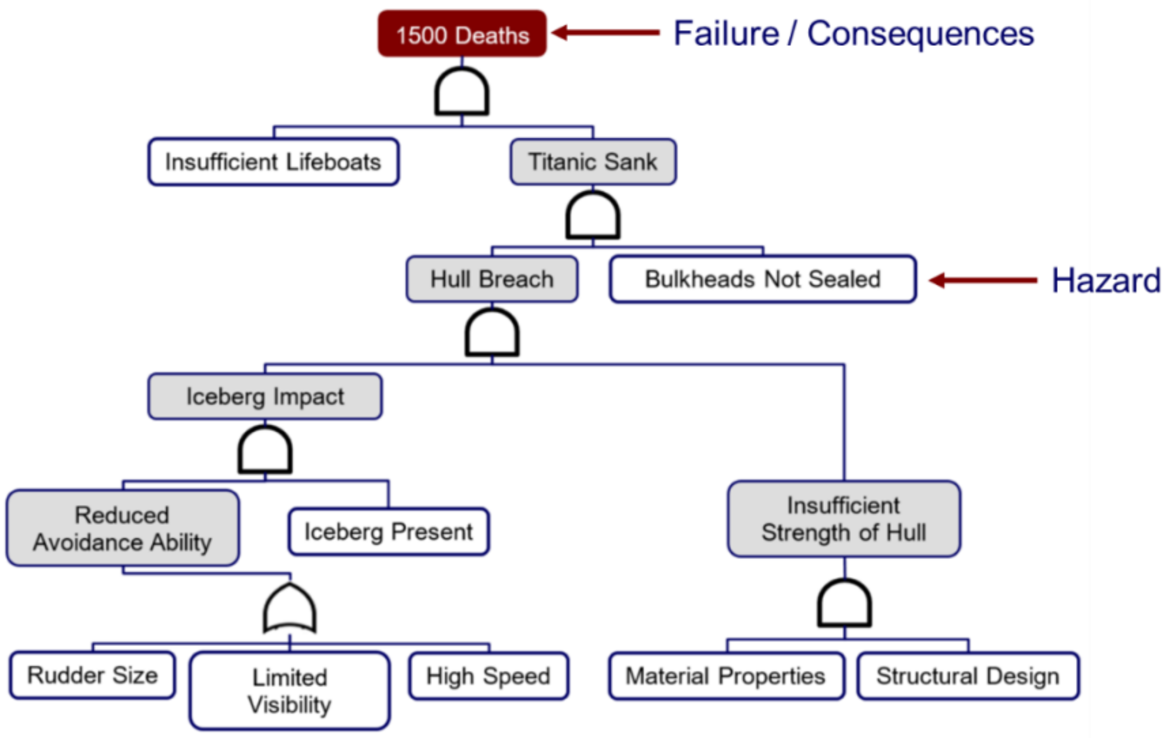


Figure 1. Simplified Fault-Tree Analysis of the sinking of the Titanic

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 a previous one. 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, which 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 is 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. Adapted from (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.

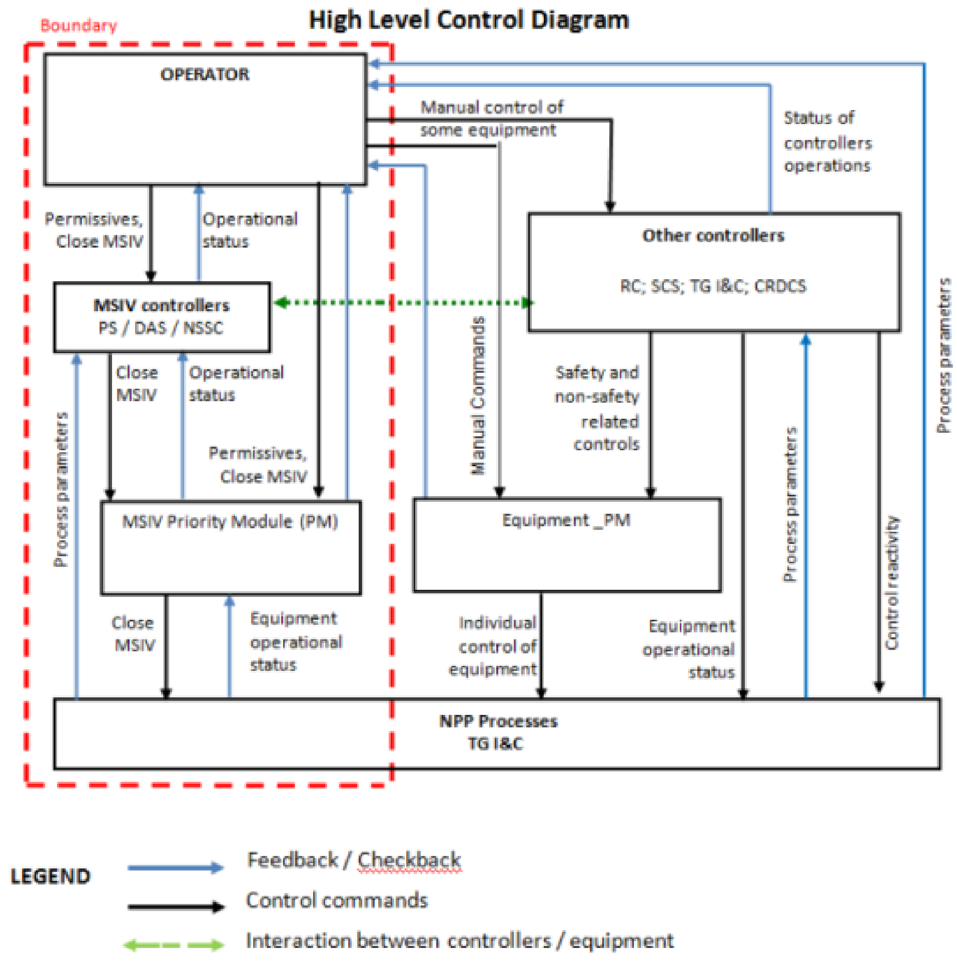


Figure 3. Example STPA Diagram. Image from (Leveson, 2013)

Most of vulnerability analysis methods fail to directly grapple with the problem of blame (though STPA does). Humans 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 putting too few lifeboats or should the government or industrial organizations set a minimum required number of lifeboats? By adjusting 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 originators of STPA themselves acknowledge, the method has been criticized for its lack of a neat, one-page explanation of 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, Ross, Rhodes, & Hastings, 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 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.

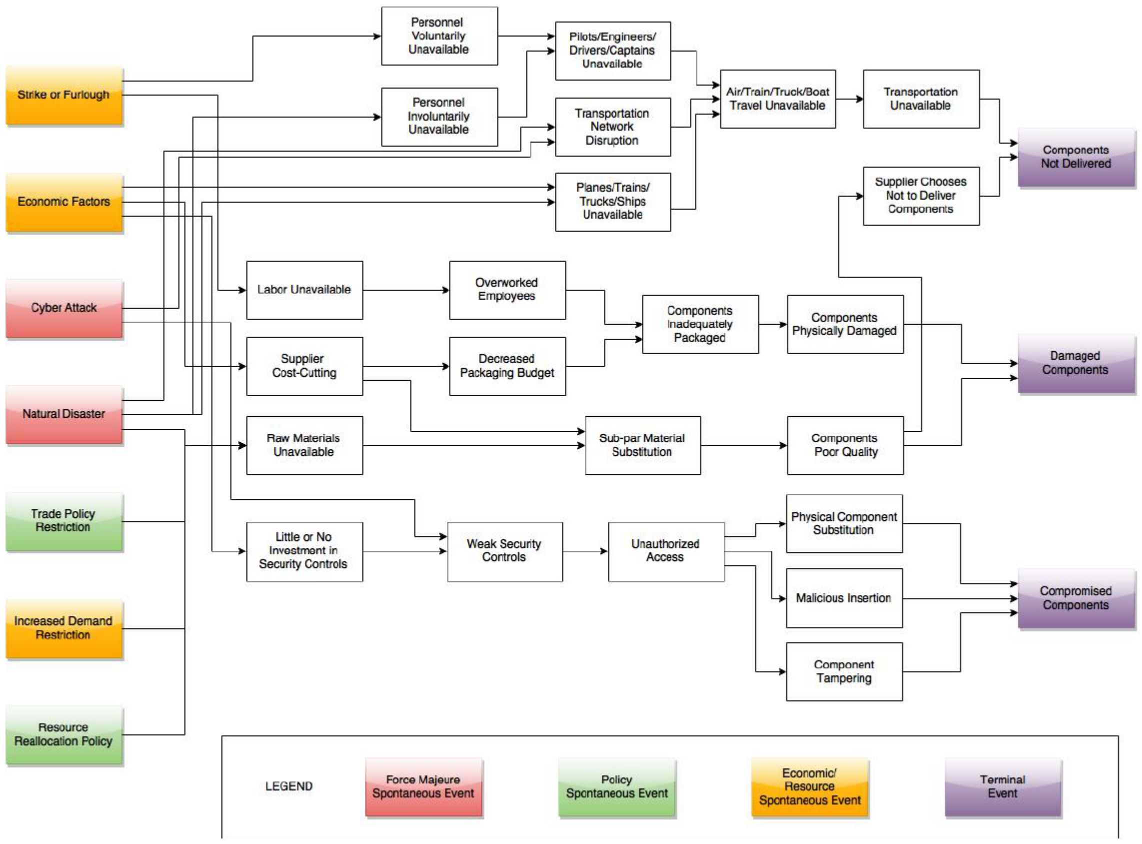

Strike or Furlough
Economic Factors
Cyber Attack
Natural Disaster
Trade Policy Restriction
Increased Demand Restriction
Resource Reallocation Policy
Personnel Voluntarily Unavailable
Personnel Involuntarily Unavailable
Pilots/Engineers/ Drivers/Captains Unavailable
Transportation Network Disruption
Planes/Trains/ Trucks/Ships Unavailable
Air/Train/Truck/Boat Travel Unavailable
Transportation Unavailable
Supplier Chooses Not to Deliver Components
Components Not Delivered
Labor Unavailable
Overworked Employees
Supplier Cost-Cutting
Decreased Packaging Budget
Components Inadequately Packaged
Components Physically Damaged
Damaged Components
Raw Materials Unavailable
Sub-par Material Substitution
Components Poor Quality
Little or No Investment in Security Controls
Weak Security Controls
Unauthorized Access
Physical Component Substitution
Malicious Insertion
Component Tampering
Compromised Components
LEGEND
Force Majeure Spontaneous Event
Policy Spontaneous Event
Economic/ Resource Spontaneous Event
Terminal Event

Figure 4. Example CEM of a supply chain. Image from (Rovito & Rhodes, 2016)

A CEM is not created for a system, but for a specific class of decision-maker. The hazards (referred to as “spontaneous events” 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 program managers may not be at fault for any of the vulnerabilities, it is still their 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.

THIS PAGE LEFT INTENTIONALLY BLANK

Vulnerabilities within Model-Centric Programs

Vulnerabilities within programs and program environments were investigated through literature and expert interviews.

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 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 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 to program managers respond to these vulnerabilities?
4. What vulnerabilities are present in MCE programs?

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, were present before the first program manager joined 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, however. Some 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.

Cybersecurity Vulnerability Assessment

Risk and vulnerability assessment methods have not failed to adapt to novel cybersecurity concerns. The aforementioned CVE database has been public since 1999. Quality Assurance testing (essentially the verification and validation of software) has been around since the beginning of commercial software. Software penetration testing (where security experts intentionally seek to break a software product) has been the industry norm for more than a decade (Arkin, Stender, & McGraw, 2005). Black-box mutational fuzzing and concolic execution are being used to automatically test for certain types of software vulnerabilities (Schwarz, 2018). Formal verification tools, initially limited to pure software domains such as cryptography (Meadows, 1994), has been rapidly advancing and finding applications in hardware (Kern & Greenstreet, 1999) and business processes (Morimoto, 2008), as well as fields that straddle the software-hardware-environment boundaries (Kamali, Dennis, McAree, Fisher, & Veres, 2016). The methods listed here just scratch the surface of approaches security researchers and engineers are taking to identify and resolve such technical cybersecurity vulnerabilities.

Beyond these specific testing methods, assessment frameworks have progressed as well. System-Theoretic Process Analysis (STPA) has adjusted, adapted, and been applied to handle cybersecurity vulnerabilities associated with additive manufacturing (Pope & Yampolskiy, 2016), Internet of Things (Pope, 2017), Air Operations (Young, 2013), and Mission Operations (Young & Porada, 2017). More recently, there have also been efforts to combine compiler technology with

STPA to automatically detect vulnerabilities in software-controlled systems (Pope, 2018).

While cybersecurity vulnerabilities in operational systems remain alarming common, from the trivial (Hanselman, 2012) to the critical (Gressin, 2017), there is some evidence that software is becoming more secure, at least in terms of defects per line of equivalent source code (Pope, 2017). In many cases, however, the acquisition or development process itself needs to be protected from outside threats and endogenous failures. Be it military information or technology-related trade secrets, there is real value in attempting to penetrate much earlier in the life cycle in order to either steal secrets (Hanna, Smythe, & Martin, 2018; Raymond, 2017) or to disrupt production (Statt, 2018) .

Defense acquisition programs have already instituted a variety of means of ensuring the security of their work. Some of these means were originally instituted to address other forms of threats but have turned out to be effective in addressing cybersecurity as well. These methods include relying on the security clearance process, the use of Sensitive Compartmented Information Facilities (SCIFs), restrictions on the use of media storage devices, separate networks such as SIPRNet and NIPRNet that are isolated or semi-isolated from the internet, and general compartmentalization of critical information. Some (non-US) defense agencies have gone so far as to revert to using typewriters where able in order to avoid security breaches and leaks (Irvine & Parfitt, 2013).

Unfortunately, many of these historically successful methods are in conflict with the more straightforward implementations of many components of an MCE environment. For example, the use of SCIFs has been quite successful in preventing unauthorized access to data. The typical use of a SCIF in design, where a small number of engineers work on a task isolated from the outside world, is not directly compatible with an MCE environment structured around model integration and collaboration across teams and locations. While this problem has been previously considered and ways to mitigate this conflict have been proposed (e.g., Reid & Rhodes, 2016), no silver bullet to resolving these tensions exists and it is likely that the increased use of MCE will result in both the exacerbation of current vulnerabilities and the creation of new ones. Furthermore, most means of assessing such vulnerabilities are aimed at assisting software and systems engineers to identify and remove cybersecurity vulnerabilities from the end system. New methods for enabling project and program managers to perform cybersecurity assessments of their enterprise and engineering environment are needed.

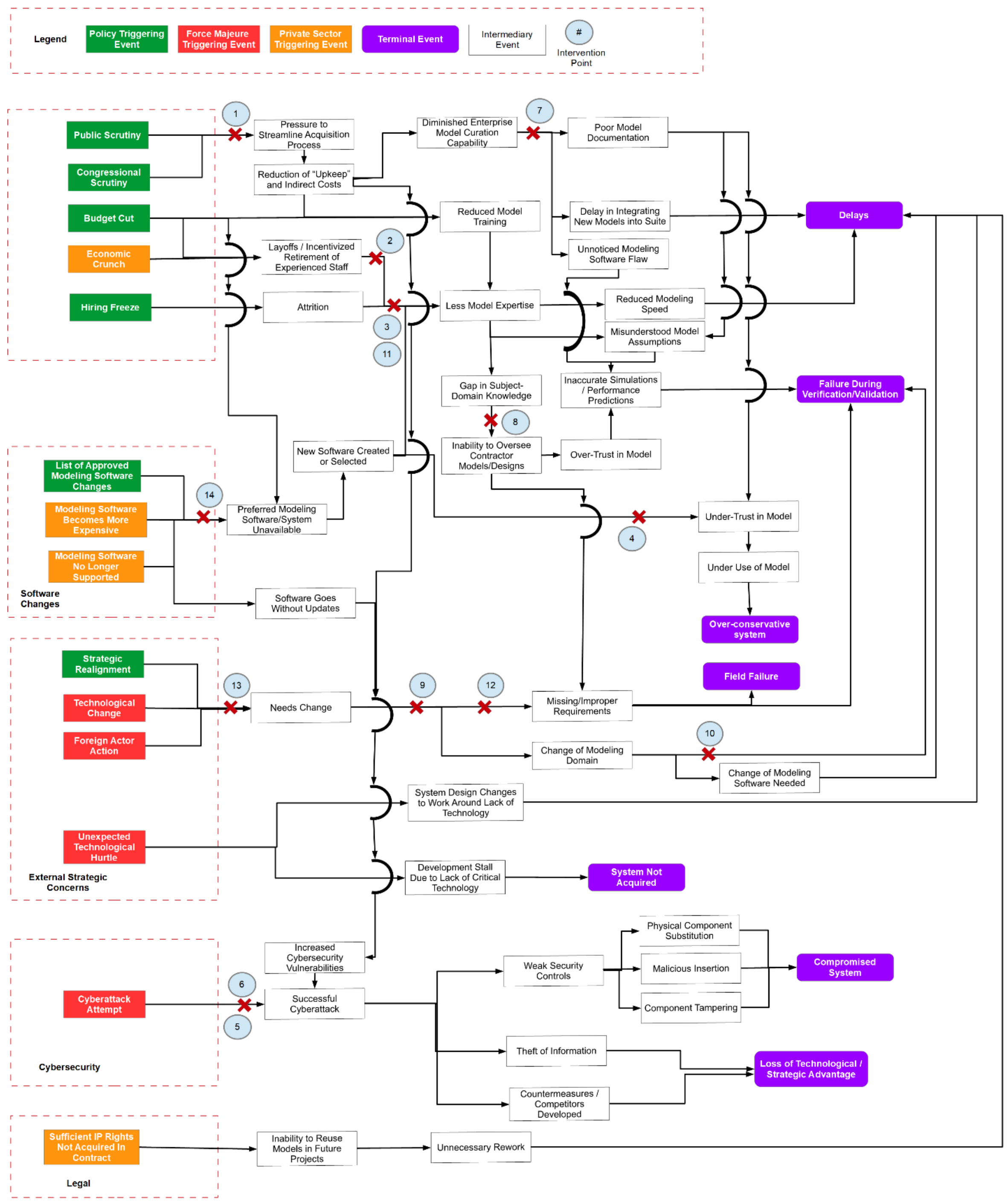
Methodology

A primary objective of this research 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 creating and using CEM in general, as well as conduct some initial usability testing of the usefulness of the Reference CEM.

The arrangement of the vulnerability chains is not random. 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 Figure 5 below. The external triggers were classified into three different domains.

- Force Majeure: This is a general term for an event that is the result of actions beyond the possibility of the program enterprise (not just the program manager) to influence. Thus it includes both malicious action and general, unforeseeable events such as Technological Change.
- Policy: An event that is the result of intentional decisions made at the organizational or enterprise level. In the case of a government-run program, this includes oversight from Congress and the general public. Non-government organizations may still be impacted indirectly by such oversight, but their proximal triggering event would be different.
- Private Sector: Any event that is the result of the actions of one or more private-sector firms outside the program enterprise.

Once the Reference CEM was created, intervention points were identified. These are indicated as the small blue circle in Figure 5.


Legend
Policy Triggering Event
Force Majeure Triggering Event
Private Sector Triggering Event
Terminal Event
Intermediary Event
#
Intervention Point
Public Scrutiny
Congressional Scrutiny
Budget Cut
Economic Crunch
Hiring Freeze
Pressure to Streamline Acquisition Process
Reduction of "Upkeep" and Indirect Costs
Diminished Enterprise Model Curation Capability
Poor Model Documentation
Reduced Model Training
Delay in Integrating New Models into Suite
Unnoticed Modeling Software Flaw
Delays
Layoffs / Incentivized Retirement of Experienced Staff
Attrition
Less Model Expertise
Reduced Modeling Speed
Misunderstood Model Assumptions
Gap in Subject-Domain Knowledge
Inaccurate Simulations / Performance Predictions
Failure During Verification/Validation
Inability to Oversee Contractor Models/Designs
Over-Trust in Model
New Software Created or Selected
List of Approved Modeling Software Changes
Modeling Software Becomes More Expensive
Modeling Software No Longer Supported
Software Changes
Preferred Modeling Software/System Unavailable
Software Goes Without Updates
Under-Trust in Model
Under Use of Model
Over-conservative system
Field Failure
Strategic Realignment
Technological Change
Foreign Actor Action
Needs Change
Missing/Improper Requirements
Change of Modeling Domain
Change of Modeling Software Needed
System Design Changes to Work Around Lack of Technology
Unexpected Technological Hurtle
External Strategic Concerns
Development Stall Due to Lack of Critical Technology
System Not Acquired
Increased Cybersecurity Vulnerabilities
Cyberattack Attempt
Successful Cyberattack
Weak Security Controls
Physical Component Substitution
Malicious Insertion
Component Tampering
Compromised System
Theft of Information
Loss of Technological / Strategic Advantage
Countermeasures / Competitors Developed
Cybersecurity
Sufficient IP Rights Not Acquired In Contract
Inability to Reuse Models in Future Projects
Unnecessary Rework
Legal
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Intervention Points

1) Initiate Internal Assessment and a PR Strategy
2) Initiate various non-monetary benefits (such as allowing pets at work or a 9/80 schedule) to encourage employees to stay
3) Seek to share resources and employees with other programs
4) Hire employees with prior experience with the new software
5) Actively conduct security training and monitor compliance
6) Isolate critical data
7) Create documentation and curation processes within the program
8) Institute handover periods to benefit from contractor expertise
9) Reevaluate training regime and needed fields of expertise
10) Increase the amount of testing conducted
11) Increase use of contractors/consultants (including previous employees) to maintain expertise level
12) Reevaluate requirements with the client and other stakeholders
13) Design for modularity to minimize impact on system
14) Negotiate with client to see if they are able to pay for the software

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

Generating the CEM

Generating a CEM can be done in different ways and to different levels of granularity, depending on the needs of the stakeholder (program manager, engineering leader, etc.). This process can be done with groups, such as project teams, as well as individually. The general process is:

1. The stakeholder lists potential hazards posed to the program.
2. The consequences of each of these hazards are traced through the intermediary events to the final terminal events.
3. The process is then done in reverse: the stakeholder looks at the terminal events, adds in any that are still missing, and works backwards on how they might come about.
4. The causal connections between each intermediary event are examined to see if there are any additional connections not previously noticed.
5. Finally, the stakeholder 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, 1994) are from prior to the rise of MCE and thus deal with general vulnerabilities. Existing case studies that directly deal with MCE tend to me 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. All of these were supplemented and confirmed by the expert interviews.

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 was 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 below:

1. By a Program Manager: Assessing potential future vulnerabilities and plan possible interventions.
2. By a Program Manager: Determining specific vulnerabilities to address in response to the presence of a specific hazard.
3. By the Program Organization: Change program processes to mitigate or eliminate vulnerabilities.
4. By MCE Researchers: Organize and classify vulnerabilities into various categories or types

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

- Identifying notable intermediary events and potential intervention points
- Conducting more detailed modeling of specific vulnerability chains
- 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 general benefit in using the Reference CEM for a model-centric program. 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, 1994). 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, the *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.

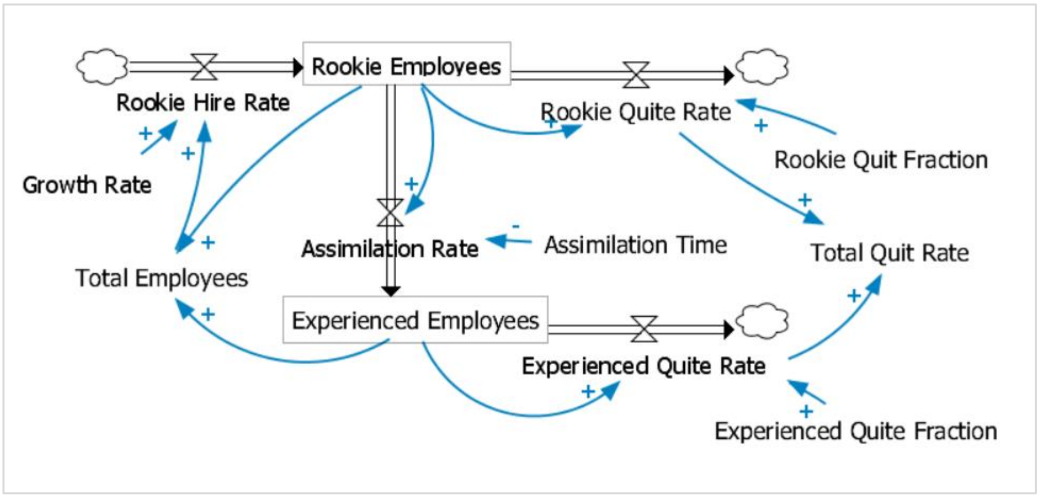


Figure 6. System Dynamics Model of Employee Training Rate. Adapted from (Sterman, 2000)

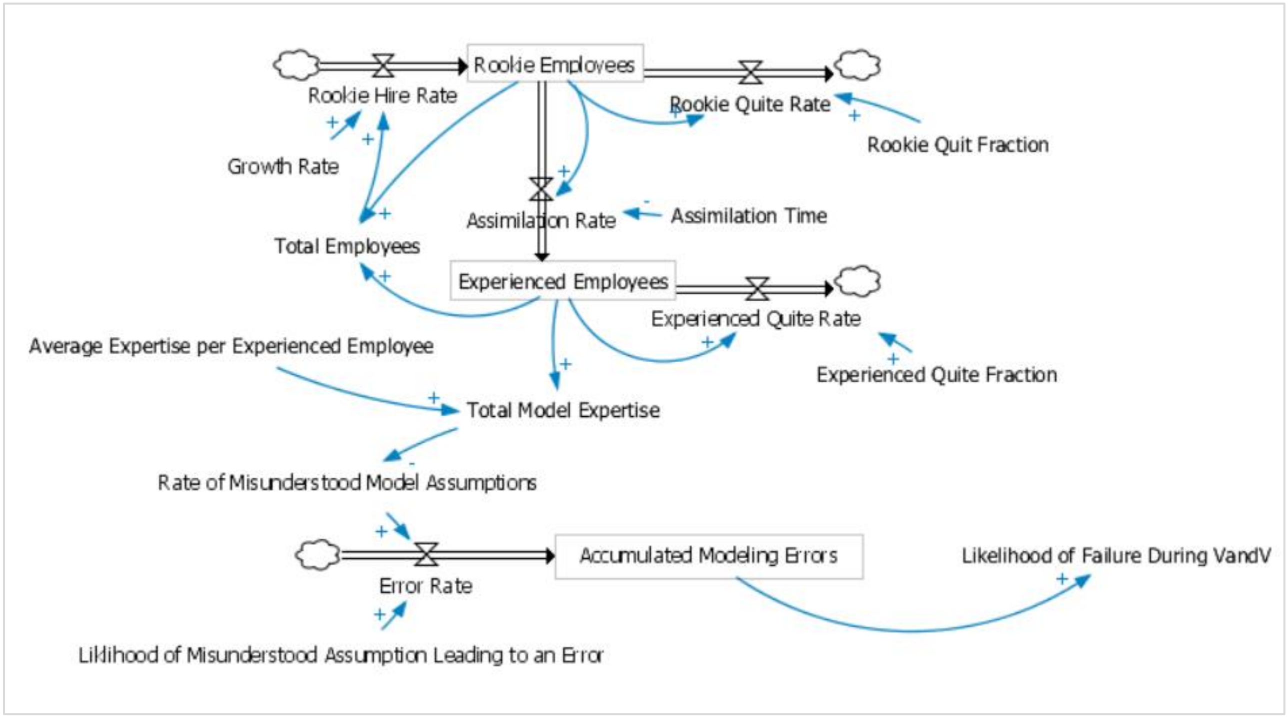

Rookie Employees
Rookie Hire Rate
Rookie Quite Rate
Rookie Quit Fraction
Growth Rate
Assimilation Rate
Assimilation Time
Total Employees
Experienced Employees
Experienced Quite Rate
Experienced Quite Fraction
Average Expertise per Experienced Employee
Total Model Expertise
Rate of Misunderstood Model Assumptions
Accumulated Modeling Errors
Likelihood of Failure During VandV
Error Rate
Liklihood of Misunderstood Assumption Leading to an Error

Figure 7. System Dynamics Model of Accumulated Modeling Errors

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 inhouse, 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, tabulated 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 forward 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.

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 results of a scenario-based exercise on vulnerability analysis using a data set that was previously generated in a graduate class activity involving techniques for investigating enterprises.

The class 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 5-7 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 US 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 1 or 2 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 are 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 of 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 your company 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 to use another set, 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.

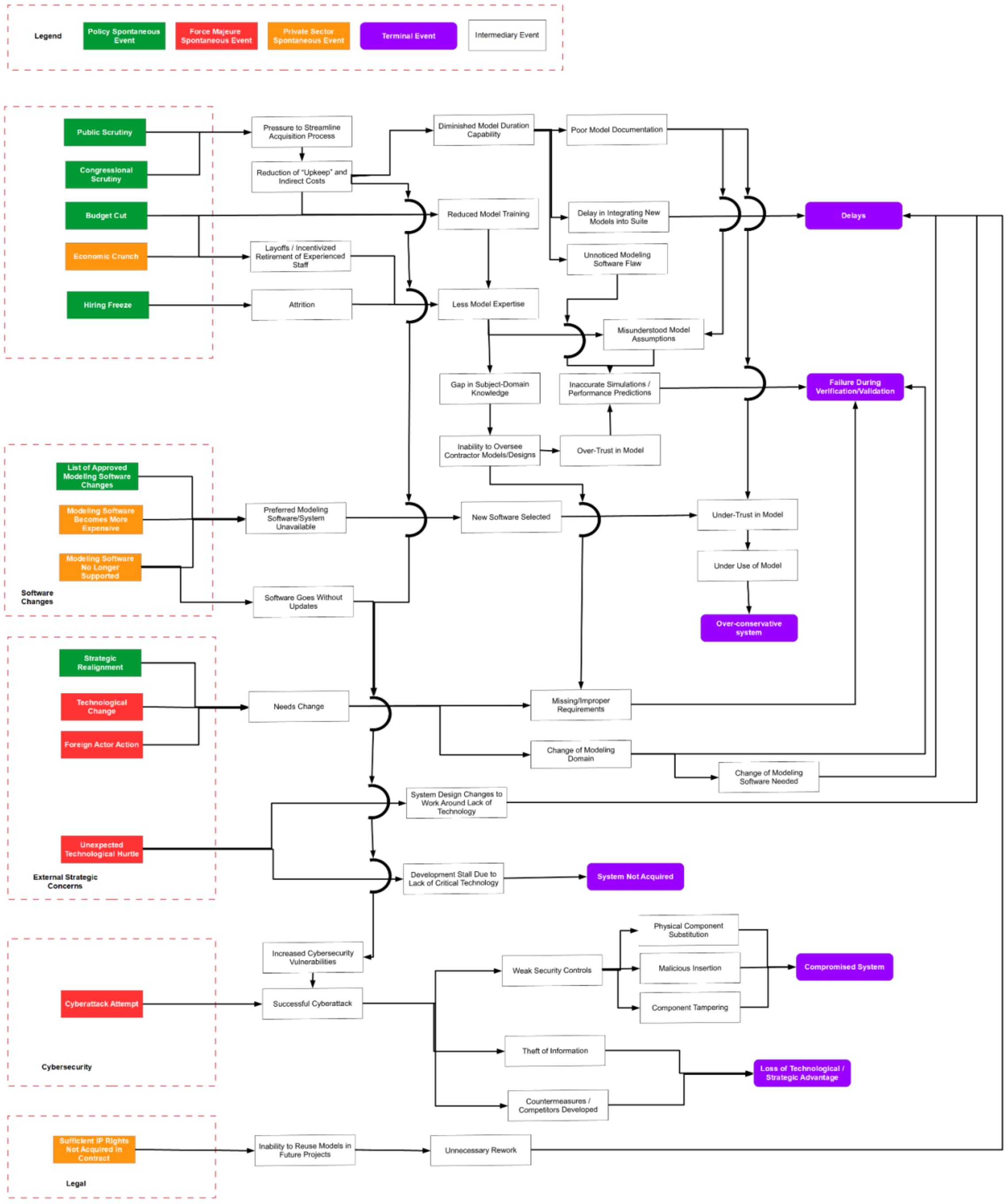
6. Another project at your company is threatening to miss its deadlines. In order to get it back in line, its project manager is requesting that personnel from other projects, such as yours, be reallocated to hers.
7. The design requirements from the military that you are currently working with were put together with the idea of using this vehicle in a currently ongoing conflict. However, US 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 requirements) may change during the design process. Furthermore, your models were created with 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 clearer, and most groups also identified additional interventions.



Legend
Policy Spontaneous Event
Force Majeure Spontaneous Event
Private Sector Spontaneous Event
Terminal Event
Intermediary Event
Public Scrutiny
Congressional Scrutiny
Budget Cut
Economic Crunch
Hiring Freeze
Pressure to Streamline Acquisition Process
Reduction of "Upkeep" and Indirect Costs
Layoffs / Incentivized Retirement of Experienced Staff
Attrition
Diminished Model Duration Capability
Poor Model Documentation
Reduced Model Training
Delay in Integrating New Models into Suite
Unnoticed Modeling Software Flaw
Less Model Expertise
Misunderstood Model Assumptions
Gap in Subject-Domain Knowledge
Inaccurate Simulations / Performance Predictions
Inability to Oversee Contractor Models/Designs
Over-Trust in Model
Delays
Failure During Verification/Validation
List of Approved Modeling Software Changes
Modeling Software Becomes More Expensive
Modeling Software No Longer Supported
Software Changes
Preferred Modeling Software/System Unavailable
Software Goes Without Updates
New Software Selected
Under-Trust in Model
Under Use of Model
Over-conservative system
Strategic Realignment
Technological Change
Foreign Actor Action
Unexpected Technological Hurtle
External Strategic Concerns
Needs Change
Missing/Improper Requirements
Change of Modeling Domain
Change of Modeling Software Needed
System Design Changes to Work Around Lack of Technology
Development Stall Due to Lack of Critical Technology
System Not Acquired
Cyberattack Attempt
Cybersecurity
Increased Cybersecurity Vulnerabilities
Successful Cyberattack
Weak Security Controls
Physical Component Substitution
Malicious Insertion
Component Tampering
Compromised System
Theft of Information
Countermeasures / Competitors Developed
Loss of Technological / Strategic Advantage
Sufficient IP Rights Not Acquired in Contract
Legal
Inability to Reuse Models in Future Projects
Unnecessary Rework

Figure 8. Reference CEM used as a basis for usability testing

THIS PAGE LEFT INTENTIONALLY BLANK

Findings

The transformation of acquisition programs to model-centric acquisition introduces vulnerabilities that either do not exist, or are less severe than those in past. While these vulnerabilities pose a challenge, they have thus far been overshadowed by addressing the difficulties of implementing MCE practices. This research has provided additional insight into what vulnerabilities may exist and how they may be identified.

In particular, viewing vulnerabilities as causal chains enables an enriched description and means to understand the complexities that are faced. Investigation suggests that program managers currently do not formally grapple with programmatic vulnerabilities as they do with vulnerabilities in the end-system. Many program managers mentally conceptualize vulnerabilities as discrete events with vague likelihoods. Conceptualizing vulnerabilities as causal chains instead encourages a more structured and detailed consideration of programmatic vulnerabilities. This work made it clear that conceptualizing vulnerabilities as causal chains is not only viable, but quite useful for improving the ability to recognize the similarities and connections between vulnerabilities as well as to identify potential means of intervention.

One of the objectives of this research was to develop Reference CEM for model-centric programs. The intent is for this CEM to serve as a both a collection of research findings and a reference for validation and further analysis. Additionally, it has the benefit of potentially serving 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. The Reference CEM 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, and lessons learned databases 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. 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 and expert interviews.   In practical use of the CEM process, program managers would create a more detailed CEM of their own MCE environment or use the Reference CEM as a general reference to better understand MCE-related vulnerabilities.

The Reference CEM provides an additional tool to program managers and researchers as an entry point to analyze MCE-related programmatic vulnerabilities in various contexts. Via the usability testing, the Reference CEM has been demonstrated to be useful for identifying and prioritizing interventions in MCE-related vulnerabilities. It can also be used to categorize vulnerabilities and promote understanding of the underlying connections between them.

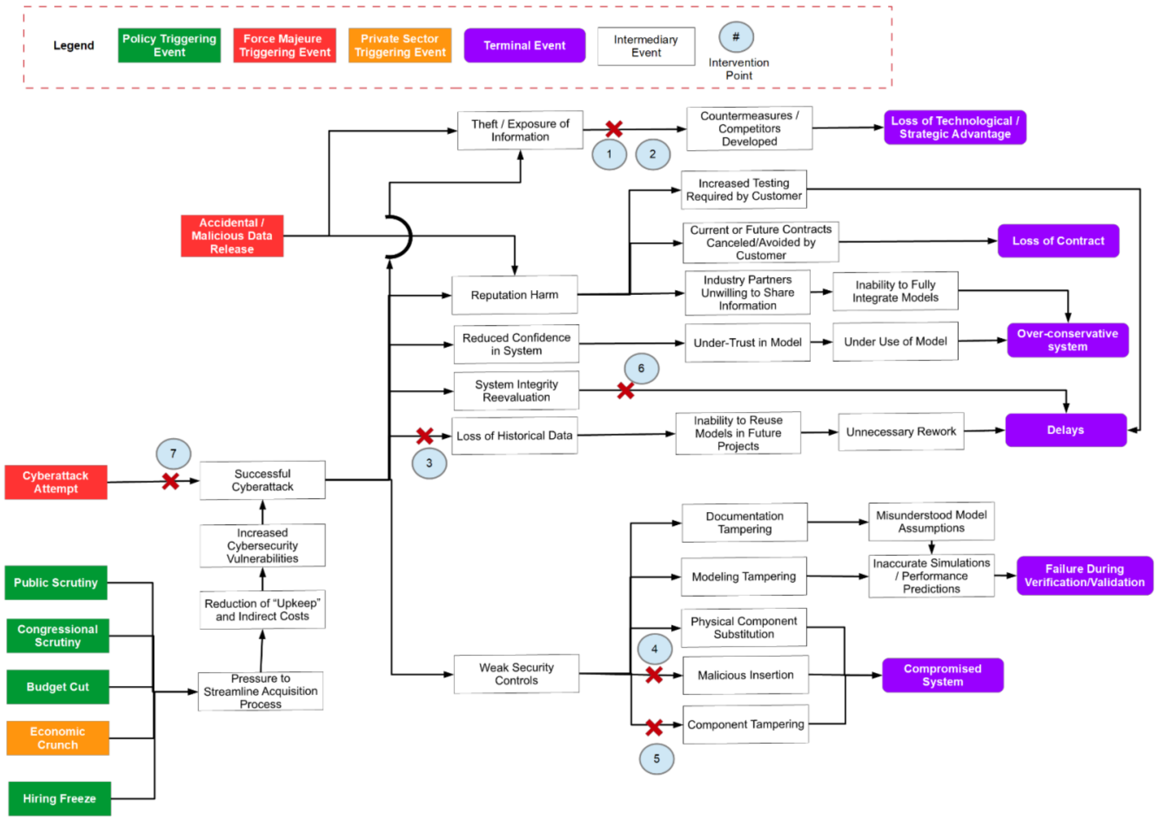
Expert interviews and usability testing were performed with program managers and systems engineers (the people who "live in" in the MCE environment). These led to various insights and confirmed the need for considering additional viewpoints from MBSE and MCE tool developers, as well as from the leaders of enterprise MCE environments. The former may yield additional potential MCE-related vulnerabilities that are 'invisible' to the users of the MCE tools, particularly on the cybersecurity front, and possibility identify ways to intervene in or preemptively eliminate some of the vulnerabilities identified in the Reference CEM. This would be particularly useful in filling in the "gap" of interventions and potentially enable the creation of new CEMs, aimed at enterprise leaders rather than program managers, as there are doubtlessly organizational vulnerabilities that threaten the success of acquisition programs just as there are programmatic vulnerabilities.

Various analytic approaches were considered during the research. These included network analysis, graph theory techniques, and use of system dynamics. This exploration reinforced the potential benefit of augmenting the use of cause-effect mapping with other analytic techniques to enrich the analysis. In particular, there is a need to better determine where to place intervention points in intertwined vulnerability chains. Simple in-degree/out-degree analysis is helpful, but more sophisticated applications of graph theory and probabilistic modeling offer greater potential benefit. These can be conducted using the Reference CEM if more information about a specific system is known. For instance, if probabilities, likelihoods, or time scales of each event transition are known, techniques such as Markov Chain Modeling, Monte Carlo Analysis, and Bayesian Networks can be brought to bear, weighting each arc of the graph instead of treating them equally.

Cybersecurity was selected as an area for additional consideration. The MCE Reference CEM shown in Figure 5 was generated using literature reviews and interviews with experts, among other sources. The cybersecurity portion of it was adapted, mostly unchanged, from previous work on supply chains (Rovito & Rhodes, 2016). As cybersecurity is a rising international concern and is particular relevant with the increasing digitization associated with MCE environments, the research team did some preliminary exploration of these aspects through interviews. When it came to the topic of cybersecurity vulnerabilities in general, the interviewees commonly raised the following issues:

- Cybersecurity needs to be thoroughly considered much earlier than it commonly is, preferably in the proposal generation stage.
- Program managers and systems engineers are sometimes intimidated by cybersecurity issues and thus seek to pass them onto specialists later in the acquisition process.
- MBSE and MCE toolset developers and proponents have not done a thorough enough job of considering programmatic cybersecurity vulnerabilities, though the tools are typically quite effective at designing for cybersecurity in end systems.
- Despite all of the above, according to the interviewees, traditional programmatic cybersecurity defensive practices tend to quite effective. This is due primarily to the conservative approaches most defense-related engineering groups use. The increased use of MCE, particularly for multi-site collaboration could change this.

The above points, many of which were commonly stated by the same expert, are clearly nuanced and complicated, with both points of success and failure. These points, along with more specific comments from the interviewees, resulted in an initial version of an expanded cybersecurity CEM that can be seen in Figure 9. Note that in its full form, this would still be a part of the general Reference CEM shown in Figure 5. Here it is shown isolated for clarity.


Legend
Policy Triggering Event
Force Majeure Triggering Event
Private Sector Triggering Event
Terminal Event
Intermediary Event
#
Intervention Point
Theft / Exposure of Information
Countermeasures / Competitors Developed
Loss of Technological / Strategic Advantage
1
2
Increased Testing Required by Customer
Current or Future Contracts Canceled/Avoided by Customer
Loss of Contract
Accidental / Malicious Data Release
Reputation Harm
Industry Partners Unwilling to Share Information
Inability to Fully Integrate Models
Reduced Confidence in System
Under-Trust in Model
Under Use of Model
Over-conservative system
6
System Integrity Reevaluation
Loss of Historical Data
Inability to Reuse Models in Future Projects
Unnecessary Rework
Delays
3
7
Cyberattack Attempt
Successful Cyberattack
Increased Cybersecurity Vulnerabilities
Reduction of "Upkeep" and Indirect Costs
Pressure to Streamline Acquisition Process
Public Scrutiny
Congressional Scrutiny
Budget Cut
Economic Crunch
Hiring Freeze
Documentation Tampering
Misunderstood Model Assumptions
Modeling Tampering
Inaccurate Simulations / Performance Predictions
Failure During Verification/Validation
Physical Component Substitution
4
Weak Security Controls
Malicious Insertion
Compromised System
Component Tampering
5

Intervention Points

1) Compartmentalize sensitive information
2) Obfuscate sensitive data with false or misleading information
3) Isolated but readily accessible back-ups of data
4) Reviews/Comparisons of models between lifecycle stages
5) Multiple, independent simulations or component checkers
6) Isolated, independent backup equipment that can be switched to while primary equipment is being evaluated
7) Conduct regular "red-team" / penetration test exercises

Figure 9. Reference Cybersecurity CEM (Preliminary)

Some of the vulnerabilities and interventions shown in Figure 9 are not unique to MCE environments. Some of the vulnerabilities will simply be exacerbated by increased use of MCE environments and processes. Some of the interventions will require new, creative means of implementing. For instance, Intervention Point #1 in Figure 9 is "Compartmentalize sensitive information." Clearly this is already done with the use of SCIFs and the Need-To-Know (NTK) information framework. However, such methods may not be feasible if the benefits of model integration and collaboration offered by MCE are desired. Instead new methods must be developed. An example of one such possibility is the Federal Drug Administration's (FDA) Sentinel Initiative, which involves querying a distributed system and receiving anonymized, aggregate data back (Office of Surveillane and Epidemiology, 2010). Such a system may allow modeling software to communicate across domains and locations, while still ensuring that even if one location is breached, only some information is exposed.

This Reference CEM does omit vulnerabilities and interventions that are entirely unchanged, however. For example, practices like the security clearance system and restricting the usual of digital storage media will remain effective interventions that are not significantly impacted by MCE environments.

One set of vulnerabilities that came up repeatedly in both the interviews and was observed in the class activity data set were those that passed through the reputation harm intermediate event (Figure 10). Despite the frequency that the potential for this vulnerability was raised, few interventions were proposed for post-breach. This suggests that program managers and systems engineers could use more training in how to respond to breaches, particularly prominent ones, instead of just how to prevent them. While in the private sector there is evidence suggesting that the reputation harm incurred by a prominent breach does not significantly impact the firm, contractors to the government are known to suffer significant financial penalties due to breaches, even when such a breach is unrelated to their government duties (Braun, 2014; Overly, 2017). In a defense acquisition environment, there is thus significant incentive to having program managers (and the enterprise as a whole) well prepared to respond to major breaches.

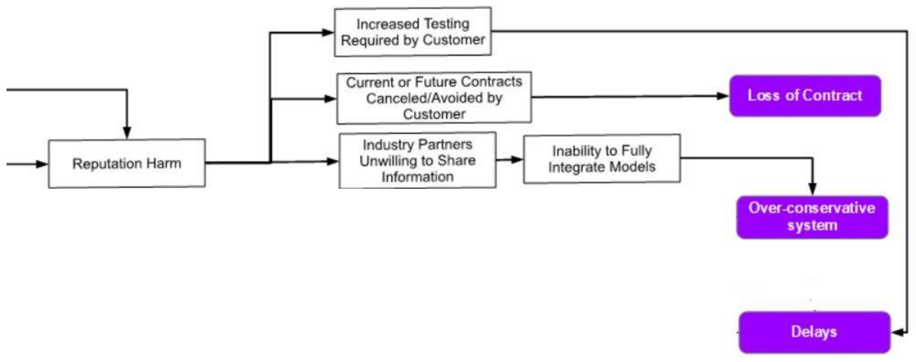

Increased Testing Required by Customer
Current or Future Contracts Canceled/Avoided by Customer
Loss of Contract
Reputation Harm
Industry Partners Unwilling to Share Information
Inability to Fully Integrate Models
Over-conservative system
Delays

Figure 10. Reputation Harm Vulnerabilities

CEM is intended to supplement, rather than replace, existing vulnerability assessment methods, particularly when it comes to cybersecurity. In this way, it can help fulfill the requirements set by NIST's Risk Management Framework (RMF) (Ross, et al., 2016) and the DoD's Defense Federal Acquisition Rules Supplement (DFARS) (Manufacturing Extension Partnership, 2017b). These regulations have shifted how government contractors handle cybersecurity. Previously, one-time assessments were completed and defensive practices instituted, now the process is more dynamic. Contractors have to continuously assess threats and develop countermeasures as they arise, both with regards to the end-system and to the enterprise. CEM can potentially assist in this, by serving as a reference that can be revisited as new threats arise.

Based on the exploratory work on cybersecurity vulnerabilities, a proposed second phase of this research is planned to investigate this topic in further detail.

Recommendations & 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 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.

As the research progress, three directions of future research are being pursued (Reid, 2018). The first is to conduct a second round of interviews with other stakeholders in the acquisition process. The second is to evolve an interactive version of the Reference CEM. The third is to compare vulnerabilities present in MCE environments with those present in other, comparable fields.

Future Interviews

The interviews thus far have been with program managers and system engineers (the people who “live in” in the MCE environment). As this research progresses, a future round of interviews with MBSE and MCE toolmakers and leaders of enterprise model-centric environments is desired. Several of the interviewees expressed an interest in increased enterprise-level ownership of MCE environments. Additionally, a few expressed concern about the degree of security in MCE toolsets. Additional work is needed to understand the varying perspectives on vulnerabilities in MCE environments. One of the benefits of future interviews with MCE tool makers would be the identification of additional cybersecurity vulnerabilities and interventions.

Interactive Tool

An interactive version of the CEM, which enables easy sorting and adding vulnerabilities, is desired. This would make the method more accessible, similar to how NIST’s Cybersecurity Assessment Tool (Manufacturing Extension Partnership, 2017a) makes the RMF (Ross et al., 2016) more approachable to small manufacturers. Additionally, it could serve as a platform for future usability testing of CEM in MCE programs. A desired future research task is to develop an interactive demonstration prototype that synthesizes the research outcomes to show how these can be used in practice.

Healthcare Industry Comparison

There is some indication that program managers may be well served by observing fields that are somewhat analogous to defense acquisition, in order to derive helpful metaphors (Karas, Moore, & Parrott, 2008) or lessons learned (German & Rhodes, 2016). The healthcare industry shows promise for such an analogy for cybersecurity in MCE environments. The healthcare industry deals with sensitive information, computer equipment, and high pressure environments. All of these are present at numerous stages of operation. Patient records have to be able to be transferred from one system to another and be available to medical practitioners. Researchers need to be able to query systems in order to provide improved medical treatment, but cannot violate individuals' privacy. They must do all this and more while under constant threat of cyberattack, as recent events have shown (Ryckaert, 2018; Woollaston, 2017; Zetter, 2016). Engineers and researchers have made significant headway in making medical devices more interoperable with one another, particularly when it comes to sharing data securely (Goldman, 2014). Increasingly, model-based methods are being used to assess and design medical systems (Pajic, Mangharam, Sokolsky, Arney, & Goldman, 2014). The FDA's Sentinel Initiative seeks to enable active querying of medical data while preserving individual privacy. All of these endeavors are strikingly similar to the challenges currently faced in defense acquisition. This suggests possible benefit in conducting a systematic comparison of the two fields. The healthcare industry, along with other fields, will be examined for potential metaphors and lessons learned that are applicable to understanding vulnerabilities in MCE environments.

Case Study Applications

Several organizations (e.g., SRPO, NAVAIR, the US Air Force, and JPL's Team X) are actively developing, testing, and applying MCE practices on their programs. A desired future activity is using the CEM process and/or the Reference CEM for the purpose of testing these in real-world contexts. This would enable gathering additional data that could contribute to a more detailed and robust Reference CEM, and contribute to validation of the framework.

THIS PAGE LEFT INTENTIONALLY BLANK

Conclusions

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 Reference CEM 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.

Research outcomes achieved in this effort include empirically-grounded vulnerabilities of model-centric programs and a cause-effect mapping reference model for identifying vulnerabilities and interventions. Preliminary investigation of cybersecurity vulnerabilities within model-centric environments uncovered specific instances, and confirmed the need for further study. The research outcomes also confirm the need to raise awareness of the importance of performing vulnerability analysis of model-centric enterprise practices and environments. Failing to uncover such vulnerabilities could ultimately jeopardize program success and lead to end-system failures.

THIS PAGE LEFT INTENTIONALLY BLANK

Appendix A (Reference CEM)

This appendix contains detailed information about the Reference CEM. Refer to Reid (2018) for more detailed images and descriptions.

Table A-1. Reference CEM Events and Descriptions (Part 1 of 4)

Event Name	Event Type	Event Description
Accidental/Malicious Data Release	External Trigger	Either accidentally or intentionally, some amount of sensitive data involving the program has been released to individuals or groups not cleared to access such information
Budget Cut	External Trigger	Current or projected funding for this program or for the organization as a whole is being reduced
Congressional Scrutiny	External Trigger	Congress has become increasingly concerned with the management or status of this program or of defense programs in general
Cyberattack Attempt	External Trigger	Some individual or group seeks to disrupt or surveil the program using a cyberattack
Economic Crunch	External Trigger	Reduced consumption, reduced willingness to lend, raising unemployment or other forms of economic recession or depression are occurring
Foreign Actor Action	External Trigger	A foreign actor has taken some significant action that impacts the intended use of the system that the program is acquiring
Hiring Freeze	External Trigger	The organization has ceased all new hiring for some period of time
List of Approved Modeling Software Changes	External Trigger	The organization maintains a list of software approved for use in programs and has changed (added and/or removed) certain software from this list
Modeling Software Becomes More Expensive	External Trigger	Some modeling software used by the program which is purchased or licensed from an external provider has become more expensive in the upcoming version/renewal
Modeling Software No Longer Supported	External Trigger	The maintainer/developer of some modeling software used by the program has ceased issuing updates and/or new versions
Public Scrutiny	External Trigger	The public and/or news media has become increasingly concerned with the management or status of this program or of defense programs in general
Strategic Realignment	External Trigger	The strategic interests of the client or other stakeholders in the program have changed
Sufficient IP Rights Not Acquired in Contract	External Trigger	A previous project with relevant components or systems that could be reused did not acquire sufficient intellectual property rights to make those components available for reuse
Technological Change	External Trigger	A significant new technology has been developed or put into application that impacts the program in some way
Unexpected Technological Hurtle	External Trigger	During the program, some desired technology either is unexpectedly unavailable or is taking an unexpectedly long time to develop

Table A-2. Reference CEM Events and Descriptions (Part 2 of 4)

Event Name	Event Type	Event Description
Attrition	Intermediary Event	Reduction in experienced staff and/or total staff available to the program
Change of Modeling Domain	Intermediary Event	The program must now model the system, subsystem, or component in a different environment or in a different manner than was previously done
Change of Modeling Software Needed	Intermediary Event	The program must now change the modeling software used
Component Tampering	Intermediary Event	A digital or physical component of the system is maliciously altered to harm the program
Countermeasures /Competitors Developed	Intermediary Event	Other individuals or organizations are able to either develop countermeasures to the system being acquired or developing competing systems
Current or Future Contracts Canceled /Avoided by Customer	Intermediary Event	The program is either cancelled by the client or future programs are never initiated
Delay in Integrating New Models into Suite	Intermediary Event	Increased time is required to integrate a new modeling software or model into the MCE environment
Development Stall Due to Lack of Critical Technology	Intermediary Event	A critical technology for the program is not available and there is not means of altering the design to circumvent the need for the technology
Diminished Enterprise Model Curation Capability	Intermediary Event	Reduced ability of the enterprise or program to track, integrate, document, and improve the modeling environment
Documentation Tampering	Intermediary Event	Documentation of assumptions or other important aspects of models is maliciously altered to harm the program
Gap in Subject-Domain Knowledge	Intermediary Event	Among program staff there is a lack of knowledge on a particular subject area of relevance to the program
Inability to Fully Integrate Models	Intermediary Event	One or more models in use by the program cannot be integrated into the MCE environment and thus remain "stove-piped"
Inability to Oversee Contractor Models/Designs	Intermediary Event	Program staff no longer have the ability to effectively understand and vet the models and/or designs provided by the contractor
Inability to Reuse Models in Future Projects	Intermediary Event	Models from previous or current programs are not available to be used in future programs
Inaccurate Simulations / Performance Predictions	Intermediary Event	Models and simulations do not accurately reflect real-world behavior or performance
Increased Cybersecurity Vulnerabilities	Intermediary Event	The program and/or MCE environment has become more susceptible to cyberattacks
Increased Testing Required by Customer	Intermediary Event	The client increases the amount of testing required during verification to demonstrate system readiness
Industry Partners Unwilling to Share Information	Intermediary Event	Subcontractors, suppliers, or other members of industry are unwilling to share information with the program due to concerns that it will be misused or inappropriately shared
Layoffs/Incentivized Retirement of Experienced Staff	Intermediary Event	Staff experienced with the MCE environment or having general relevant experience are laid off or incentivized to retire
Less Model Expertise	Intermediary Event	Among program staff there is either a reduced total amount of experience or reduced average amount of experience with the MCE environment

Table A-3. Reference CEM Events and Descriptions (Part 3 of 4)

Event Name	Event Type	Event Description
Loss of Historical Data	Intermediary Event	Data from current or previous programs has been lost or corrupted such that it is unavailable for future reference or reuse
Malicious Insertion	Intermediary Event	A new digital or physical object is inserted into the model or system with intent to harm system integrity
Missing/Improper Requirements	Intermediary Event	The current set of requirements do not sufficiently match stakeholder needs, either due to a lack of relevant requirements or due to a conflict between the requirements and needs
Misunderstood Model Assumptions	Intermediary Event	Some member of the program staff misunderstands or fails to consider one or more of the assumptions underlying the model in such a way as to potentially impact the program
Modeling Tampering	Intermediary Event	The models themselves are maliciously altered to harm the program
Needs Change	Intermediary Event	The needs of the client stakeholder(s) are changing in a way that impacts the program
New Software Created or Selected	Intermediary Event	A new modeling or simulation software must be created or selected (if COTS) and integrated into the MCE environment
Over-Trust in Model	Intermediary Event	One or more members of the program staff trust the current state of the model or results of simulation when these do not represent reality in some significant manner
Physical Component Substitution	Intermediary Event	A physical component in the system being acquired is substituted, threatening system integrity
Poor Model Documentation	Intermediary Event	The model documentation is missing relevant information or presented in an inaccessible manner
Preferred Modelling Software/System Unavailable	Intermediary Event	The preferred modeling software or system, either due to prior experience or particular relevance to the program, is unavailable for use in this program
Pressure to Streamline Acquisition Process	Intermediary Event	External pressure, either from the organization, client, or other stakeholder, is exerted on the program manager to minimize costs, timing, and "bloat."
Reduced Confidence in System	Intermediary Event	Members of the program, organization, or clients have reduced confidence in the ability of the MCE environment to be able to operate effectively and securely
Reduced Model Training	Intermediary Event	Reduction in training in the use of the MCE environment is available to program personnel
Reduced Modeling Speed	Intermediary Event	Time required to effectively use the modeling software is increased
Reduction of "Upkeep" and Indirect Costs	Intermediary Event	Reduction or elimination of "unessential" procedures, tests, and personnel. Particularly likely targets include any procedures aimed at improving reuse or other programs
Reputation Harm	Intermediary Event	The professional, political, or public reputation of the program, organization, or client has become significantly harmed
Software Goes Without Updates	Intermediary Event	Software important to the MCE environment goes without updates or patches for longer than ideal, resulting in lack of capability or security concerns
Successful Cyberattack	Intermediary Event	A cyberattack was not repelled or stopped until after it significantly impacted the program

Table A-4 Reference CEM Events and Descriptions (Part 4 of 4)

Event Name	Event Type	Event Description
System Design Changes to Work Around Lack of Technology	Intermediary Event	A critical technology for the program is not available and thus the design of the system must be altered in order to circumvent the need for the technology
System Integrity Reevaluation	Intermediary Event	The program or organization must reevaluate the integrity of the MCE environment and/or the system being designed to ensure that it has not been compromised
Theft/Exposure of Information	Intermediary Event	Sensitive or otherwise non-public information is either intentionally stolen or otherwise made available to those uncleared to possess it
Under-Trust in Model	Intermediary Event	One or more members of the program staff do not place do not believe the current state of the model or results of simulation when these do represent reality to a sufficient extent
Under-Use of Model	Intermediary Event	The program does not make full effective use of the MCE environment available
Unnecessary Rework	Intermediary Event	Work previously accomplished in the current program or a previous one must be done again when it should not have to be
Unnoticed Modeling Software Flaw	Intermediary Event	Some modeling bug, error, or other form of inaccuracy develops and goes undetected by the model curation staff and/or the program
Weak Security Controls	Intermediary Event	No or few security controls are in place to prevent physical or virtual security compromises
Compromised System	Terminal Event	The system is put into operation, but suffers from lack of integrity
Delays	Terminal Event	Acquisition of the program is delayed and/or increased costs incurred
Failure During Verification/Validation	Terminal Event	The system fails during verification and/or validation, prompting redesign to occur or cancellation of the program
Field Failure	Terminal Event	The system passes verification or validation but fails during operation
Loss of Contract	Terminal Event	The program is cancelled
Loss of Technological/Strategic Advantage	Terminal Event	The system does not provide the advantage or superiority that it was intended to
Over-conservative System	Terminal Event	The system is "over-engineered" and thus more costly and/or has reduced performance that should be possible
System Not Acquired	Terminal Event	The program is cancelled and the system is not acquired

References

Arkin, B., Stender, S., & McGraw, G. (2005). Software penetration testing. IEEE Security and Privacy, 3(1), 84–87. https://doi.org/10.1109/MSP.2005.23

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

Braun, S. (2014, September 10). OPM plans to terminate contracts with USIS. Federal News Radio. Retrieved from https://federalnewsradio.com/management/2014/09/opm-plans-to-terminate-contracts-with-usis/

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 O Materials and Manufacturing*, *2*(1), 298–309. Retrieved from http://www.sae.org

Department of Defense (Ed.). (2015). Glossary: Defense Acquisition Acronyms and Terms. Fort Belvoir, Virginia 22060-5565: Defense Acquisition University Press. Retrieved from https://dap.dau.mil/glossary/Pages/Default.aspx

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

Glaessgen, E. H., & Stargel, D. (2012). The Digital Twin paradigm for future NASA and US Air Force vehicles. In *53rd Struct. Dyn. Mater. Conf.* (pp. 1–14).

Goldman, J. M. (2014, November). Solving the Interoperability Challenge. IEEE Pulse. Retrieved from https://pulse.embs.org/november-2014/solving-interoperability-challenge/

Gressin, S. (2017). The Equifax Data Breach: What to Do. Retrieved March 27, 2018, from https://www.consumer.ftc.gov/blog/2017/09/equifax-data-breach-what-do

Hall, D. (2018). Risk Identification Challenge. In *INCOSE 2018 International Workshop*. Jacksonville, FL.

Hanna, J., Smythe, C., & Martin, C. (2018, January 24). China's Sinovel Convicted in U.S. of Stealing Trade Secrets. Bloomberg. Retrieved from https://www.bloomberg.com/news/articles/2018-01-24/chinese-firm-sinovel-convicted-in-u-s-of-trade-secret-theft

Hanselman, S. (2012). Everything's broken and nobody's upset. Retrieved March 27, 2018, from https://www.hanselman.com/blog/EverythingsBrokenAndNobodysUpset.aspx

Kamali, M., Dennis, L. A., McAree, O., Fisher, M., & Veres, S. M. (2016). Formal verification of autonomous vehicle platooning. Science of Computer Programming, 1, 1–19. https://doi.org/10.1016/j.scico.2017.05.006

Karas, T. H., Moore, J. H., & Parrott, L. K. (2008). Metaphors for Cyber Security. Sandia Report. Albuquerque, NM: Sandia National Laboratories. Retrieved from http://evolutionofcomputing.org/Multicellular/Cyberfest Report.pdf

Kellner, T. (2015). Wind in the Cloud? How the Digital Wind Farm Will Make Wind Power 20 Percent More Efficient. *GE Reports*, *2015*(October 23). 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.

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 2008, Vols 1-4*, 1042–1050\r2248.

Meadows, C. A. (1994). Formal verification of cryptographic protocols: A survey. In International Conference on the Theory and Application of Cryptology (pp. 133–150). Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0000430

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

Morimoto, S. (2008). A Survey of Formal Verification for Business Process Modeling (pp. 514–522). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-69387-1_58

NASA Office of the Chief Engineer. (1994). NASA Public Lessons Learned System. Retrieved July 13, 2017, from https://llis.nasa.gov/

Office of Surveillane and Epidemiology (Ed.). (2010). The Sentinel Initiative.

Overly, S. (2017, October). IRS temporarily suspends contract with Equifax. Politico. Retrieved from https://www.politico.com/story/2017/10/12/irs-equifax-contract-suspended-243732

Pajic, M., Mangharam, R., Sokolsky, O., Arney, D., & Goldman, J. (2014). Model-driven safety analysis of closed-loop medical systems. IEEE Transactions on Industrial Informatics, 10(1), 3–16. https://doi.org/10.1109/TII.2012.2226594

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

Pope, G. (2017). A Hazard Analysis Technique for the Internet of Things (IoT) and Mobile. In STAMP Workshop. Cambridge, MA.

Pope, G. (2018). Combining STPA with Compiler Technology to Identify Vulnerabilities and Hazards in Software-Controlled Systems. In STAMP Workshop. Cambridge, MA.

Pope, G., & Yampolskiy, M. (2016). A Hazard Analysis Technique for Additive Manufacturing. In Better Software East Conference. Orlando, FL. Retrieved from https://arxiv.org/ftp/arxiv/papers/1706/1706.00497.pdf

Puchek, B., Bisconti, M., Zimmerman, P., Guerrero, J., Kobryn, P., Kukkala, G., … Mulpuri, M. (2017). Digital Model-based Engineering: Expectations , Prerequisites , and Challenges of Infusion. Washington D.C.: Office of the Deputy Assistant Secretary of Defense. Retrieved from https://www.acq.osd.mil/se/docs/DMbE-20170524-0602.pdf

Raymond, N. (2017, August 31). U.S. charges Chinese-Canadian citizen with trade secret theft. Reuters2. Retrieved from https://ca.reuters.com/article/topNews/idCAKCN1BB2K8-OCATP

Reid, J. B., & Rhodes, D. H. (2016). Digital System Models : An investigation of the non-technical challenges and research needs. In *Conference on Systems Engineering Research*. Huntsville, AL.

Reid, J. B., & Rhodes, D. H. (2018). Assessing Vulnerabilities in Model-Centric Acquisition Programs Using Cause-Effect Mapping. In 15th Annual Acquisition Research Symposium. Monterey, CA: Naval Postgraduate School.

Reid, J. B., & Rhodes, D. H. (2018). Applying Cause-Effect Mapping to Assess Cybersecurity Vulnerabilities in Model-Centric Acquisition Program Environments In 15th Annual Acquisition Research Symposium. Monterey, CA: Naval Postgraduate School.

Reid, J.B., Assessing and Mitigating Vulnerability Chains In Model-Centric Acquisition Programs, MIT Master's Thesis, June 2018.

Ross, R., Dempsey, K., Pillitteri, V. Y., Jacobs, J., & Goren, N. (2016). Risk Management. Retrieved March 29, 2018, from https://csrc.nist.gov/projects/risk-management/risk-management-framework-(RMF)-Overview

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 *2016 Annual IEEE Sytems Conference, Proceedings*. Orlando: IEEE.

Ryckaert, V. (2018). Hackers held patient data ransom, so Greenfield hospital system paid $50,000. The Indianapolis Star. Retrieved from https://www.indystar.com/story/news/crime/2018/01/17/hancock-health-paid-50-000-hackers-who-encrypted-patient-files/1040079001/

Schwarz, E. (2018). Automating Vulnerability Discovery in Critical Applications. Retrieved March 27, 2018, from https://www.sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=6487

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

Statt, N. (2018, March). Boeing production plant hit with WannaCry ransomware attack. The Verge. Retrieved from https://www.theverge.com/2018/3/28/17174540/boeing-wannacry-ransomware-attack-production-plant-charleston-south-carolina

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.

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

Virani, S., & Rust, T. (2016). Using Model Based Systems Engineering in Policy Development : A Thought Paper. In *Conference on Systems Engineering Research*. Huntsville, AL.

West, T. D., & Pyster, A. (2015). Untangling the Digital Thread: The Challenge and Promise of Model-Based Engineering in Defense Acquisition. *INSIGHT*, *18*(2), 45–55. https://doi.org/10.1002/inst.12022

Woollaston, V. (2017, May). The NHS trusts and hospitals affected by the Wannacry cyberattack. *Wired*. Retrieved from http://www.wired.co.uk/article/nhs-trusts-affected-by-cyber-attack

Young, W. E. (2013). A System Saftey Approach to Assuring Air Operations Against Cyber Disruptions. In *STAMP Workshop*. Cambridge, MA.

Young, W. E., & Porada, R. (2017). System-Theoretic Process Analysis for Security (STPA-SEC): Cyber Security and STPA. In *STAMP Workshop*. Cambridge, MA.

Zetter, K. (2016, March). WHY HOSPITALS ARE THE PERFECT TARGETS FOR RANSOMWARE. *Wired*. Retrieved from https://www.wired.com/2016/03/ransomware-why-hospitals-are-the-perfect-targets/