A Systems-Theoretic, Model-Based Methodology for Identifying and Evaluating Resiliency Strategies for Cyber-Physical Systems

---

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment
of the requirements for the degree

Master of Science

by

Bryan T. Carter

December 2018

APPROVAL SHEET

This Thesis
is submitted in partial fulfillment of the requirements
for the degree of
Master of Science

Author Signature: 

This Thesis has been read and approved by the examining committee:

Advisor: Cody Fleming

Committee Member: Peter Beling

Committee Member: Barry Horowitz

Committee Member: _____

Committee Member: _____

Committee Member: _____

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, School of Engineering and Applied Science

December 2018

Abstract

Despite their name, cyber-physical systems (CPS) possess unique characteristics that limit the applicability and suitability of traditional cybersecurity techniques and strategies. While software systems can be secured using defensive measures, the physical and component interactions inherent to CPS require them not only be defended against, but to also be resilient to cyber vulnerabilities. Given the complex nature of CPS, the identification and evaluation of appropriate resiliency strategies must be handled in a targeted and systematic manner. Specifically, what resiliency strategies are appropriate for a given system, where, and which should be implemented given time and/or budget constraints? This thesis presents a systems-theoretic, model-based methodology for identifying and prioritizing appropriate resiliency strategies for implementation in a given CPS and mission. This methodology is demonstrated using a case study based on a hypothetical US Army weapon system. A comparison of the results to the Cyber Security Requirements Methodology (CSRM) suggest that the technique presented in this thesis can augment and enhance existing techniques with model-based evidence.

Acknowledgements

I would like to begin my many thanks with Dr. Peter Beling. I would not be where I am today if Professor Beling did not help me find my way onto this project; I am grateful for his kindness and support from the end of my undergraduate studies through the completion of my Master's degree. An equally large thank you goes to Dr. Cody Fleming for his support throughout my graduate studies. I never would have gotten through writing my proposal, let alone my thesis, without the guidance of Professor Fleming- advising meetings with you were always something to look forward to. A big thank you goes to Dr. Barry Horowitz for chairing my thesis committee and for letting me learn so much from the stories about your experiences at MITRE and as a professor. Thank you to the entire Systems Engineering Department; it was an absolute pleasure working with and getting to know you all. I would also like to extend a special thank you to Dr. Kathryn Neeley, who has been kind enough to let me TA for her STS classes for the duration of my graduate studies. It has been a pleasure working with you- who knew you could ever learn so much on the other side of the classroom. I would also like to acknowledge the Systems Engineering Research Center and the US Army ARDEC for supporting the projects on which my work is based. One final thank you goes out to all my friends and family- thank you to my parents for supporting me in this latest step of my life, thank you to my friends for all the laughs and good times, and thank you, Lucy for being my rock and making sure I stay on track.

Table of Contents

1. Introduction

Cyber-physical systems (CPS) utilize a variety of networks, hardware, and software components to control, interact with, or influence a physical process in the real world. As one might expect, adversaries may attempt to affect those physical processes via attacking or exploiting the cyber component of CPS. Therefore, securing these systems against adversarial actions becomes a critical step to ensuring system integrity and functionality.

Generally, security for cyber-physical systems refers to the application of defensive and resilience measures implemented to help sustain acceptable levels of operation in the face of adversarial actions. More specifically, defensive measures are the steps taken to prevent an adversary from disrupting, terminating, or taking over control of the operation of a system. Resilience, on the other hand, refers to the actions taken to ensure the continuity of the system's expected service in spite of adversarial actions. Methodologies and techniques for achieving enhanced system security in the cyber domain are prevalent and well-researched, and have been applied to CPS as well. However, the methods and techniques used for enhancing the security of strictly cyber systems are not sufficient for CPS due to their lack of focus and inability to account for the physical interactions inherent to CPS and the system's usage in a broader mission.

The methodology presented in this thesis aims to combat these issues by using Systems Theory and Model-based Systems Engineering to identify and recommend appropriate resiliency solutions for implementation.

1.1 Motivation

Typically, security is handled in the post-design phase of a system's lifecycle. Security needs and solutions are developed based on the perceived threats to the system and potential vulnerabilities identified in the system's design. Often, this means that vulnerabilities are only

discovered after a security breach [1, 2] or after detrimental effects have already occurred [3]. Of course, it is impossible to discover all potential vulnerabilities, and systems must be routinely updated and adapted as new threats emerge; however, new trends call for integrating security into all parts of the system's lifecycle, including the pre-design and design phases [4]. Research shows that efforts and decisions made in the early parts of a system's life cycle have both decreased costs due to design changes as well as increased ability to impact the performance of a system. It is logical that cybersecurity efforts would follow a similar pattern and it therefore becomes advantageous to address security from the beginning of a system's life.

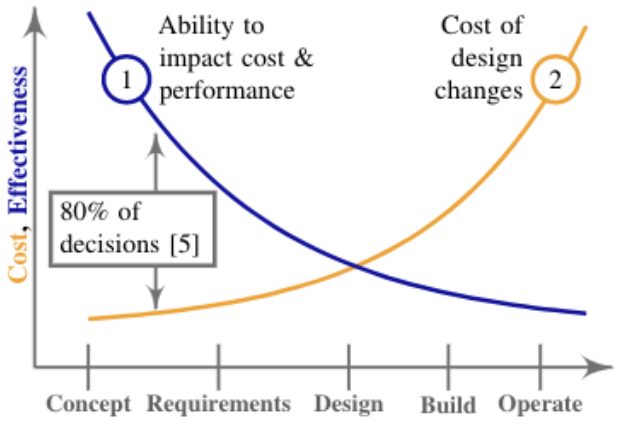

Figure 1. Decision effectiveness and cost against system life cycle stages (from [5]).

Since the typical cybersecurity approach relies on having a design to augment or take advantage of to achieve enhanced security, addressing security earlier in a system's life cycle necessitates the formulation of a new methodology capable of handling the lack of a final system design. Consequently, such an approach must examine the system's purpose, requirements, and expected service- or mission- to identify security needs and potential security solutions.

The goals of such a methodology mirror the needs of safety-critical systems. Particularly within the aerospace industry, designing in safety from the beginning of a system's life cycle has been standard operating procedure for decades. In CPS, security is intertwined with safety due to

inherent computer-controlled physical interaction with the real world. In some cases, design features or operating procedures intended to increase safety result in secure architectures. Consequently, it makes sense to draw upon these methodologies created to generate safe systems for the purpose of creating secure designs. In particular, the systems-theoretic models and analysis tools developed by Leveson and her group at MIT, such as the Systems Theoretic Accident Model and Process (STAMP) [6] and Systems Theoretic Process Analysis (STPA) [7], show promise in aiding the generation of requirements and understanding the behavior of CPS with respect to safety and security. Furthermore, a system in the preliminary stages of design will not have yet specific hardware and software components defined. A control-theoretic approach to defining the system behavior enables analysis without being gnostic to specific hardware and software implementations.

This is not enough, however, identifying appropriate security solutions for CPS requires analysis to determine the compatibility of a solution with the system's mission and control structure and its effect on behavior. Model-based approaches that simulate system behavior enable compatibility testing for resiliency solutions. Furthermore, the simulation allows for the solution's effects on system behavior to be measured. Consequently, different solutions can be assessed against each other to generate recommendations for which solutions should be implemented.

1.2 Purpose, Scope, and Contributions

The purpose of this thesis is to present a methodology for systematically identifying and analyzing alternative security solutions for CPS in a mission context. This methodology supports the Mission-Aware framework for Cyber-Physical System Security developed by University of Virginia, Virginia Commonwealth University, and the Software Engineering Institute and funded

through the Systems Engineering Research Center. The techniques presented in this thesis outline the development of mission and system definitions and specifications following systems-theoretic principles and use model-based systems engineering to analyze system behavior to make informed, data-driven assessments of appropriate resiliency solutions.

This methodology, referred to as the Systems-Theoretic Resiliency Assessment Tool (STRAT), can be used in conjunction with other techniques, such as the Cyber Security Requirements Methodology (CSRM) [29], to augment and enhance their findings and recommendations or it can be used on its own. This new approach is tested on a case study involving the hypothetical US Army weapon system developed for the CSRM. The results of both methodologies are compared to assess the validity of the STRAT as a viable tool for identifying and recommending resilience solutions for a CPS. The methodology describes the formulation of a graphical system model borrowing concepts from graph theory, however, further work on this topic is needed to define the formalisms associated with the graph structure. This work could be further expanded to also include cost analysis of resilience solutions and techniques for automating the model-based analysis.

The major contribution of this work is the potential for increased scalability of cybersecurity efforts to complex systems. The findings of this thesis present similar results to the CSRM, but required significantly less time and manpower. This allows experts to be used for confirming results rather than producing them, which can reduce project costs for expert time and increase expert productivity. The techniques presented in this work could allow for increased automation in security analysis, which would further increase the effectiveness and productivity of a traditionally labor-intensive process. A second contribution of this work is the creation of a common resilience management process that could be used in the development of new systems

in multiple industries. Having such a methodology would allow for earlier integration of operational test and evaluation in the design process as well as simplifying the design audit and review process.

2. Background & Literature Review

This section reviews the state-of-the-art for cybersecurity practices with a focus on CPS, systems-theoretic safety and security analysis methods, and previous work upon which this thesis is based.

2.1 Requirements Engineering

Requirements engineering (RE) is the process of eliciting, documenting, analyzing, and communicating the requirements for a particular system [8]. The practices and techniques for generating requirements are numerous and vary greatly between, and even within, industries. While requirements engineering has its role in every engineering project, the software and aerospace industries have some of the most researched and well-developed RE practices. This fact is not surprising; as Leveson [9] states, flawed requirements are the most common cause of errors and software-related accidents. Therefore, considerable time and effort is devoted to generating complete, consistent, and clear requirements.

In general, requirements engineering methodologies follow a four-part pattern defined by Kontonya and Somerville [10]: requirements elicitation, requirements analysis and negotiation, requirements documentation, and requirements verification and validation. Elicitation refers to the generation of requirements by engaging the relevant system stakeholders. Analysis typically involves requirement modeling such that the abstract descriptions from elicitation are interpretable. Documentation provides the vehicle for communication and management throughout the system life cycle. Validation and verification (V&V) attempts to ensure that the

requirements are both appropriate for the system and reflect the wishes and goals of the stakeholders accurately.

Requirements elicitation techniques vary from simple interviews with stakeholders to model-driven and goal-based methods. Nuseibeh and Easterbrook [8] distinguish several different classes of elicitation techniques including group elicitation techniques, prototyping, model-driven, cognitive, and contextual. The choice of elicitation technique is typically driven by the amount of time and resources available to the engineering team, as well as any established practices within their organization. Despite the various techniques available to requirements engineers, the ultimate goal of each elicitation technique is to facilitate the transfer of knowledge and goals from the clients or stakeholders to the engineering team. For example, the traditional techniques described by Nuseibeh and Easterbrook are simply surveys, questionnaires, or interviews with the stakeholders and clients. On the other hand, contextual elicitation techniques use ethnomethodology and conversation analysis in the attempt to extract patterns in conversation and interactions with stakeholders that may elucidate goals or hidden requirements that are difficult to articulate using traditional techniques [8].

Nuseibeh and Easterbrook go further to distinguish requirements elicitation techniques from requirements elicitation methods [8]. The elicitation techniques are simply the mechanism to extract information from the clients and stakeholders, while requirements elicitation methods provide guidance on how to best use elicitation techniques. For example, CREWS (Co-operative Requirements Engineering With Scenarios) is a method developed to semi-automatically generate use cases from which engineers and stakeholders can more quickly derive and validate software requirements. It should be noted, however, that using a complete requirements elicitation methodology is not necessary and a particular set of elicitation techniques will suffice.

In such scenarios, there exist technique-selection guidance frameworks to help engineers choose the appropriate elicitation technique for their particular system [11].

Requirements modeling and analysis aims to organize and represent requirements generated from the elicitation process in a more interpretable form. Unlike the requirements elicitation process, however, the goals of different requirements modeling techniques are not always the same. For some applications, particularly in software requirements engineering, modeling is used to organize and define a structure such that formal methods or other V&V techniques can be used to validate the requirements. Alternatively, some requirements modeling techniques are used to graphically present the elicited requirements and any relationships that might exist between them.

Nuseibeh and Easterbrook summarize several requirements modeling approaches, including enterprise modeling, data modeling, behavioral modelling, and domain modeling [8]. Some of these approaches are not specific to requirements modeling, such as enterprise modeling. Instead, these approaches take a holistic view to abstractly represent the purpose of the system and the behavior of the environment or organization in which the system operates [12]. In the case of enterprise modeling, the high-level goals or objectives of the system of larger organization can be used as a basis from which to elicit operational requirements and their relationships to those higher-level goals [13]. Similarly, data, behavioral, and domain modeling outline the relationships, interactions, and environments associated with a system and its stakeholders [8].

Modeling in the requirements engineering process is also aided by specific requirements modeling languages such as the Unified Requirements Modeling Language (URML) and the Unified Modeling Language (UML). These ontologies define specific syntaxes and model

grammar that succinctly and precisely describe the requirements, goals, and features of system and the relationships between those entities [14]. Modeling languages also enable traceability between requirements- a crucial characteristic for determining cascading effects from changes to the model.

2.2 Security Requirements

More specific to the software community, security requirements outline how a system shall prevent itself from being exploited by adversaries or mitigate the effects of adversarial actions. The International Organization for Standardization published the ISO/IEC 27000-series for best practice recommendations on information security management [15]; however, this applies more so to organizations managing their existing systems instead of the development of new systems. For new systems, Sindre and Opdahl [16] developed the concept of misuse cases, used to elicit security requirements. Misuse cases are the negative of use cases; rather than representing how a user may interact with a system, a misuse case represent how an adversary may attempt to compromise a system.

Outside of security requirements elicitation, Haley et al [17] developed a comprehensive framework for security requirements engineering. This framework follows the basic, 4-part pattern defined by Kontonya and Somerville [10], but with adjustments made to cater specifically to developing secure software systems. Haley et al [18] also outlined a method for validating security requirements using structured argumentation. This work explains how satisfaction arguments can show that a system meets its security requirements. However, like the validation and verification techniques based on risk analysis described by Fenz and Ekelhart [19], these methods do not provide mathematical proof of requirement satisfaction like formal

methods can for software. The lack of formal proof can be forgiven though due to the nature of security- mainly the unpredictability of attackers and zero-day vulnerabilities.

Security requirements engineering also typically involves threat modeling and analysis efforts to guide the choice of appropriate countermeasures. STRIDE (spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege) is a commonly used framework to help guide threat analysis by categorizing the main types of attacks typically targeted towards software systems [20]. More specifically, Shostack [20] describes STRIDE as an elicitation technique to help discover threats to a system. Shostack also summarizes attack trees as an alternative to STRIDE to both find threats and organize threats found via other methods. Attack trees were developed by Scheier [21] as a way to "provide a formal, methodical way of describing the security of systems … [by representing] attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as the leaf nodes." Finally, Mead et al [22] describe a crowd-sourcing approach to Personae non Gratae threat modeling, which outlines the attack goals and techniques of unwanted or malicious users to aid in the development of security requirements.

2.3 Systems-Theoretic Hazard Analysis

As previously stated, traditional security requirements techniques are insufficient for CPS due to the physical and component interactions inherent to such systems. For example, in contrast to purely software systems where an attacker may aim to steal information, the physical processes inherent to CPS create the possibility of attackers creating hazardous conditions that could harm humans or other resources. Leveson [6] developed the systems-theoretic accident model and process (STAMP) as a method to analyze system accidents. STAMP asserts that

safety be viewed as a control problem and that accidents result from an inadequate control structure rather than a component failure.

The inability of older hazard analysis techniques to handle the causal factors identified by STAMP prompted the development of Systems Theoretic Process Analysis (STPA) to aid in creation of safer systems [7]. STPA reasons about how potential cases of inadequate control within a system can lead to a hazardous state and identifies the causal factors that could lead to such a state. The results of STPA can then be used to design protocols or other safeguards within a system to reduce the risk of entering a hazardous state. Finally, Young and Leveson [23] extended STPA into STPA for Security (STPA-Sec) to address the "growing problem of securing software intensive systems against intentional disruptions." STPA-Sec aims to refocus the cyber-security problem away from responding to threats towards controlling vulnerabilities. This refocusing to a top-down approach enables analysis to maintain a system-level perspective while also having the ability to focus in on loss scenarios in high levels of detail. While STPA-Sec does not inform designers what counter-measures should be implemented, it acts as a focusing tool for security efforts for specific systems.

STAMP and STPA-Sec are gaining traction in literature. Beach et al describe an approach to developing quantifiable measures of resiliency using STAMP and STPA-Sec as a framework [24]. Furthermore, Friedberg et al extend STPA-Sec into a methodology called STPA-SafeSec, which aims to ensure system safety as well as security [25].

2.4 Cyber resiliency and System Aware Cybersecurity

The concept of cyber resiliency is relatively new. Traditionally, cybersecurity was thought of in terms of where perimeter defenses should be erected to keep attackers out. Cyber

resiliency, however, refers to the ability of a system to provide and maintain an acceptable level of performance in the face of faults, disruptions, or adversarial actions [27].

Jones and Horowitz [26] developed System Aware Cybersecurity as an approach to achieving resiliency through the implementation of reusable design patterns. The initial design patterns described in [26] include data continuity checking, configuration hopping, and honeypots. Experience and applications to new systems expanded the number of resilient design patterns available in literature. Goldman, McQuaid, and Picciotto describe two overarching categories of resilience strategies: proactive techniques and reactive techniques [27]. Proactive techniques include: segmentation, isolation, and containment, diversity and randomness, moving target and distributedness, non-persistence, and data and system integrity and availability. Reactive techniques include dynamic reconfiguration, deception, dynamic reconstitution, dynamic composition, and alternative operations.

Jones and Horowitz describe an updated selection of resilient design patterns in [28]. These patterns include diverse redundancy, physical configuration hopping, data consistency checking, and verifiable voting. Diverse redundancy involves the introduction of components from different suppliers and can perform the same task within the system. Physical configuration hopping involves the regular switching of system configurations. When combined with diversely redundant components, physical configuration hopping intermittently changes the component controlling some process, thus increasing the difficulty for an attacker to coordinate his or her attacks with a compatible system configuration. Data consistency checking involves the comparison of data at different points in the system to ensure there is agreement between them. In the event of disagreement, an operator can be alerted and the dis-agreeing data reverted to a correct value. Building on data consistency checking, verifiable voting involves the confirmation

of data via comparison between different monitors. If there is disagreement, then it is possible that a cyber-attack has occurred and the data from the compromised monitor is ignored.

2.5 Cyber Security Requirements Methodology (CSRM)

This thesis builds upon the techniques and methods described in the CSRM, a joint research project between the University of Virginia, the Software Engineering Institute (SEI), the Virginia Commonwealth University (VCU) and the US Army's Armament Research Development and Engineering Center (ARDEC). The CSRM is a methodology to develop cyber security requirements during the preliminary design phase for physical systems [29]. The methodology addresses the integration of both defense and resilience solutions and security-related software engineering solutions. The CSRM consists of six steps:

1. High-level development of functional and architectural system descriptions by a systems engineering (SE) team using tools such as SysML
2. Blue team consequence elicitation and analysis, whose deliverable is a prioritized list of undesirable functional outcomes
3. SE team derivation of potential resilience solutions based on the results of step 2
4. Red team prioritization of defense, resilience, and software engineering solutions
5. SE team refactoring of system descriptions based on Red team recommendations
6. Blue team response to the refactored system descriptions.

The CSRM defined a hypothetical weapon system, known as Silverfish, for demonstrating its application. This weapon system consisted of a rapidly deployable set of approximately 50 ground-based munition systems, termed obstacles. These obstacles deny a geographic area from unauthorized trespassers through the use of force, if necessary, to support the protection of a strategically sensitive location. An operator remotely monitors this denied area using a variety of

sensors and visual surveillance. The operator controls the arming, disarming, and firing of the obstacles remotely via a wireless communication network.

The final recommendations of the CSRM regarding resiliency, in order of priority, involved adding diverse communication systems, adding resilient design patterns to the situational awareness components of the system, and adding resilient design patterns to the system's weapon control components. These results are used for comparison with the recommendations of the methodology described in this thesis in Section 4.5.

3. Methodological Approach

This section describes the methodology used to identify appropriate resiliency solutions based on systems-theoretic control and behavior models. The methodology expands on the concepts defined in the CSRM that lead to the identification of potential resiliency-enhancing strategies for a given system. The CSRM identifies potential resiliency solutions based on the mission and system descriptions, inputs from stakeholders, and the judgment of the Systems Engineering team. The methodology introduced in this section can be used to augment the CSRM by providing model-based justification for the Systems Engineering (SE) team, or the methodology can be used on its own to identify and evaluate appropriate resiliency enhancements.

The methodology described in this section, the Systems-Theoretic Resiliency Assessment Tool (STRAT) is composed of four main steps: mission and system specification, systems theoretic consequence analysis, model-based solution identification, and solution evaluation.

3.1 Development of Mission and System Specifications

The STRAT method shares its initial steps with the CSRM. Both the system and the mission it performs are specified at a high-level along with a, preferably rank-ordered, set of

unacceptable outcomes to that mission. These pieces of information form the basis of the STAMP-based analysis from which the system's control structure and potential loss scenarios are derived.

Ideally, the mission and system descriptions are generated by consensus in an iterative process between the SE team and the system owners. However, if the system owners are not available for engagement or if the SE team represent the system owners, then the SE team can complete the descriptions independently. At a minimum, the initial mission description should describe in natural language:

1. The overall mission objective and any sub-objectives,
2. The greater purpose the mission supports,
3. Criteria for mission success and failure,
4. and any constraints on the environment in which the system operates to complete the mission.

The system description shall also describe in natural language how the system is intended to complete the mission, any known components within the system, a basic functional description of the system's operation, and any other known constraints on the system's operation. Preliminary mission and system descriptions should be developed by the SE team and the system owners over two or three iterations and the length of the descriptions should not exceed one to two typed pages. Agreeing upon a concise description has the dual benefit of scoping analysis to a more manageable degree for complex systems as well as preventing confusion about the goals of the mission and how the system is used to help reach those goals.

Following the development of the mission and system descriptions, if the system owners are available for engagement, the STRAT method borrows from Step 2 of the CSRM- the Blue

Team consequence elicitation meeting. The Blue Team meeting engages the SE team with the system owners to elicit a prioritized set of undesirable consequences or outcomes with respect to the use of the system in the mission. The development of the list of undesirable outcomes is based on the agreed upon mission and system specifications described previously. The SE team is responsible for facilitating the discussion and documenting the outcomes along with other relevant pieces of information from the system owners. Such information could include, but is not limited to, the components that would likely need to be attacked to produce that outcome and the potential method of attack. The CSRM, for example, identifies an additional piece of information- STAMP type- to further characterize the undesirable outcome in terms of the control action (or lack thereof) needed to produce the outcome. All of this information collected from the systems owners forms the foundation for the STAMP-based analysis and construction of the system's control model.

In the event that conducting a Blue Team meeting as described in the CSRM is not possible, then the SE team will need to rely on their understanding of the system and mission description and personal expertise. Under these circumstances, the value of having a clear and consistent system and mission description becomes evident. If the descriptions are easily understood, then it becomes more likely that a non-user or non-expert will be able to identify valid, and well-formed, undesirable outcomes. Regardless of the team that develops the list of undesirable outcomes, the rationale behind each outcome should be documented to enable any cascading effects in future analysis to be traceable.

3.2 Systems-Theoretic Consequence Analysis

Following the specification of the mission, system, and undesirable outcomes, the SE team performs a systems-theoretic consequence analysis to define the system's functional control

structure, behavior, and potential scenarios that might produce undesirable outcomes. More specifically, this step of the methodology is based on Leveson's STAMP model and STPA/STPA-Sec analysis tools. The STRAT follows the concepts of the STAMP model and performs most of the steps in the STPA-Sec analysis tool, but the goals of each method differ. STPA-Sec identifies scenarios, that could be the result of a cyber-attack, to focus cybersecurity efforts; however, STRAT uses the STAMP-based analysis to guide the construction of models that are used to identify appropriate locations and types of cyber-resilience strategies [23].

STPA and STPA-Sec begin with the identification of unacceptable losses in the mission at hand. STRAT uses the information collection methods described in section 3.1 to perform this same task. The set of undesirable outcomes generated by the SE team or the system owners are directly mapped into the unacceptable losses used in the consequence analysis. Unacceptable losses in STPA-Sec syntax are high-level events that typically imply total mission failure. Consequently, it may be possible that some of the undesirable outcomes generated in the previous step may be too specifically defined to be well-formed unacceptable losses. In such cases, there is likely an implicit higher-level loss event tied to that outcome that should be defined. For example, multiple undesirable outcomes may be able to be categorized as a more general type of unacceptable loss. It should be noted, however, that the one of the purposes of beginning with the definition of unacceptable losses is to scope later analysis, and therefore, the set of unacceptable losses should not be so specific that the problem space becomes too complex.

After the definition of unacceptable losses in the mission, a set of hazardous conditions that could contribute to one of the unacceptable losses are identified. In fact, some of the more specific undesirable outcomes from the Blue Team elicitation are likely to describe a hazardous scenario that could lead to a higher-level loss event. Hazardous conditions outline scenarios that

could occur during the operation of the system within the mission that would lead to an unacceptable loss if they were to occur in combination with the presence of a worst-case environment. Young and Leveson illustrate this by describing a nuclear power plant that has an unacceptable loss of not producing power to the grid. A hazardous scenario for the power plant would be the shutdown of the reactor, however, the associated unacceptable loss only occurs if there are no auxiliary generators or if the reactor is shutdown longer than the endurance of the auxiliary generators [23].

Following the identification of hazardous scenarios, the basic control structure of the system is defined. The development of the control structure is based on the controller, actuator, controlled process, and sensor feedback loop seen in Figure 2.

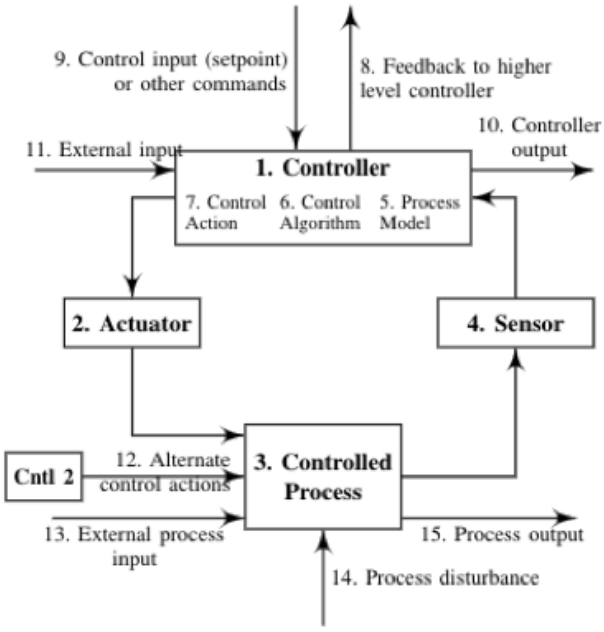

Figure 2. The generic control loop structure that is used to formulate the control model. (From [30]).

The system's control structure emerges as these loops are stacked on top of one another, in parallel, or merged together, similar to the control structure of a fictional missile defense system shown in Figure 3. The combination of these loops creates a hierarchy of controllers and

controlled processes that begins to describe the technological and organizational mechanisms that the system uses to operate within its mission domain. More specifically, the hierarchical control structure defines how commands and control actions propagate from the higher-level controllers to lower-level controllers or controlled processes and how those lower-level entities provide feedback to their higher-level controllers [30]. Identifying how the system accomplishes these tasks is the first step to understanding how unintended or uncontrolled system behavior can lead to unacceptable losses.

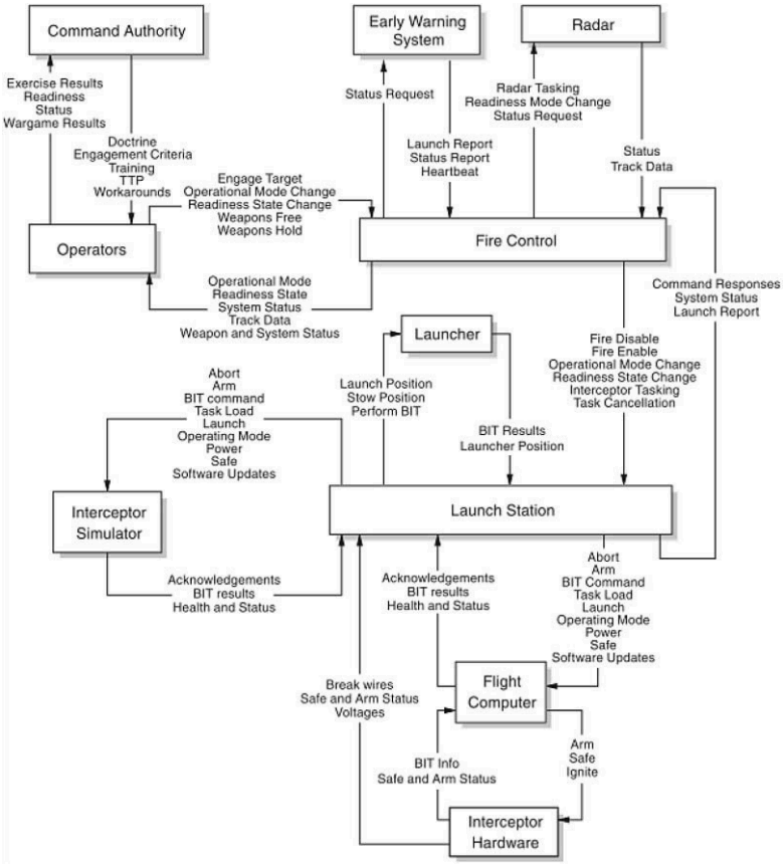

Figure 3. A "stacked" control structure of a fictional missile defense system (From [24]).

Defining the control structure allows for the enumeration of the control actions available to each controller within the hierarchy. Since the STAMP causality model asserts that hazardous conditions are the result of performing control actions improperly, the enumeration of control

actions allows the SE team to identify the scenarios under which improperly implemented control actions lead to hazardous conditions, and thusly, potential unacceptable losses. Improper control actions can be categorized into four types of implementation:

1. Providing the control action leads to a hazardous condition
2. Not providing the control action leads to a hazard
3. Providing the control action too early, too late, or in the incorrect order leads to a hazard
4. Stopping a control action too soon or performing a control action too long leads to a hazard.

By creating a table of the possible control actions and how each type of improper implementation of those control actions can lead to hazardous conditions, the SE team begins to identify potential areas of concern within the control structure through the process of elimination. Some control actions will not have scenarios that create hazardous conditions for all of the improper implementation types because of the nature of the control action. Thus, those cases can be ignored in future analysis, thereby reducing the problem space. Furthermore, the SE team will be able to take note of any control actions that can lead to the same hazardous condition for multiple improper implementation types. These control actions can be flagged as areas to investigate more thoroughly in later analysis.

The final step in STPA and STPA-Sec involves the construction of causal scenarios that describe why an improper control action was taken. The identification of these scenarios facilitates an understanding of the impact cyber events have on the mission- something with which traditional security methodologies may struggle. Furthermore, identifying the potential mechanisms through which adverse outcomes can occur helps motivate the choice of appropriate resilient design patterns later on.

The main artifacts of the STAMP-based consequence analysis are the definition of the system's functional control structure and the documentation of the relevant system losses, hazards, hazardous control actions, and causal scenarios. These artifacts aid the SE team in understanding the manner in which vulnerabilities can propagate through the system in addition to forming the foundation for the construction of the system and behavior models.

3.3 Model-based Solution Identification

3.3.1 The System Model

While the STAMP-based consequence analysis facilitates understanding of the system's control structure and identifies potential pathways for vulnerabilities that lead to adverse outcomes, it does not produce an analyzable model. Consequently, it becomes advantageous to represent the system's control structure, unacceptable outcomes, and other STAMP-based analysis information in graphical form. This representation allows for the visualization of the control actions, the resultant changes to the system, and the emergence of mission-level consequences from those actions. Furthermore, the graphical formulation allows for the beginnings of a quantification of the qualitative subject matter obtained in the mission and system specifications and the consequence analysis.

The graphical representation of the system, its control structure, and the consequence analysis necessitates a special definition of its graphical objects. This graph, known as the specification graph, or S-graph, shares similarities to the definition of a *multidigraph* or *quiver* [31]. However, the S-graph's vertices and edges are supersets of dissimilar sets of vertices and edges. The need for differing types of vertices and edges arises from the representation of the elements in the control loop shown in Figure 2. The S-graph models actors in the system by having its vertices represent an entity from the generic control loop, a combination of those

entities, a physical state, or a function that represents an outcome. It follows that the edges in the graph represent the actions performed by or resulting from the actors.

Following these concepts, the S-graph is composed of a combination of six types of vertices: outcome vertices, state vertices, actuator vertices, sensor vertices, controller vertices, and *meta* vertices. The outcome vertices describe the presence or absence of certain conditions from the consequence analysis, such as the presence of a hazardous condition or the occurrence of an unacceptable loss. State vertices are broadly defined as the set of variables or controlled processes that are not also controlling a lower-level process, such as vehicle's location, speed, etc. As the name implies, actuator vertices represent an actuator in the system's control structure that receives input from a controller and acts upon a controlled process. Likewise, sensor vertices represent a sensor in the system's control structure that monitors a controlled process or state and sends feedback to a controller. The S-graph's controller vertices represent a controller in the system's hierarchical control structure. Due to the control hierarchy, a controller vertex will both receive inputs from a higher-level actuator and send control actions to a lower-level actuator, and vice-versa for its corresponding sensor vertices, unless the vertex is the highest-level controller in the system. Finally, *meta* vertices can be used to represent a combination of controllers, actuators, sensors, or controlled processes. This allows for the possibility that an entity in the hierarchical control structure shares the responsibilities of two or more of the parts of the generic control loop. An example of such shared responsibilities is illustrated in Figure 3.

The edges of the S-graph are also of different types. These types include action edges, feedback edges, and conditional edges. Action edges represent the control actions or dynamics through which a higher-level vertex influences a lower-level vertex. For example, a controller vertex may have multiple action edges from itself to the subsequent actuator vertex that represent

the control actions available to that controller in the system's control structure. Feedback edges represent data or information that is propagated from a lower-level vertex to a higher-level vertex, such as the feedback from a sensor to its higher-level controller. Finally, the conditional edges represent the inputs to an outcome vertex.

Using these definitions to the varying types of vertices and edges, a mathematical definition of the S-graph is as follows. The specification graph, S, is a 4-tuple similar to a multidigraph, or quiver:

$$S \coloneqq (V, E, p, t)$$

where V is the superset of nodes in the graph, E is the superset of edges, $p : E \to V$ assigns each edge to its parent vertex, and $t : E \to V$ assigns each edge's target vertex. Furthermore, the superset V is defined:

$$V \supseteq O, X, A, D, C, M$$

where O is the set of outcome vertices, X is the set of state vertices, A is the set of actuator vertices, D is the set of sensor vertices, C is the set of controller vertices, and M is the set of meta vertices. Likewise, the superset E is defined:

$$E \supseteq B, Y, Z$$

where B is the set of action edges, Y is the set of feedback edges, and Z is the set of conditional edges.

The structure of the S-graph shares striking similarities to a quiver- such as the potential to have multiple edges between two nodes, each with its own identity. However, the combination of disparate sets of vertices and edges, each representing a system component or behavior that may governed by incompatible mathematics, presents significant challenges to performing mathematical operations on the S-graph. Consequently, the S-graph is currently used to simply

represent a visualization of the system and help formulate the behavior model in Simulink. The mathematical definition of the S-graph, however, does provide a starting point for future efforts intending to automate analysis of the system model.

3.3.2 The Simulink Behavior Model and Simulation

Due to the S-graph's potential limitations with respect to automated analysis techniques, it becomes necessary to use simulation for identifying appropriate resiliency strategies. Simulink provides the necessary tools for simulating the structure and behavior described in the S-graph without the difficulties associated with the mathematics of the graphical representation. Simulink's and Stateflow's combinatorial and sequential decision logic tools and other model elements enables the abstraction of some of the S-graph's complexity into a more easily executable form. More specifically, through a series of source blocks, mathematical operators, state machine diagrams, and flowcharts, this step of the methodology constructs a simulation of the system's intended behavior. The simulation of normal behavior is then used to determine where and how adverse behavior can be introduced, thus leading to the identification of appropriate resiliency strategies and their location within the system.

As previously mentioned, one of the difficulties associated with the definition of the S-graph is the diversity of what the vertices and edges represent. The Simulink model allows for these different types of vertices and edges to take on actual implementations of what they represent. For example, one particular controller in the system may follow a decision model that is describable in a truth table, whereas another controller operates based on a set of differential equations. Simulink enables both to be encoded to the desired level of granularity in the behavior model.

It should be noted that each behavior model and simulation is heavily dependent on the system in question and its associated mission. However, by using the S-graph as a starting point, each of the types of vertices and edges generally map to similar model elements within Simulink regardless of the system being modeled. Table I presents the types of S-graph elements mapped to a Simulink model element that should sufficiently describe its behavior for most applications.

Table I. A mapping of S-graph elements to a Simulink model element that should sufficiently represent its behavior in most applications.

Type of S-graph Vertex	**Corresponding Simulink Model Element**
Outcome Vertex	Truth table
State Vertex	Source block and/or state machine diagram
Actuator Vertex	State machine diagram
Sensor Vertex	State machine diagram and/or math operator blocks
Controller Vertex	State machine diagram or truth table
"Meta" Vertex	State machine diagram
Type of S-Graph Edge	
Action Edge	Embedded in truth table or state machine diagram
Feedback Edge	Inputs/outputs to and from sensor vertex model element
Conditional Edge	Inputs/outputs to outcome vertex truth tables

By using Simulink source blocks and state machine diagrams to represent the state vertices, the simulation takes on a scenario-based format. This allows the SE team to generate conditions that produce the intended behavior of the system within the mission. Once the system's normal, or intended, behavior is represented by the simulation, then the SE team can explore ways to generate unintended or undesirable system behavior.

The SE team can take two approaches to producing undesirable behavior: by creating starting conditions based on the hazards defined in the consequence analysis, or by finding ways to generate the causal scenarios outlined in the consequence analysis. For each approach, the SE team documents the nature of any undesirable behavior that is generated, how it was generated, a list of potential mitigation strategies for that behavior, and the locations within the system for

implementing those strategies. The potential mitigation strategies are based on the resilient design patterns described by Jones and Horowitz for System-Aware cybersecurity [28].

The first approach defines a set of starting conditions that are either immediately hazardous, or likely to become hazardous. As an example, imagine that the model describes an autonomous vehicle. In this case, some of the state vertices would describe any obstacles in the vehicles path along with the vehicle's current speed and heading. Following the scenario-based construction of the model, these variables would be programmable to be immediately hazardous at the start of the simulation, i.e. an obstacle directly in the path of the vehicle's current heading and within the vehicle's safe-maneuvering perimeter. In this scenario, the system's behavior should account for this hazardous condition and attempt to mitigate the danger. If the system is unable to adequately handle such inputs, then the SE knows to investigate the introduction of some safeguards or resiliency measures to mitigate such situations. The selection of potential resiliency strategies and their location depends on the nature of the hazardous condition and varies from system to system.

The second approach to generate undesirable behavior in the simulation aims to generate hazards from within the model, rather than starting with hazardous conditions. More specifically, the initial starting conditions are such that the system should be expected to behave in its intended manner, however, the SE team changes parameters, noise levels, or other model elements with the intent of producing hazardous or unacceptable outcomes. Following the autonomous vehicle example described above, the intent would be to produce unsafe behavior from "normal" conditions. One potential method for doing so could involve introducing additional noise or bias into the system's obstacle detection sensors. In a non-resilient system, this could easily result in the failure to detect and avoid an obstacle, thus creating a hazardous

scenario and a potential unacceptable loss. Where these changes intended to produce unintended behavior occur within the system and what is being changed define the possible resilient design patterns that would be appropriate. For instance, in the above example, if increased noise in the obstacle detection sensors leads to undesirable behavior, then an appropriate resiliency strategy would be to include a noise monitoring algorithm and redundant backup sensors.

Both of these approaches should be used to describe all of the hazardous conditions and causal scenarios from the consequence analysis as appropriate- it is possible that not all the items from the consequence analysis are applicable to both approaches. Once all of the consequence analysis items are exhausted, the SE team should have a list of potential resilience strategies and the locations within the system for their implementation. At this point, the SE team should use their discretion to remove any strategies that address scenarios that might be unrealistic or are otherwise infeasible. Furthermore, it is possible that the list may contain some duplicate strategies; these items in the list should be merged and the number of duplicate entries recorded as this can be used as a measure of priority in strategy evaluation.

3.4 Evaluating Resiliency Solutions

The choice of which resiliency solutions to implement is a multi-criteria decision problem primarily involving the cost of the solution, the impact of the solution on the adverse outcome(s) to be mitigated, and the likelihood of the adverse outcome(s) occurring. How each of these factors, among the many others not mentioned, is dependent on the preferences and worldviews of the decision-makers. Furthermore, the cost of a resiliency solution, which includes the monetary value, the complexity of design, and the ease of integration into the system, varies greatly depending on the application. Thus, analysis of the cost factor is outside the scope of this thesis. However, the simulation and STAMP-based consequence analysis enable

an evaluation of the adverse outcomes to be addressed by the solution as well as the solution's impact on those adverse outcomes. By taking advantage of the similarity of these two factors to the traditional definition of risk, the set of resiliency solutions can be prioritized into "risk" categories. These categorized solutions form a cost-agnostic recommendation of which resiliency measures to pursue.

For every entry in the list of solutions identified based on analysis of the simulation, there is an associated list of adverse outcomes addressed by a particular solution. These adverse outcomes and the impact of those solutions form the basis of the "risk" measure based on the traditional definition [32]:

$$risk = impact \times likelihood.$$

For the purposes of this application, impact is a measure of the number of adverse outcomes that a solution intends to address, the priority of those outcomes in the consequence analysis, and the effect of adding that solution on the operation of the system. This solution's effect on the system can be determined by adding in a representation of the solution to the simulation and comparing the results to the unaltered system if the nature of the solution allows. Otherwise, the effect must be judged qualitatively. Likelihood is a measure of the ease of achieving adverse outcomes in the simulation. More specifically, the number of changes to the simulation needed to achieve an adverse outcome and the severity of those changes. It should be noted that this definition of likelihood does not incorporate a probabilistic assessment of the ability of potential adversaries to create those changes in the system as such is out of the scope of this thesis. However, methods for creating such an assessment could easily augment the methods described here.

Given the nature of the factors that make up the impact and likelihood measures, quantitative metrics defining each dimension of the "risk" score are difficult, if not impossible to

identify. Therefore, impact and likelihood are categorized into rankings of low, medium, and high. Thus, the risk matrix framework can be readily applied to this application and resiliency solutions are categorized into low, medium, and high priorities for implementation [32].

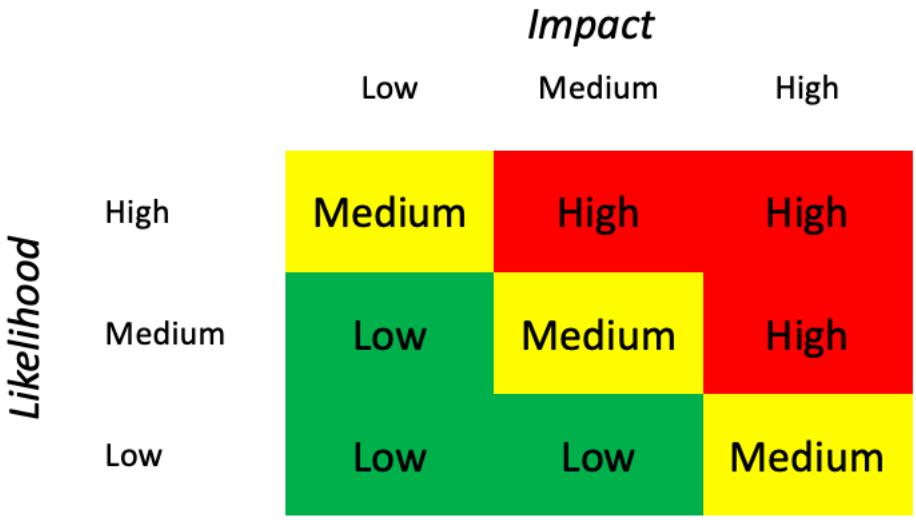

Figure 4. The risk matrix prioritization framework for resiliency solutions.

After generating the set of recommended resiliency solutions in the risk matrix framework, all or a subset of the resiliency solutions can be applied to the system and the analysis iterated on the "new" updated system. A strength of the methodology presented in this section is the ability to refactor in resiliency solutions at multiple different steps. Solutions could be refactored into the initial system and mission descriptions or simply incorporated into the simulation model. Either approach offers greater confidence that all appropriate resiliency solutions are considered for a particular system.

4. Demonstration of Method

This section details an application of the approach described in Section 3 on the hypothetical US Army weapon system analyzed for the CSRM, known as Silverfish. Results are then compared with the recommendations of the CSRM to assess the compatibility of the

methodology with existing techniques. The methodology presented in this section uses the same mission and system descriptions and Blue Team-defined unacceptable consequences as the CSRM to allow comparison of recommendations. As stated in Section 3.1, the methodology used in this paper does not require that this information be collected in the same way as the CSRM, however, it acknowledges that engaging the system owners increases the veracity of the collected information.

4.1 Mission and System Specification

The Silverfish system was initially developed to be a testbed for the application of the CSRM. Although it is a hypothetical system, the US Army Armament Research, Development, and Engineering Center (ARDEC) determined that the system is both representative of a system that could potentially be used by the Army and is suitable for the demonstration of cybersecurity techniques. The assembled Blue Team, composed of members of the ARDEC, and the SE team developed the initial Silverfish mission and system descriptions through a series of iterations before agreeing upon the final description below [29].

The Silverfish system performs an area denial mission to aid the protection of a strategically sensitive location. More specifically, Silverfish deploys a set of 50 ground-based weapon systems, known as obstacles, that can engage unauthorized persons or vehicles within the denied area. The denied area measures up to approximately .16 square miles in size, with each obstacle capable of protecting a 300 foot by 300 foot area. A set of surveillance sensors including static infrared and video cameras and target characterization sensors, such as acoustic and seismic sensors, provide situational awareness by monitoring the area for persons and vehicles. An unmanned aerial vehicle also provides surveillance and early warning information by monitoring the periphery of the denied area. The Silverfish operator controls the obstacles and

situational awareness sensors remotely from a nearby vehicle that can be maneuvered to give the operator "eyes-on" monitoring over portions of the denied area.

The operator has control over the obstacles' armed or disarmed states and fire capabilities. He or she uses the situational awareness information available to determine target identity and the appropriate obstacle with which to engage the target. A wireless network relays the operator's commands from the control station to the obstacles. Furthermore, the operator has the ability to communicate with a command and control center to receive orders and additional situational awareness information.

For the purposes of this thesis, the analysis and recommendations are limited to the components that are "owned" by the Silverfish system. This means that the command and control center, the UAV, and the vehicle have their capabilities and inputs to the system considered when identifying resiliency strategies, but changes to these systems are out of scope for analysis.

Following the framework described in Section 3.1, the criteria for mission success are simple: all unauthorized persons or vehicles in the denied area are engaged correctly for the duration of the mission. Mission failures result from unauthorized persons or vehicles successfully traversing the denied area or friendly fire incidents.

Following the finalization of the mission and system descriptions, the Blue Team and SE team met to develop a prioritized list of unacceptable consequences with respect to the Silverfish mission. The CSRM supplemented this meeting with SysML representations of the agreed upon mission and system descriptions. Each entry in the list of consequences received a priority based on the following Likert Scale:

1. Unacceptable and highest priority to provide resiliency

2. Avoid as long as resiliency solution does not over-complicate operation
3. Would like to avoid, but solution needs to be incremental
4. Lowest priority, low-cost, simplistic solutions should be considered.

Within each prioritization level, the consequences were further ranked based on the Blue Team's perception of their severity. For each consequence, the potential targets of an attack that would produce that outcome were identified, along with the potential method for completing the attack. Finally, the types of inappropriate control actions that would be associated with that consequence were identified using the following categorization:

1. Providing a control action causes a hazard
2. Not providing a control action causes a hazard
3. Incorrect timing or improper order of control actions causes a hazard
4. A control action is applied too long or stopped too soon.

The output of this meeting is presented below in Table II.

Table II. The list of unacceptable consequences identified by the Blue Team.

Likert Rank	Consequence	Attack Target(s)	Attack Method	Control Action Type
1.1	Inappropriate firings via manipulating operator commands	Operator control display, radio comm links	External, supply chain, insider	1, 2, 3
1.2	Delays in fire time (sufficient delay to cross field)	Obstacles, control station, radio comm links	External, supply chain, insider	2, 3
1.3	Delays in deployment	Obstacles, deployment support equipment	Supply chain, insider	2, 3
1.4	Deactivation of a set of obstacles	Obstacles	External, insider	1, 3
2.1	Delays in situational awareness	Operator display, sensors	External, insider, supply chain	1, 2, 3
2.2	Prevent or corrupt transmission of situational awareness data	Radio comm links, operator display, sensors	External, insider, supply chain	1, 2, 3
2.3	Gain information to help adversary navigate through field	Obstacle, operator control station	External, insider	2, 3
3.1	Reduced operational lifespan	Obstacle	External, supply chain, insider	1, 2, 3,
3.2	Prevent transmission/execution of non-firing commands	Operator display, obstacles	External, insider, supply chain	1, 2
4.1	Delays in sending/receiving C2 information	Operator display, radio comm links	External, supply chain	1, 2, 3
4.2	Delays in un-deployment	Obstacles	External, insider, supply chain	1, 2, 3

The list of unacceptable consequences, along with the Silverfish mission and system description, form the ground truth from which all further analysis is based. Following the completion of this step, the STRAT does not require further involvement of the system owners (Blue Team) or other non-SE team members, unlike the CSRM.

4.2 Systems-Theoretic Consequence Analysis

As described in Section 3.2, following the definition of the mission and system descriptions and the identification of undesirable consequences, the STRAT uses STAMP and STPA-Sec concepts to document unacceptable losses, hazards, and the system's control structure.

The mission and system descriptions defined the basic conditions for mission success and failure. Silverfish achieves mission success if no unauthorized agents traverse the denied area for the duration of the mission; mission failure occurs when unauthorized agents successfully traverse the denied area or obstacles are fired upon friendly forces. These two definitions translate into the following unacceptable losses or outcomes for this mission:

- L1- Enemy forces or other unauthorized persons/vehicles traverse the denied area without the operator's knowledge or intent,
- L2- Friendly forces, civilians, or other non-combatants are killed or harmed by Silverfish,
- L3- Silverfish obstacles are fired without a valid target.

Unacceptable losses L1 and L2 clearly map to the stated mission of Silverfish; however, L3 was derived as an additional, lower priority unacceptable loss because of the implications it has on the outcome of the mission. As seen in the outcomes described in Table II, the Blue Team is concerned about losing control of Silverfish or Silverfish not being able to operate as intended for the mission's duration- L3 describes a third end-result of such consequences that does not involve friendly fire or the immediate traversal of unauthorized agents through the denied area.

Following the definition of the unacceptable losses, the STRAT defines the hazardous conditions that could lead to an unacceptable loss. These hazards define conditions that do not

immediately result in an unacceptable loss, but will lead to an unacceptable loss given an improper implementation of a control action or the presence of a worst-case environment. Table III defines a set of hazardous conditions, the worst case environment for those occur in, and the unacceptable losses associated with that hazard.

Table III. Hazardous Conditions that could lead to an unacceptable loss.

Hazard	Worst Case Environment	Associated Losses
H1- Failure to fire correct obstacle	Imminent threat entering denied area	**L1**
H2- Incorrect obstacle armed or fired	Friendly in denied area	**L1, L2, L3**
H3- Wireless link to obstacles down	Imminent threat in denied area	**L1**
H4- Situational Awareness data inaccurate, delayed, or unavailable	Imminent threat entering denied area; friendly agent in denied area	**L1, L2**

Next, the basic control structure of the Silverfish system is defined. Using the control loop format described in Section 3.2, Silverfish is decomposed into its main controllers, sensors, actuators, and controlled processes. Based on the system description, Silverfish consists of an operator who controls the obstacles and visual sensors through a control station over a wireless network. This involves the operator overseeing three controlled processes: fire control, surveillance, and target characterization. The operator manages all three processes through the control station. The obstacles actuate fire control commands, the visual sensors actuate surveillance and target characterization, and the characterization sensors (acoustic and seismic) also enable target characterization. The sensors provide feedback to the operator on the three controlled processes via the control station. This basic structure is presented in Figure 5. It should be noted that the simplicity of this particular system is not necessarily shared by other systems.

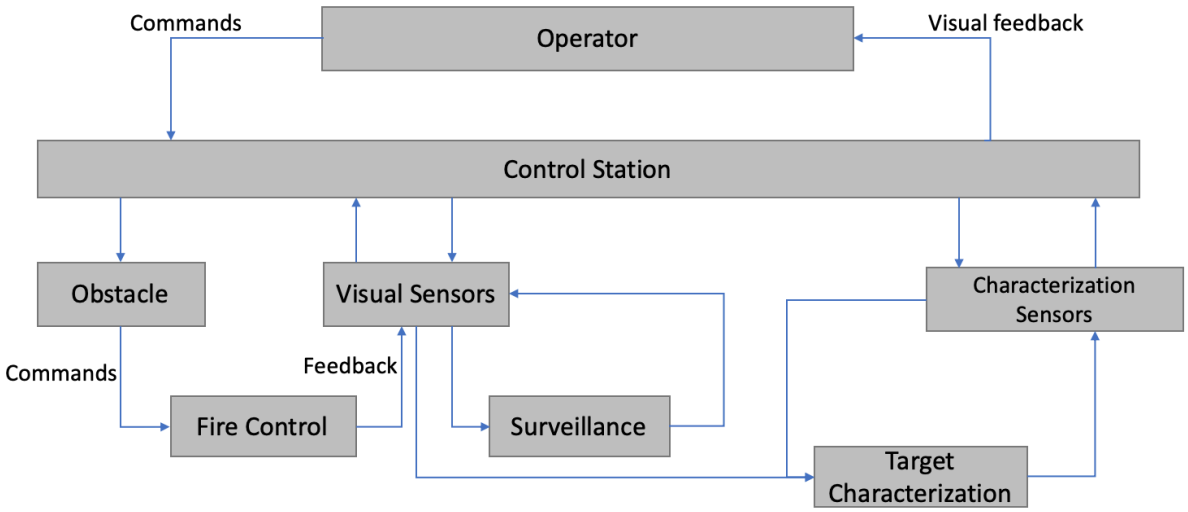

Figure 5. The basic control structure of Silverfish.

From this control structure, the control actions available at each hierarchical level are enumerated, and the conditions under which each control action contributes to a hazard identified. In the representation of the system described in Figure 5, the control actions available to operator are effectively identical to those available to the corresponding lower levels of the system. Consequently, those control actions for the other hierarchical levels of the system are omitted from the control actions in Table IV as they would be redundant. Again, this characteristic is a result of the simple nature of the Silverfish system, and not indicative of other applications. Understandably, however, if the lower level controllers do not enact the operator's control actions accurately, then hazards are likely to occur.

Table IV. Control actions and the conditions under which they would contribute to a hazard.

Control Action	Not Providing causes hazard	Providing Causes hazard	Incorrect Timing or Order	Stopped too soon or applied too long
Operator Control Actions				
CA 1.1- Arm/Disarm Obstacle	Target in denied area- H1, H3	Friendly in denied area- H2	Target not in range- H1	Target not in range- H1
CA 1.2- Fire Obstacle	Target in range- H1, H3	Friendly in range- H2	Target not in range- H1	Target not in range- H1
CA 1.3- Adjust visual sensor field of view	Target not identified- H1, H3, H4	Target goes unidentified- H1, H4	Target unidentified- H1, H4	Target goes unidentified- H1, H4
Obstacle/Sensor Control Actions				
CA 2.1- Send feedback	Operator doesn't receive data- H1, H3, H4	Data is corrupted- H1, H2, H4	N/a	Target goes unengaged- H1

The final step of the consequence analysis involves the generation of causal scenarios that describe the implementation of improper control actions. The undesirable outcomes defined by the Blue Team motivate the definition of causal scenarios associated with each control action. These causal scenarios help inform the choice of appropriate resiliency measures in the next step by providing details on what might cause a control action to be implemented improperly. Table V presents control actions mapped with an associated causal scenario and the priority rank of the related undesirable outcome(s) from the Blue Team.

Table V. Causal scenarios for implementing an improper control action mapped to undesirable Blue Team outcomes.

Control Action	Causal Scenario	Blue Team Outcome
CA 1.1	Legitimate operator control action overridden or altered due to cyber-attack on control station or network	1.1, 1.2, 1.4,
CA 1.2	Legitimate operator control action given, but improper due to misclassification of target	2.1, 2.2, 3.2, 4.1
CA 1.3	Legitimate control action overridden or altered due to cyber-attack	2.1, 2.2, 2.3, 3.2
CA 2.1	Cyber-attack causes delay, denial, or increased rate of control action application	2.1, 3.1, 3.2

4.3 Model-based Resilience Solution Identification

4.3.1 Model Construction

The next step in the STRAT begins with the development of the graphical system model. This model shares the same basic shape as the hierarchical control structure identified in the consequence analysis, but incorporates additional STAMP-related information. Using the definitions outlined in Section 3.3.1, the S-graph for the Silverfish system is presented in Figure 6. Each vertex and edge is color-coded to the types described previously.

The vertices labelled 1, 2, 3, 5, and 7 map directly to their corresponding blocks in the control structure shown in Figure 5. The vertices labelled 4 and 6 represent the physical states that define the presence or absence of an unacceptable loss. The obstacle state describes whether or not the obstacles are armed and whether or not an obstacle has been fired. Likewise, the denied area state describes any agents within the denied area and their location. The vertex labelled 8 represents the outcome matrix, which describes the presence or absence of an unacceptable loss or hazardous condition. The edges labelled a, b, c, h, and i are the action edges, which describe the control actions or dynamics through which the parent vertex influences the target vertex. The edges labelled d, e, f, g, k, and l represent feedback from the parent vertex to

the target vertex. Finally, the edges m and n represent the conditional edges that are the inputs to the outcome matrix.

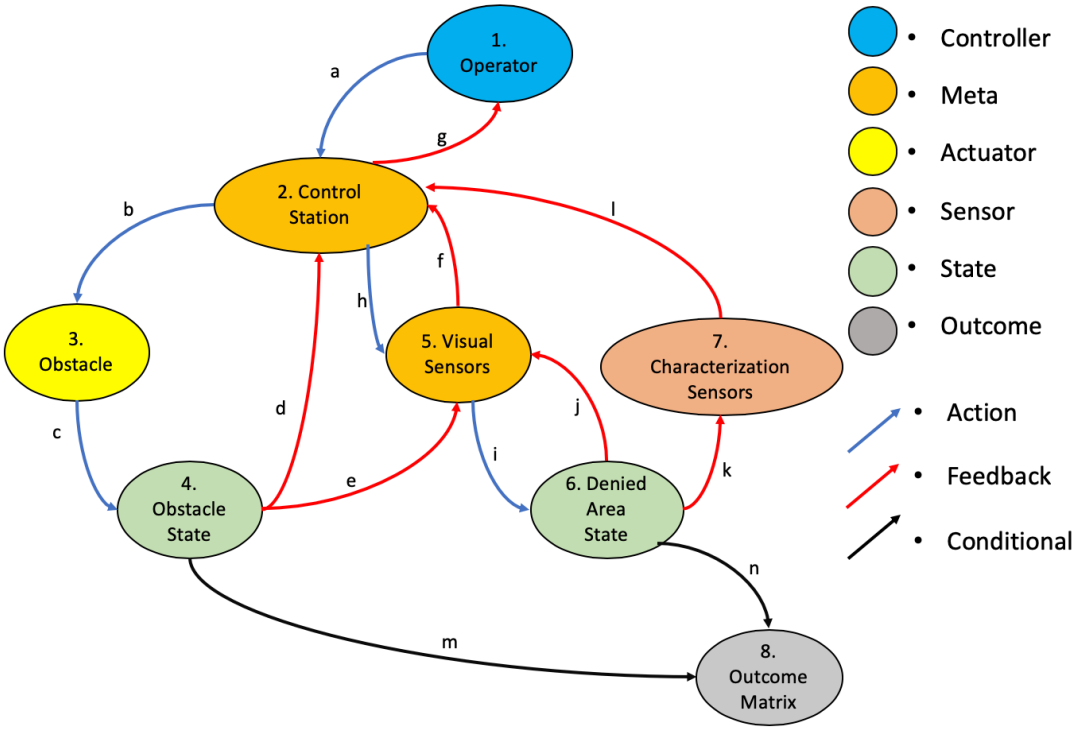

Figure 6. The S-graph for the Silverfish System.

As stated previously, at this point of development, the S-graph mainly serves as the foundation for the Simulink behavior model. Future research on the mathematics of the S-graph formulation could allow for further analysis on the system's control structure.

The Simulink model follows the construction guidelines defined in Table I. For the purposes of this particular system, however, the operator and control station are represented as a single entity. This is because the control station and the operator follow the same decision logic, thus making separate model representations superfluous. A screenshot of the Simulink model is shown in Figure 7.

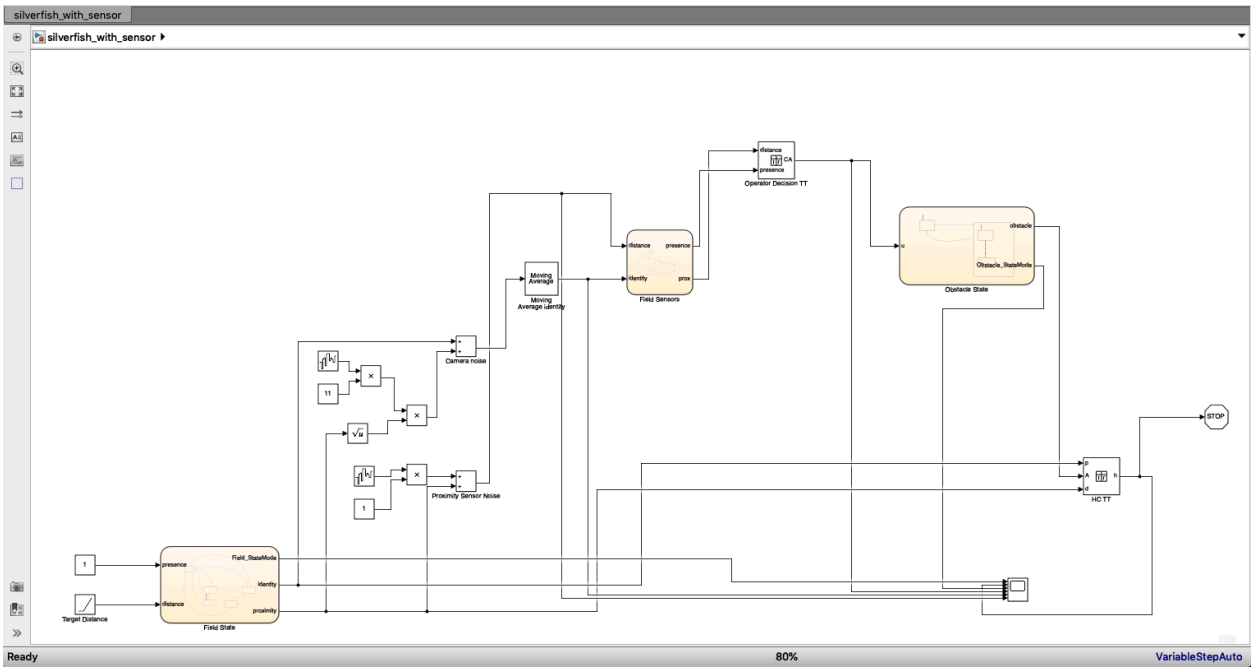

Figure 7. A screenshot of the Simulink behavior model.

The Simulink model follows the scenario-based formulation described in Section 3.3.2. That is, the state variables not controlled by the system, agent identity and proximity to an obstacle are defined as source inputs to the simulation. For the purposes of this application, agent identity is defined as a constant and proximity decreases linearly from an initial with time. These variables are the inputs to a state machine diagram that defines the "true" state of the denied area based on the values of the source variables. This "true" state of the denied area is defined as one of the following states in the state machine diagram (combinations of these states are not considered as the operation of Silverfish in such situations becomes dependent on the operator's specific rules of engagement):

1. No agent present in the denied area
2. A non-enemy agent in the denied area, but out of range of an obstacle
3. An enemy agent in the denied area, but out of range of an obstacle
4. A non-enemy agent in the denied area, and in of range of an obstacle

5. An enemy agent in the denied area, and in of range of an obstacle.

The state of the denied area is monitored by the surveillance and target characterization sensors, which introduces noise into the estimate of the state of the denied area. This estimation is represented by another state machine diagram with the same states defined above, however, the inputs are combined with Simulink noise blocks. The estimate of the state of the denied area forms the input to the operator's decision model regarding which control actions to take. This decision model is encoded in a truth table that maps the estimated denied area states to an appropriate control action. This truth table is presented below in Table VI.

Table VI. A truth table representation of operator decision logic.

Condition	D1	D2	D3
Agent Present in Denied Area	T	T	-
Confirmed Enemy Agent	T	T	-
Agent in Range of Obstacle	T	F	-
Control Action	Fire	Arm	Disarm

The resulting control action of the truth table then forms the input to the state machine diagram that represents the state of the obstacle. Each obstacle can be armed, disarmed, or fired. The state of the obstacle is then combined with the "true" state of the denied area to form the inputs to the outcome matrix. The outcome matrix is also defined as a truth table, mapping S-graph states to consequence analysis losses and hazards, seen below in Table VII.

Table VII. A truth table description of the outcome matrix.

Condition	D1	D2	D3	D4	D5	D6	D7	D8	D9
Agent Present in Denied Area	T	T	T	T	T	T	F	F	-
Confirmed Enemy Agent	T	T	T	T	F	F	F	F	-
Agent in Range of Obstacle	T	T	T	F	T	T	F	F	-
Obstacle Armed	T	T	F	F	T	T	T	T	-
Obstacle Fired	T	F	F	F	T	F	T	F	-
Outcome	n/a	L1	H1	H1	L2	H2	L3	H2	n/a

Running the simulation in its baseline configuration always results in the "n/a" outcome defined in Table VII, which indicates that the system is operating as intended given a particular set of starting conditions.

4.3.2 Identifying Resiliency Solutions from Simulation Changes

As described in Section 3.3.2, changes to the simulation intended to create the hazardous conditions and unacceptable losses from the consequence analysis identify the locations for potential resilience solutions. For this particular system, the first approach to producing adverse outcomes- introducing hazardous starting conditions for the simulation- does not apply. Since the sources variables describe the agent identity and proximity, the only possible hazardous starting condition would be an enemy agent in range of an obstacle. As stated in the previous section, the system's decision logic would immediately resolve the situation.

The second approach to producing adverse outcomes in the simulation, however, yields meaningful results. The causal scenarios identified in the consequence analysis and defined in Table V describe potential ways that improper control actions can lead to adverse outcomes. These causal scenarios can be categorized into three types of root causes:

1. Legitimate control actions made based on erroneous decisions

2. Legitimate control actions overridden by invalid control actions
3. Control actions blocked or delayed in implementation.

Using these three types of root causes as a basis for design, changes are made to the simulation with the intent of producing the hazardous conditions defined by the consequence analysis.

For example, adding bias to the estimate of an enemy agent's proximity to a particular obstacle can result in the failure to fire the correct obstacle- H1. Depending on the geometry of the denied area, as little as a 10% bias to the proximity estimate can result in the firing of an incorrect obstacle (assuming that the obstacles have a 50 meter range and the effective ranges of adjacent obstacles overlap by 5 meters). Following this example, each hazardous condition from the consequence analysis is mapped to a list of changes in the simulation that produce the outcome and their corresponding locations in the system in Table VIII.

Table VIII. A mapping of simulation changes to the hazardous conditions they contribute to.

Outcome	Changes to Simulation to Produce Outcome	Location
H1	Negative bias or increased noise in identity estimate	Visual sensors, classification algorithms (if applicable)
	10% bias in proximity estimate (under right conditions)	Characterization sensors, control station log of obstacle locations
	Confusion of control actions between operator input and obstacle implementation	Control station, obstacle
	Incorrect reporting of obstacle state	Control station, obstacle
	No control input to obstacles	Control station, network
H2	Positive bias or increased noise in identity estimate	Visual sensors, classification algorithms (if applicable)
	Confusion of control actions between operator input and obstacle implementation	Control station, obstacle
H3	No control input to obstacles	Control station, network
H4	Increased noise or bias added to identity and proximity estimates	Sensors, classification algorithms (if applicable)

The combination of the type of change made to the simulation and the corresponding location within the Silverfish system drive the selection of resiliency solutions that would mitigate the hazardous condition mapped to the change. For example, a resiliency strategy that would address the lack of control input to the obstacles associated with the network would be the inclusion of a backup communication system to control the obstacles. Whereas a strategy that addresses the same change to the simulation, but a different location within the system, would be the inclusion of diverse hardware components within the control station that rotate responsibility for sending commands to the obstacle over the network. Following this pattern, resiliency

solutions are identified for each of the remaining simulation changes and corresponding locations in Table IX.

Table IX. Potential Resiliency Solutions mapped to simulation changes.

Change to Simulation	Location	Resiliency Solution
Bias or increased noise in identity estimate	Visual sensors	Redundant camera system with lesser performance
	Classification algorithms (if applicable)	System parameter assurance
10% bias in proximity estimate (under right conditions)	Characterization sensors	Triple redundant acoustic sensors for increased confidence in proximity measurement
	Control station based log of obstacle locations	System parameter assurance
Confusion of control actions between operator input and obstacle implementation	Control station	Diversely redundant, hopping command sending capability
	Obstacle	Two-factor command authorization
Incorrect reporting of obstacle state	Control station	System parameter assurance
	Obstacle	Operational consistency checking for obstacle feedback
No control input to obstacles	Control station	Diversely redundant, hopping command sending capability
	Network	Backup communication network

The list of potential resiliency solutions is now consolidated into a mapping of each solution to the locations for implementation, the hazardous condition(s) to be mitigated, and the associated Blue Team adverse outcomes. This mapping is shown in Table X.

Table X. Resiliency solutions mapped to their locations for implementation and mitigated hazardous conditions.

Resiliency Solution	Location	Mitigated Hazard(s)	Associated Blue Team Outcomes
Redundant camera system with lesser performance	Visual sensors	H1, H2, H4	1.2, 2.1, 2.2
System parameter assurance	Control station, control station based log of obstacles, classification algorithms	H1, H2, H3, H4	1.2, 2.1, 2.2
Triple redundant acoustic sensors for increased confidence in proximity measurement	Characterization sensors	H1, H4	1.2, 2.1, 2.2
Diversely redundant, hopping command sending capability	Control station	H1, H2, H3	1.1, 1.2, 1.4, 3.2
Two-factor command authorization	Obstacle	H1, H2	1.1, 1.4
Operational consistency checking for obstacle feedback	Obstacle	H1	1.4, 3.1
Backup communication network	Network	H1, H3	1.2, 2.1, 2.2, 3.2

4.4 Evaluation of Identified Resiliency Solutions

Given the set of resiliency solutions identified in the previous step, each solution is now evaluated in terms of the risk-based framework described in Section 3.4. The impact of each solution is a factor based on the number of adverse outcomes addressed, the priority of those adverse outcomes, and the solution's effect on system operation. The priority of outcomes is defined as the average likert-scale priority ranking of the associated Blue Team adverse outcomes. As stated in Section 3.4, some resiliency solutions may be possible to represent in the simulation. For this particular application, the only solution solutions immediately representable within the simulation are the ones addressing increased noise or bias within the system's sensors.

For example, the triple-redundant acoustic sensors lowers the amount of noise perceived by the system, and allows for a sensor giving bad measurements to be voted out. However, despite this solution's apparent effect on the accuracy of proximity measurements, the operation of Silverfish is not majorly affected by an increase in precision from its acoustic sensors.

Silverfish relies heavily on the visual surveillance from the cameras monitoring the denied area. Furthermore, the operator uses his or her own judgment for target identification and characterization- which is in itself an effective resiliency measure. The ability of the operator to maneuver within the denied area to make "eyes-on" assessments of agents within the denied area reduces the impact of increasing noise or bias to the sensors used for surveillance. However, it should be noted that, in the feasible near-future scenario where target identification and classification is automated, the impact of solutions that mitigate the introduction of noise or bias to the system's sensors increases significantly. Table XI presents an overall impact rating for each resiliency solution based on the three factors mentioned above along with a rationale for the rating of the solution's effect on the system.

Table XI. Impact ratings for identified resiliency solutions.

Solution	# of outcomes addressed	Priority of outcomes	Solution Effect	Overall Impact rating
Redundant camera system with lesser performance	3	1.67	Effect diminished due to ability of operator to visually confirm	Medium
System parameter assurance	4	1.67	Confidence in accuracy of state estimations increased, changes easily detected	High
Triple redundant acoustic sensors for increased confidence in proximity measurement	2	1.67	Minimal effect due to ability of operator to visually confirm	Low
Diversely redundant, hopping command sending capability	3	2	Assurance that commands are not altered within the control station is enhanced	High
Two-factor command authorization	2	1	Assurance that obstacles only perform legitimate commands is enhanced	Medium
Operational consistency checking for obstacle feedback	1	2	Assurance that obstacle is reporting the correct feedback enhanced	Low
Backup communication network	2	2	Backup network allows mission to continue if primary network goes down, huge impact on ability to complete mission	High

Following the classification of each resiliency solution's impact, the likelihood of each solution's outcomes to be mitigated is assessed. This likelihood measure is based on the number and severity of the changes made to the behavior simulation to achieve adverse outcomes. Table XII presents each resiliency solution mapped to these two factors.

Table XII. Likelihood ratings for identified resiliency solutions.

Solution	# of changes to achieve adverse outcome	Severity of changes	Overall Likelihood Rating
Redundant camera system with lesser performance	2	Low- difficult to achieve needed amount of noise or bias to affect behavior	Low
System parameter assurance	3	High- simple changes drastically affect system behavior	Medium
Triple redundant acoustic sensors for increased confidence in proximity measurement	2	Low- difficult to achieve needed amount of noise or bias to affect behavior	Low
Diversely redundant, hopping command sending capability	3	Medium- simple changes to affect system behavior, but difficult to achieve	Medium
Two-factor command authorization	1	Medium- simple changes to affect system behavior, but difficult to achieve	High
Operational consistency checking for obstacle feedback	1	Medium- simple changes to affect system behavior, but difficult to achieve	High
Backup communication network	1	High- simple changes drastically affect system behavior	High

Now that each solution's impact and likelihood ratings have been recorded, each solution can be classified into a "risk" prioritization category as described in Section 3.4. Table XIII presents each resiliency solution's impact and likelihood ratings along with its prioritization category.

Table XIII. Resiliency solutions classified in their prioritization categories.

	Solution	Impact Rating	Likelihood Rating	Prioritization Category
A.	Redundant camera system with lesser performance	Medium	Low	Low
B.	System parameter assurance	High	Medium	High
C.	Triple redundant acoustic sensors for increased confidence in proximity measurement	Low	Low	Low
D.	Diversely redundant, hopping command sending capability	High	Medium	High
E.	Two-factor command authorization	Medium	High	High
F.	Operational consistency checking for obstacle feedback	Low	High	Medium
G.	Backup communication network	High	High	High

Figure 8 presents this same information in the risk matrix figure from Section 3.4.

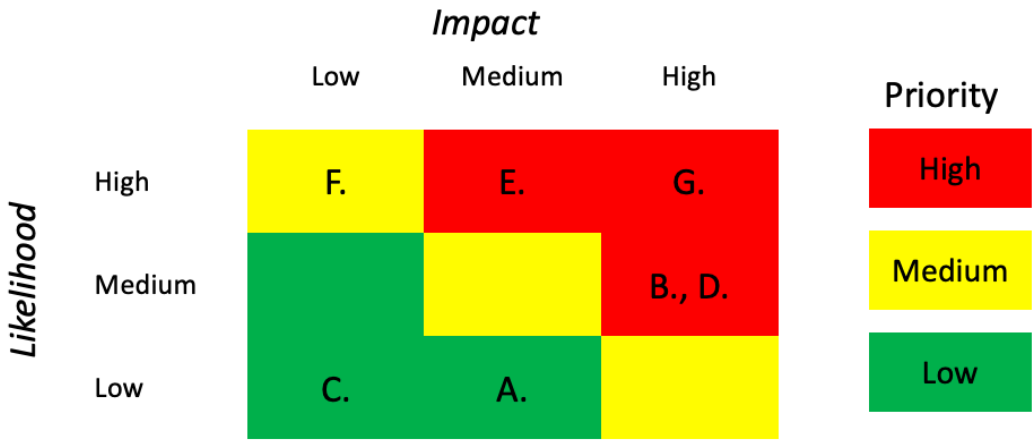

Figure 8. Resiliency solutions mapped to their position in the risk matrix.

As seen in the above table and figure, the STRAT identified a total of seven resilience strategies appropriate for the Silverfish system. Of these seven, four are recommended to receive high priority for consideration for implementation, one for medium priority, and the remaining two should receive low priority. The low priority strategies, triple redundant acoustic sensors and

a backup camera system, received a low ranking due to the nature of the Silverfish system. Since Silverfish uses a human-in-the-loop for identifying agents within the denied area and determining which obstacle to fire (in the event that the agent is an enemy), the likelihood that noise or bias in the sensor feedback causes incorrect behavior is minimal. In the event that the operator is unable to identify a target using the sensors, he or she will likely resort to visual confirmation, which makes the system resilient against friendly fire incidents at the slight risk of enemy agents traversing the denied area before they can be positively identified and engaged.

The medium-priority solution, operational consistency checking for obstacle feedback, received its ranking due to the ease with which adverse outcomes could be achieved by altering the system's perception of the obstacles states. More specifically, the concern behind this resiliency solution involves the breakdown of mission function if the system believes that an obstacle has been fired when it has not. Such a scenario could enable an adversary to traverse the denied area unimpeded if there were a pathway created by obstacles thought to have been fired.

The high-priority solutions labelled D and E in Table XIII involve measures to ensure that the commands sent by a system component match the commands received by the intended recipient. It is clear how such mis-matches would result in unintended behavior. The solution labelled B in Table XIII involves ensuring that changes to the algorithms, decision models, or other parameters are detected and accounted for before any system processes are adversely affected.

Finally, the highest priority solution for consideration for implementation is the introduction of a backup communication network. Without a working network, the Silverfish system loses all functionality and cannot complete its mission.

4.5 Comparison to CSRM Results

Comparing the results of the CSRM to the STRAT must first be qualified by the difference in goals between the two methodologies. First and foremost, the CSRM intends to develop cyber security requirements for a developing system. Such requirements include the incorporation of both resilience and traditional security solutions into the system design. The STRAT, on the other hand focuses solely on the identification and evaluation of resiliency strategies. Consequently, the comparison of results between the two methodologies will be limited to the CSRM's recommendations for resiliency. Secondly, the CSRM leverages domain experts throughout the process, which provides excellent credibility in its results, but limits its applicability in practice due to scheduling and budget constraints. The STRAT utilizes domain experts to a lesser degree in the hope that model-based evidence can provide a similar level of credibility without consuming as much manpower.

The potential resilience strategies identified by the CSRM included resilient weapon control capabilities, diverse communications sub-systems, and resilient situational awareness capabilities. Based on the inputs from the Blue and Red Teams, the CSRM identified the system's communication subsystems (network) as the top priority area for resiliency. The resilient weapon control and situational awareness capabilities incorporated a variety of solutions such as diverse redundancy, confidence testing, and situational awareness introspection. The STRAT also identified the network as the top priority for resiliency, and specified that resiliency should be achieved through redundancy. The other resiliency strategies however, can be classified as sub-strategies of the weapon control and situational awareness categories identified in the CSRM.

In general, CSRM identified a broader selection of resiliency strategies than the STRAT. This could be a result of the abstraction of system hardware components in the STRAT's definition of the control structure and system model. CSRM defined the existence of certain hardware components explicitly in SysML representations, which likely aided the identification of resiliency strategies such as separating situational awareness information from weapon control functions.

The similarity of the top priority recommendations in the CSRM and the STRAT suggests that the STRAT is compatible with existing methodologies. Since STRAT does not involve as many domain experts as the CSRM, the STRAT could be a viable alternative technique for recommending a prioritized set of resiliency strategies. Both methodologies, however, are limited in that neither currently account for the cost of resiliency.

5. Conclusions

Existing techniques for identifying and evaluating security solutions for CPS are lacking. Most security enhancements to CPS utilize perimeter defenses, which fail to account for the complex physical and component interactions inherent to this application. Cyber-resiliency strategies have the potential to plug this gap in capability, however, identifying which resiliency strategies are appropriate for a given system and choosing the most effective solutions is an on-going area of research.

This thesis presented a methodology, termed the Systems-Theoretic Resiliency Assessment Tool, that uses concepts grounded in systems and control theory along with model-based systems engineering to identify a set of resiliency strategies appropriate for a given system and provide a prioritized recommendation for implementation. This work builds upon the Cyber Security Requirements Methodology and uses its results for comparison. Consequently the

STRAT uses the same hypothetical system described in the CSRM as a case study for the demonstration of approach.

The major contribution of this thesis is the increased scalability offered to cybersecurity efforts for CPS. The CSRM, one of the only methodologies currently documented that shares similar goals to the STRAT, was a project that involved a team of roughly 15 people operating in different roles and lasted for a duration of six months. Not counting the shared initial steps of both methodologies and the consideration that the CSRM did not utilize the full time of all the people involved, the STRAT produced similar recommendations to the CSRM in only a month of work and utilized only one researcher, compared to a team of experts. This decreased labor requirement could also allow experts to be used for confirming the results of the STRAT, which decreases their time commitment to a single project. Furthermore, the modeling techniques described in the STRAT are designed to enable automated analysis given future research. This characteristic would allow the STRAT to be applied to systems significantly more complex than the Silverfish system described in this thesis.

A second contribution of this work is the potential of creating a common methodology for the design of secure physical systems within the defense and other industries. A major concern in the development of new systems is consistency in risk management with respect to cybersecurity across different projects. The STRAT outlines a process through which cyber resilience is addressed and managed in the early stages of a system's lifecycle. If used, the STRAT would allow a common resilience risk management technique to be applied to different projects. The use of a common technique simplifies the auditing and review process for new designs. Furthermore, the use of the STRAT in the preliminary design phase for new systems

could allow for earlier interaction with Operational Test and Evaluation teams, thus reducing the potential of a new system requiring major overhauls after the completion of the design phase.

5.1 Future Work

The STRAT defines a formulation of a graphical system model, referred to as the S-graph. The S-graph shares characteristics to a quiver, or multidigraph, yet this classification has not been proven. The specification of different types of vertices and edges complicates the mathematical definition of the S-graph. Future work could involve the formalization of the S-graph definition. Furthermore, the current application of the S-graph is limited to forming the foundation of the system's behavior model in Simulink. Defining the mathematics of the S-graph and developing an algorithm for analyzing its behavior would allow for automated analysis. For complex, real-world systems, automated analysis methods will be necessary to perform resiliency assessments in a timely manner.

Other future work could involve the quantification of definitive metrics for prioritizing resiliency strategies. The current methodology relies on using qualitative metrics to categorize solutions into a prioritized recommendation. One of the stated limitations to the risk-based framework used to generate the prioritized recommendations is the lack of a probabilistic assessment of likelihood. Integrating methods that take into account the capabilities of adversaries would increase the veracity of the prioritized recommendations for implementation.

Finally, the biggest gap in current literature regarding the assessment of resiliency solutions is the inability to accurately quantify the cost of a particular resilience strategy. Identifying a method to incorporate cost, which includes both monetary and time costs, into resilience strategy assessments would close the loop in determining the proper choice of solutions given real-world constraints on system designs.

References

[1] B. Krebs, “Equifax hackers stole 200k credit card accounts in one fell swoop,” https://perma.cc/64EM-SAZF, accessed: 2017-16-09.

[2] “Alert (ICS-ALERT-14-176-02A) ics focused malware (update a),” https: //ics-cert.us-cert.gov/alerts/ICS-ALERT-14-176-02A, accessed: 2016-12- 05.

[3] “Advisory (ICSA-10-201-01C) usb malware targeting siemens control software (update c),” https://ics-cert.us-cert.gov/advisories/ ICSA-10-201-01C, accessed: 2017-10-05.

[4] K. Baldwin, Systems security engineering: A critical discipline of systems engineering, INCOSE INSIGHT 12 (2009), 11–13.

[5] A. Strafaci, “What does bim mean for civil engineers,” CE News, Transportation, 2008.

[6] N. Leveson, “A new accident model for engineering safer systems,” Safety Science, vol. 42, no. 4, p. 237–270, 2004.

[7] N. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety, MIT, 2011.

[8] B. Nuseibeh and S. Easterbrook. Requirements engineer- ing: a roadmap. In Proc. of the IEEE Int. Conf. on Soft. Eng. (ICSE), pages 35–46, 2000.

[9] N. Leveson, “A new approach to system safety engineering,” in Aeronautics and Astronautics. Cambridge, MA: Massachusetts Inst. Technol., 2002.

[10] I. Sommerville and G. Kotonya, Requirements Engineering: Processes and Techniques, John Wiley & Sons, 1998.

[11] N. Maiden (1998). CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements. Automated Software Engineering, 5(4): 419-446.

[12] P. Loucopoulos & E. Kavakli (1995). Enterprise Modelling and the Teleological Approach to Requirements Engineering. International Journal of Intelligent and Cooperative Information Systems, 4(1): 45-79.

[13] A. Dardenne, A. v. Lamsweerde, & S. Fickas (1993). Goal-Directed Requirements Acquisition. Science of Computer Programming, 20: 3-50.

[14] J. Helming et al., “Towards a unified Requirements Modeling Language,” in 2010 Fifth Int. WS on Requirements Eng. Visualization, 2010, p. 53–57.

[15] G. Disterer, "ISO/IEC 27000, 27001 and 27002 for Information Security Management", Journal of Information Security. 2013.

[16] G. Sindre and A.L. Opdahl, “Eliciting Security Requirements by Misuse Cases,” Proc. 37th Int’l Conf. Technology of Object-Oriented Languages and Systems, p. 120-131, 2000.

[17] C.B. Haley, J.D. Moffett, R. Laney, and B. Nuseibeh, “A Framework for Security Requirements Engineering,” Proc. 2006 Software Eng. for Secure Systems Workshop with the 28th Int’l Conf. Software Eng., p. 35-41, 2006.

[18] C.B. Haley, J.D. Moffett, R. Laney, and B. Nuseibeh, “Arguing Security: Validating Security Requirements Using Structured Argumentation,” Proc. Third Symp. Requirements Eng. for Information Security with the 13th Int’l Requirements Eng. Conf., 2005.

[19] S. Fenz, A. Ekelhart, "Verification, Validation, and Evaluation in Information Security Risk Management," Security & Privacy, IEEE , vol.9, no.2, pp.58-65, March-April 2011.

[20] A. Shostack, Threat Modeling: Designing for Security, John Wiley & Sons, 2014.

[21] B. Schneier, “Attack trees: Modeling security threats,” Dr. Dobb’s Journal, December 1999.

[22] N. Mead, F. Shull, J. Spears, S. Heibl, S. Weber, and J. Cleland-Huang, “Crowd Sourcing the Creation of Personae Non Gratae for Requirements-Phase Threat Modeling,” 2017 IEEE 25th International Requirements Engineering Conference (RE), 2017.

[23] W. Young and N. Leveson, “Systems Thinking for Safety and Security,” in Proceedings of the 29th Annual Computer Security Applications Conference (ACSAC 2013). ACM, 2013, p. 1–8.

[24] P. M. Beach, R. F. Mills, B. C. Burfiend, B. T. Langhals, & L. O. Mailloux, “A STAMP-Based Approach to Quantifiable Meausres of Resilience,” The 16th International Conference on Embedded Systems, Cyber-Physical Systems, and Applications (ESCS’18). ASCE, 2018, p. 103-109.

[25] I. Friedberg, K. McLaughlin, P. Smith, D, Laverty, & S. Sezer, “STPA-SafeSec: Safety and security analysis for cyber-physical systems,” Journal of Information Security and Applications, 34(2017): 183-196.

[26] R. Jones & B. Horowitz, “System-Aware Cyber Security,” ITTNH, 2011 Eighth International Conference on Information Technology: New Generations, April 2011, p. 914-917.

[27] H. Goldman, R. McQuaid and J. Picciotto, "Cyber Resilience for Mission Assurance," in Proceedings of the 2011 IEEE Conference on Technologies for Homeland Security, Waltham, MA, 2011.

[28] R. Jones & B. Horowitz (2012) "A System-Aware Cyber Security architecture." Systems Engineering 15(2):224–240

[29] B. Horowitz, P. A. Beling, C. H. Fleming, S. A. Adams, B. T. Carter, T. Sherburne, C. R. Elks, G. Bakirtzis, F. Shull, N. Mead, "Cyber Security Requirements Methodology," Systems Engineering Research Center, Technical Report SERC-2018-TR-110, July. 2018.

[30] C. H. Fleming, "Systems theory and a drive towards model-based safety analysis," in 2017 Annual IEEE International Systems Conference (SysCon), Apr. 2017, p. 1–5.

[31] H. Derksen, J. Weyman, Quiver representations, Notices of the American Mathematical Society. 52(2), 200–206 (2005).

[32] PR Garvey, ZF Lansdowne. Risk matrix: an approach for identifying, assessing, and ranking program risks. Air Force Journal of Logistics 1998; 22(1):18–21.