DISSERTATION

ELICITING CYBERSECURITY GOALS FOR CYBER-PHYSICAL SYSTEM CONCEPTUAL DESIGN

Submitted by

Martin "Trae" Span

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2025

Doctoral Committee:

Advisor: Jeremy Daily

Thomas Bradley
Steve Simske
Dan Wise

ABSTRACT

ELICITING CYBERSECURITY GOALS FOR CYBER-PHYSICAL SYSTEM CONCEPTUAL DESIGN

This research contributes to the systems engineering body of knowledge by advancing security by design for Cyber-Physical Systems (CPS). It leverages Systems Thinking and Model-Based Systems Engineering (MBSE) methodologies to address both organizational and technical challenges in early-stage secure system development. The research is structured around two primary themes: (1) What recommendations can improve CPS Design Teams with respect to security? and (2) Proving secure system design be improved through early system security goal elicitation.

To address the first research question, a systematic analysis utilizing Systems Thinking tools, such as iceberg models, causal loop diagrams, and system modeling, is conducted. These analyses identify the root causes of weak security design within CPS development teams, revealing systemic organizational challenges, ineffective mental models, and gaps in team member knowledge skills and abilities. The research presents targeted recommendations to enhance security considerations within design teams by implementing Systems Thinking principles, refining organizational structures, and prioritizing security training. However, findings indicate that training alone is insufficient for achieving secure CPS design, necessitating a more structured approach to security design consideration elicitation in early system development.

The second research question is answered with the development of Eliciting Goals for Requirement Engineering of Secure Systems (EGRESS), a novel methodology designed to facilitate system security goal elicitation during the conceptual design phase of CPS. By addressing a critical gap in current systems engineering practices, EGRESS provides a structured and traceable approach to defining security goals before an architecture is established. This method incorporates best practices from Systems Thinking, loss-driven engineering analysis, and MBSE to ensure secu-

rity is foundational in CPS design rather than an afterthought. Furthermore, the research evaluates the applicability of the Risk Analysis and Assessment Modeling Language (RAAML) standard for cybersecurity and proposes refinements to enhance its utility for security analysis in CPS design.

The key contribution of this work utilizes Popper's falsification principle to evaluate the hypothesis that secure system design can be improved through early security goal elicitation. Given the lack of long-term operational data proving increased security over a system's lifecycle, falsification serves as a rigorous alternative by testing for refutation rather than statistical verification. The research demonstrates that EGRESS cannot be falsified, supporting its validity in improving secure system design. This claim is further reinforced through peer-reviewed evaluations and expert discussions within the system engineering and security communities, where, through publication, the methodology's utility was recognized and endorsed.

Beyond methodology development, this research contributes to the broader systems engineering body of knowledge by addressing the distinction between requirements and security-focused system goals. It also explores the balance between common and custom SysML profiles to improve security goal elicitation. These contributions collectively support the advancement of more resilient and secure CPS architectures, aligning with the broader vision of integrating security as a fundamental design consideration alongside functionality and safety.

ACKNOWLEDGEMENTS

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the U. S. Air Force, the Department of Defense, or the U.S. Government.

I want to thank God for his continued blessing on my life and for ordaining my path and the completion of this work. I would be remiss not to thank my wife, Katie, for her support of my 'unique' path and work schedule during these past few years. I am grateful for your patience and steadfast backing. I want to thank my family for the constant support throughout my life and encouragement to achieve and push myself into the best version I can be.

I must thank my advisor, Dr. Jeremy Daily. Your mentorship and friendship has been instrumental in my PhD endeavor. I am extremely grateful for your support and for our adventures together on this path to obtaining my PhD. To my research group peers and friends, Thank You! Gabe Salinger for all the collaborative thinking sessions and contagious positive attitude. Mars Rayno for the relentless editing comments and assistance. Sarah Rudder for your expert MBSE and Cameo skills and help with both. To the rest of the team, thanks for letting me learn about vehicle networks and truck hacking. The group is accomplishing great things!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction and Background

1.1 Motivating the Problem

In the rapidly evolving landscape of technology and interconnected systems, security has become a paramount concern. This concern goes well beyond traditional Information Technology (IT) roots; as systems grow in complexity and integration, the need for robust security requirements becomes increasingly critical. A significant shift has occurred in the last 50 years with functionality being largely delivered previously through mechanical actuation to a high reliance now on software and electrically implemented functionality [4]. This functionality change has resulted in a majority of complex systems now being Cyber-Physical Systems (CPS), where the mechanical physical functionality is largely delivered or controlled via an electronic, cyber-based approach. Modern CPSs are increasingly interconnected in the Industrial Internet of Things (IIOT). The push for the interconnectedness of devices increases functionality and convenience features, but these also increase the vulnerability profile for both accidents and malicious actors. More than ever, a systems approach is needed to address the complex emergent system properties induced by increased interconnectedness, software, and electronic functionality.

The use of cyberspace has followed a logical path beginning in the 1960's with the realization that computers can be exploited to reveal sensitive data. The pace of this development has increased over the decades, and beginning in the 1990s, it became apparent that adversaries also had the capability to attack our cyber-enabled systems and thus began an increased focus on developing and defining cybersecurity for systems. CPSs are an increasingly popular cyberattack target. Global critical infrastructure has been under attack since the Stuxnet attack on the Iranian Nuclear Power Plant in 2010 [5]. CPS vulnerabilities have been increasing every year. According to the Claroty report [6], there has been a 41 percent increase in the CPS vulnerabilities year over year.

Current military conflicts and non-state ransomware attacks have targeted CPS beyond traditional weapons systems and IT networks [7]. Reports of attacks on the power grid and generation CPS, medical devices, and vehicles [8] highlight the importance of securing CPSs design.

Considering these threats, systems engineering must have appropriate tools and techniques for performing systems security engineering (SSE) and analysis. For example, recent United States Department of Defense (U.S. DoD) – which has historically valued systems security – updates have expanded assessments for all cyber-physical weapon systems [9], [10], and efforts to standardize "Secure Cyber Resilient Engineering" (SCRE) are underway [11]. These efforts guide and standardize elements of integrating security efforts into existing systems engineering processes for cyber-physical system acquisitions, and work to ensure security considerations hold equal footing with other requirements and design trade-offs at major program milestones.

In recent years, SSE has undergone a revitalization as the Department of Defense (DoD) has refocused efforts in SSE for the acquisition of new systems. This effort began in force with the 2012 SSE revitalization paper [9] published by Kristin Baldwin, the Acting Deputy Assistant Secretary of Defense for Systems Engineering. In 2015, the International Council of Systems Engineering (INCOSE) dedicated a section in the INCOSE handbook version 4.0 [12] for SSE which was updated in the current version 5.0, and in 2016 the National Institute of Science and Technology (NIST) published NIST SP 800-160 volume 1 focusing on SSE. The NIST SP 800-160 volume 1 was revised and released with updates in 2022 [13]. These advances in the study and application of system security in recent years are representative of the increasing criticality of security in complex cyber-physical systems, and the importance of defining and maintaining security requirements in the systems engineering lifecycle processes. As our reliance on remote capabilities and cyber-physical systems continues to increase, the need is to secure these assets as much as practicable.

Beyond the Department of Defense, the motor vehicle industry is also increasingly interested in the cybersecurity of vehicles, both automobiles and heavy trucks. The importance of cybersecurity has increased as the reliance on software-based functionality continues to grow. It is not uncommon for a vehicle to now have over 100 million lines of code [14]. This need has been further

acknowledged and acted upon with the formation of Information Sharing and Analysis Centers (ISAC) for Automobiles [15]. These communities of interest were created to share and analyze intelligence about emerging cybersecurity risks to the vehicles and to collectively enhance vehicle cybersecurity across the global automotive industry. Additionally, Dr. Jeremy Daily, at Colorado State University, started a CyberChallenge hands on immersive learning opportunity to recruit and train young talent for cybersecurity. The current cyber challenge events include: CyberAuto, CyberTruck, CyberTractor, and CyberBoat. The most successful of these is CyberTruck which brings together nearly all major retailers of heavy trucks along with most of the leading SME's in the vehicle security space. Dr. Daily and I published the findings on this model of success for cyber education and manufactuer collaboration at the INCOSE International Symposium 2024 [16].

While initiatives exist to generate change in cyber-physical system security, more work is needed in both a systems approach to holistic analysis of emergent security properties but also a change in the design teams themselves designing and fielding cyber-physical systems.

1.2 Research Question Introduction

This dissertation provides novel contributions centered around two primary research questions:

1. What recommendations can improve Cyber-Physical System (CPS) Design Teams with respect to Security?

2. Hypothesis: Secure system design can be improved through early system security goal elicitation.

The investigation and research in answering question one resulted in the conclusion that training people is effective but not a universal solution to improving CPS design with respect to security. Research question one's conclusions lead to research question two; a better method for capturing CPS security goals in early system design is necessary to provide a more universal solution to secure CPS design challenges. Research question two presents the hypothesis that secure system design can be improved through a method to elicit system-level goals for security before a pre-

liminary architecture is available. The Popper falsification approach, described in Section 1.4, is used in providing evidence of the inability to disprove this hypothesis. The ability to prove system design is truly more secure would be beyond the scope of this work as it would require a system to be designed using the primary methodology introduced in this work, EGRESS (Eliciting Goals for Requirement Engineering of Secure Systems), with metrics collected over the system's operational lifecycle proving it is more secure than similar systems designed without the elicitation of early security goals. The data to prove the reduced security vulnerability of the system would require many years of data collection to prove objectively the value of EGRESS. While the data over a system lifecycle would be extremely valuable to proving this hypothesis, it is beyond the scope of this dissertation effort. Instead, this work presents evidence that the hypothesis of EGRESS improving secure system design cannot be falsified.

1.3 Dissertation Overview and Contribution Summary

1.3.1 Structure

The structure of this dissertation is as follows:

1. An introduction and extensive background of work relevant to answering the two research questions.

2. RQ1: Recommendations to improve CPS Design Teams with respect to Security

3. RQ2: Introduction to the EGRESS Methodology

4. RQ2: Demonstrations of EGRESS applied to multiple Systems of Interest from different domains.

5. Summary and Conclusions

1.3.2 Contributions

This dissertation provides novel contributions to enhance the field of systems engineering. Specifically, it contributes to the efforts introduced above in enabling 'Security by Design' for complex CPS's.

- (RQ1) Analysis utilizing Systems Thinking tools on the current state of Cyber-Physical System Design Teams - Chapter 2.
- (RQ1) Recommendations, supported by Systems Thinking principles and tools, to improve CPS Design and Design Teams with respect to Security - Chapter 2.
- (RQ2) Definition of a security goal in the context of stakeholder needs, system goals, and system requirements - Section1.5.1.
- (RQ2) Upon conducting an extensive literature review of research on requirements engineering, there is a notable scarcity of resources addressing security requirements from a systems engineering perspective. A consolidated reference for systems engineers integrating insights from both software-centric security engineering and traditional systems engineering is a contribution to the SE Body of Knowledge - The review is dispersed through Section 1.7, and published as standalone work [17].
- (RQ2) Consolidated and clear method to generate system goals with a focus on security for cyber-physical systems (CPS) in conceptual design - Chapter 3.
- (RQ2) Recommendations of how to balance utilizing common vs custom SysML diagrams and languages, and traditional system requirements vs system goals within the security goal elicitation method - Section 4.3.2
- (RQ2) Evidence of the security goal elicitation method, EGRESS, applied to complex CPS design - Chapter 4.

1.4 Popper Falsification Approach

Popper's falsification approach is particularly useful for research hypotheses where data is not available to positively prove them statistically true. It shifts the focus from verification to potential refutation or falsification. This approach aligns with the scientific method's fundamental aspect of testing the null hypothesis, which has its basis in Popper's falsification. The original citation for Karl Popper's falsification principle comes from his book "Logik der Forschung" published in 1934. This work was later translated into English and revised by Popper himself, appearing as "The Logic of Scientific Discovery" in 1959 [18]. In this seminal work, Popper introduced the concept of falsifiability as a criterion for differentiating science from non-science. Popper's key argument can be summarized as follows: "A theory or hypothesis is falsifiable if it can be logically contradicted by an empirical test." His principle emphasizes that scientific theories should make predictions that can potentially be proven false through observation or experimentation [19]. This forms the basis for research question two's approach to provide evidence that the hypothesis: *eliciting security goals improves secure system design* is not able to be proven false from the case study evidence presented.

1.5 Key Terminology

This section begins with some key terminology, followed by a detailed introduction and literature review specific to each Research Question.

Stakeholder: A stakeholder is any entity (individual or organization) with a legitimate interest in the system. It typically includes users, operators, decision-makers, and regulatory bodies [12].

Requirement: The INCOSE handbook defines Requirements as formal structured statements that can be verified and validated [12]. The ISO 29148 defines a requirement as a statement that translates or expresses a need and its associated constraints and conditions [20]. A requirement statement is the result of a formal transformation of one or more needs or parent requirements into an agreed-to obligation for an entity to perform some function or possess some quality (within specified constraints with acceptable risk) [21].

Goal: System Goal: a ‘Goal’ (sometimes called ‘business concern’ or ‘critical success factor’) refers to the overall, high-level objectives of the system. Goals provide the motivation for a system but are often vaguely defined [20]. A system goal is a refinement of a stakeholder need, it addresses an issue that needs resolved. Goals are more overarching than requirements. The system should rather than shall is acceptable for defining a goal since it is often broad and provides context for further specified requirements.

A System goal:

- is an intention regarding the objectives, properties, or use of the system
- is solution independent
- documents the intention of stakeholders
- is in a hierarchical structure
- does not demand a specific form solution or architecture (Goals should be more abstract)

1.5.1 Security and Safety

As the terms "security" and "safety" are often ambiguous, ill-defined, and context dependent, working definitions are proposed in this work. Additionally, the appendage of “cyber” as a prefix to security or systems often adds more confusion; thus, for the purposes of this work, here are the following definitions:

• Safety is freedom from those conditions that can cause death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment. As adapted from MIL-STD-882E and the NASA Safety Handbook [22].

• Security is freedom from those conditions that can cause death, injury, or occupational illness; damage to or loss of equipment or property; damage to the environment; damage or loss of data or information; or damage to or loss of capability, function, or process. According to NIST SP 800-160, volume 1 [13].

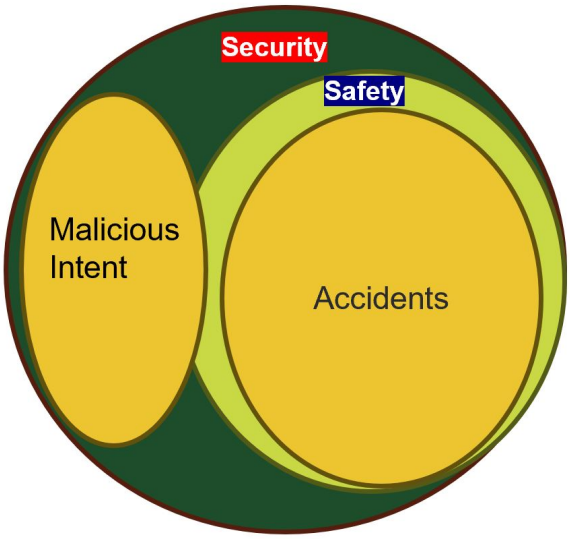

Figure 1.1: The Overlapping Relationship between Safety and Security

Consistent with systems engineering principles, the specific scope of safety and security is defined by stakeholders in terms of the entities to which they apply and the consequences against which they are assessed. Historically, security has typically had a limited scope that only considered malicious actions and tended to focus on the confidentiality, integrity, and availability of data. Likewise, safety has had a limited scope and does not explicitly consider intentional (malicious) actions. However, with increasing frequency, these concerns are being addressed holistically with a growing number of common analysis methods. For example, safety and security approaches can be combined under Systems Thinking approaches by capturing stakeholder-defined unacceptable losses. This overlapping relationship is depicted in Figure 1.1.

1.6 Introduction of Design Teams

This section introduces Cyber Physical System (CPS) design teams and the associated challenges in implementing 'Security by Design'.

1.6.1 The CPS Design Team Challenge

Present-day Cyber-Physical Systems (CPSs) are designed and fielded by teams of engineers and program managers, who often pay inadequate attention to cybersecurity. The structure of the CPS design team requires investment in systems security and Systems Thinking to create a culture that can achieve 'Security by Design'. A complex CPS incorporates several electrical, mechanical, controls, software, and communications subsystems. Many of these subsystems rely on software, Internet, over-the-air, and remote electronic transmissions to perform their intended function. Too often, security design experts are not even included in the design team. Current design teams often view cybersecurity as merely a checklist process that primarily concerns Information Technology (IT) and often do not implement a systems approach to designing for a secure implementation of the system functionality.

In the author's experience, many CPS design teams are not equipped with the mental model to understand security vulnerabilities and assess possible attack paths as 'cybersecurity' typically falls outside the design team's core responsibility. Attackers discover new techniques to exploit the system, but many current CPS design teams lack the expertise to address these issues during the CPS design. Additionally, vulnerabilities are often not known until they are exploited, resulting in dangerous zero-day attacks. Further, attacks against CPSs are often realized as a complex chain of events (i.e., a dynamic process that is difficult to study). Secure system design must shift from primarily low-level tactics-driven employment of control mechanisms to a strategy-driven philosophy for system-level protection [13]. CPS safety and security design teams must be better prepared to face these challenges.

1.6.2 Systems Thinking Introduction

Systems Thinking is founded in General Systems Theory and is applied to a wide range of problems across fields and disciplines [23]. Systems Thinking encourages a holistic perspective to problem-solving that goes beyond traditional reductionist methods. Complicated problems can often be solved by using a reductionist method of decomposing the problem into smaller parts;

however, reductionist approaches are often not sufficient for interactively complex CPSs. Safe and secure CPS designs require a systematic approach that considers each system holistically. Further, while the Systems Engineering Body of Knowledge devotes a chapter to Systems Thinking [24], many systems engineers do not fully grasp its key tenets and utility in problem-solving [23].

Systems thinking has been advanced and proliferated by great minds at prestigious institutions such as the Massachusetts Institute of Technology in Peter Senge's widely published book, *The Fifth Discipline* [25] and Donella Meadows' work *Thinking In Systems: A Primer* [26]. Famously, Meadows' Iceberg model is used to intuitively decompose the root cause of events, going beyond the patterns of behavior to understand the root cause as a part of a flawed mental model. An Iceberg model of CPS design team challenges is presented in Chapter 2 along with a subset of these systems principles. Advocating for the value of visual tools as a part of a blended Systems Thinking and design approach, [27] leverages several visual tools including an Iceberg model, Causal Loop Diagrams (CLD), and MBSE models to decompose and provide recommendations for complex problems.

Recent advancements in MBSE provide a critical development in Systems Engineering and System Design. MBSE's adoption and realization of benefits is a key effort of professional systems engineering organizations such as the International Council on Systems Engineering (INCOSE) [28]. In parallel, systems modeling and architecture have evolved and developed formalized languages, where SysML is the most widely used MBSE modeling language [28]. SysML is built on the popular Unified Modeling Language, which guides readily understandable visual and modeling aids such as block definition diagrams and context diagrams.

While the Systems Engineering Body of Knowledge devotes a chapter to Systems Thinking, [24] many systems engineers do not fully grasp its key tenets and utility in problem-solving [23].

Dr. Kamran Eftekhari Shahroudi of Colorado State University developed specific principles for teaching Systems Thinking and its application to Complex System Development. They build upon fundamental texts, presenting a set of seven key systems principles supported by case studies of real-world system failures [29]. These principles were further expanded into an entire text on

Systems Thinking, *Turbocharge Your Future in a Complex World with Systems Thinking*, in which this dissertation author was a co-author. This book simplifies and condenses the Systems Thinking approach and insights embedded inside many references and case studies. The result is a small set of concise, practice-derived systems principles that are always true and always improve our odds of success when applied to complex real-world challenges. The text provides Systems Thinking philosophy, principles, and rigorous methods [30].

1.6.3 Related Work

Recent research efforts and publications support the growth and application of Systems Thinking and MBSE for solving complex design problems. This section summarizes some of this related work.

Model-Based Systems Engineering has been applied to address complex design problems across various domains. Recent publications include the application of MBSE to autonomous driving, [3], aircraft maintenance systems, [31], and even the application of MBSE for design, development, and deployment of the Internet of Things (IoT) in smart city applications [32].

In [33], the authors investigate the effect of managerial experience on an individual's Systems Thinking skills toward complex problem-solving. Based on the authors' investigation, managerial experience alone does not ensure expertise in Systems Thinking skills to solve complex problems. Organizations need to develop a Systems Thinking culture through training and practice to solve complex organizational problems.

In [34], Ray Jonkers proposes an integration of project management and systems engineering by applying Systems Thinking principles to address project management problems leading to cost overruns, project delays, and design inconsistencies. This work presents a flight management simulator to represent a digital twin of SE based design, design change, knowledge, and agile project management practices. It incorporates Systems Thinking principles to achieve a holistic view of the flight simulator by bringing different elements together. These elements include data,

sub-models, cross-functional processes, and the technical, social, economic, and cultural factors affecting system, organizational, and project performance.

Based on the related work discussed, most of the research effort is focused on the application of Systems Thinking and MBSE for solving complex design problems, rather than complex organizational problems.

1.7 Background on System Security Goal Elicitation

Research question two is the following hypothesis: *Secure system design can be improved through early system security goal elicitation.*

1.7.1 Introduction

"Unless security is designed into a system from its inception, there is little chance that it can be made secure by retrofit..." [35]. This quote is from a 1972 report on the future of computer security. These words ring true 50 years later as the problem of secure system design continues to increase in importance. The Systems Engineering community has acknowledged this gap in security requirements. The International Council on Systems Engineering (INCOSE) Vision 2035 sets goals for systems engineering as a whole in enabling engineering solutions for a better world. A key objective outlined is "Cybersecurity will be as foundational a perspective in systems design as system performance and safety are today" [36]. To achieve this vision statement and address security designed in from system inception, system goals for security must be elicited and translated into functionality focused security requirements for CPS design. A Government Accountability Office Report on Weapon System Cybersecurity from 2021 [7] identifies a glaring omission of cybersecurity requirements in major weapon system contractual documents. It states not only is an effort to include cybersecurity required, but that these requirements should be treated as other types of system level requirements. The systems engineering community continues to focus on and improve system requirements elicitation [21] as requirements provide the design to specifications that are verifiable and testable to ensure the system meets the user needs.

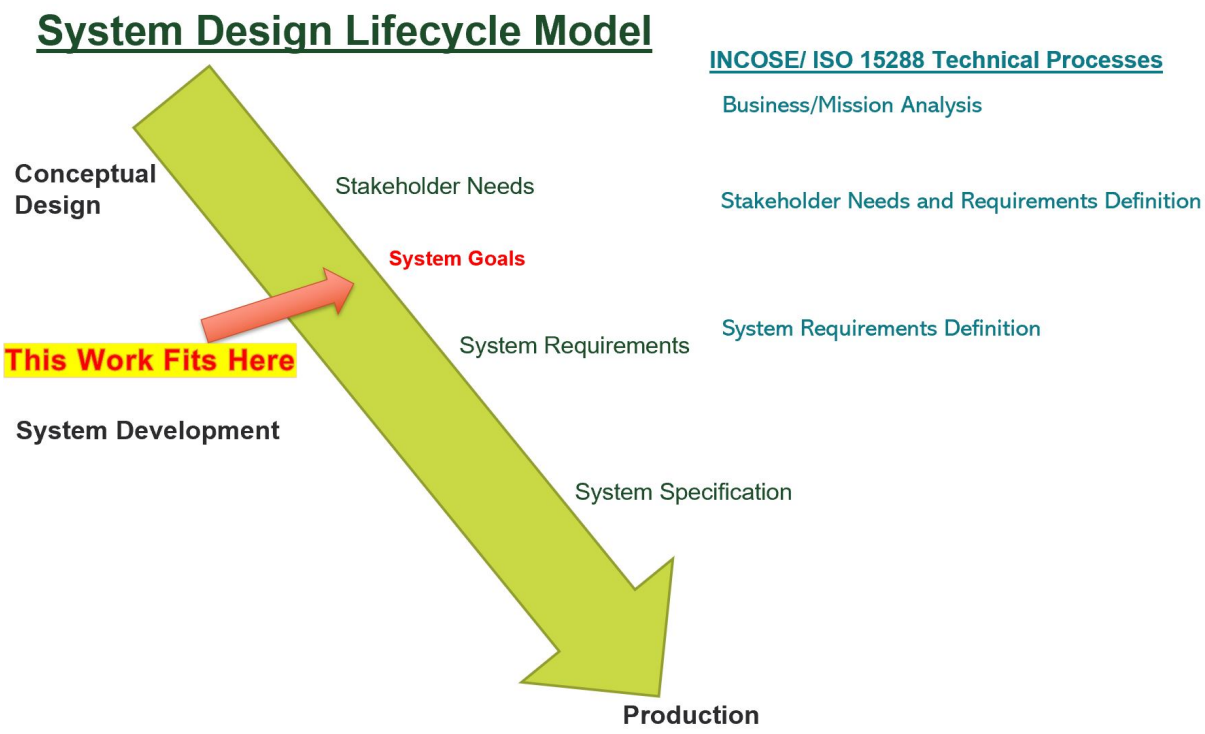

Figure 1.2: Focused Contribution in the SE Lifecycle

1.7.2 Identified Need and Gap in the SE Body of Knowledge

In a survey of the literature, many methods, tools, and techniques exist for conducting security threat and hazard assessments. However, these almost always required preliminary or detailed design architectures. Upon further study, a clear gap was evident in conceptual design phase guidance, techniques, and methods for security requirements elicitation. Figure 1.2 shows the focus area in the system design lifecycle for this research. A method is needed ***before*** a preliminary architecture is available for the system of interest (SoI). A method usable in conceptual design of a system best addresses the INCOSE Vision 2035 need for security to be as foundational of a requirement in system design as functionality and safety.

This work focuses on contributing an approach to generate security goals in the conceptual design phase of system development for Cyber-Physical System design. These goals maximize the system design trade space by specifying goals for security that influence early system design decisions and form the baseline of further security requirement elicitation throughout the system design process, ultimately achieving a system that is secure by design.

1.7.3 Current Landscape of System Security for CPS Design

In the rapidly evolving landscape of technology and interconnected systems, security has become a paramount concern. This concern goes well beyond traditional Information Technology (IT) roots; as systems grow in complexity and integration, the need for robust security requirements becomes increasingly critical.

Complex systems, such as those found in critical infrastructure, are characterized by their interdependencies and interfaces. Unlike traditional IT systems, which primarily focus on availability, integrity and confidentiality, these complex systems must ensure the safety, reliability, and resilience of both the cyber components and physical components of these Cyber-Physical Systems (CPS). A security breach in these systems can have serious consequences, ranging from personally identifiable information (PII) theft, endangering of human lives, to national security.

Security requirements in complex systems extend beyond traditional IT systems by addressing unique challenges, examples of which include cyberattacks involving the military and ransomware targeting corporations and government entities. Systems engineers, who are responsible for the design, implementation, and maintenance of these systems, must recognize that security is not an isolated concern, nor one capable of being truly addressed by an after-the-fact patch, but is an integral aspect of the overall system architecture. By incorporating security requirements early in the system development lifecycle, systems engineers can identify and mitigate potential vulnerabilities, ensuring the robustness of the final system.

This brings about a need for systems engineering to have a tool kit capable of conducting systems security engineering (SSE). Efforts to create a standard for Secure Cyber Resilient Engineering (SCRE) are being conducted [37]. As Cyber-Physical Systems constitute the majority of modern complex systems, the systems engineer must consider the implications of cyber threats on physical processes and vice versa. An example of this is the J1939 Network Protocol, used in heavy trucks, buses, industrial machinery, and military vehicles. If an attacker has access to the Controller Area Network (CAN) of the vehicle, many different attacks are possible that can directly affect the physical vehicle itself, several of which are detailed by Chatterjee et al. [38].

This interdisciplinary viewpoint is crucial for developing comprehensive security strategies that safeguard both digital and physical assets. As systems become more interconnected and reliant on advanced technologies such as artificial intelligence (AI) and autonomous control systems, the potential attack surface expands, making security requirements even more critical.

1.7.4 Literature Review Approach

An extensive literature review was conducted across systems engineering standards, methods, and best practices for security requirements elicitation. Much literature is available on requirements engineering but little on security requirements from a systems perspective. More work is available on security requirements engineering but has a security engineer focus. This work attempts to combine relevant security engineering and traditional systems engineering into a consolidated reference of works of interest for systems engineers developing security requirements as a part of their early system design.

This work surveys relevant standards, security focused MBSE methods and developments, Systems Theory Approaches, and Security Engineering Work found relevant to the authors' pursuit of improving system security requirements. It is likely this survey does not capture every potentially relevant work, but it provides a consolidated source of useful reference material for systems engineers attempting to improve security requirements elicitation.

First, key systems engineering standards that guide requirements elicitation are reviewed. Then Systems Theory approaches including Systems Thinking and Systems Theoretic Process Analysis (STPA) [39] use for security requirements is explored. Next, the use of MBSE for Security Requirements is addressed [40]. Finally, security-focused literature is reviewed: systems security standards and security requirements engineering practices. This work does not assess each of the methods, but some commentary is provided on the utility of the methods the authors have utilized. The intent is to provide a consolidated source of potential reference material for systems engineers seeking best practices for requirements development in support of secure system design.

1.7.5 Key Systems Engineering Standards

The key guidance documents influencing the system development and requirements elicitation efforts in this work are the ISO/IEC/IEEE 15288 (System Life Cycle Processes) [41], ISO/IEC/IEEE 29148 (Requirements Engineering) [20], and the INCOSE Handbook for Systems Engineering [12].

INCOSE Technical Processes

INCOSE defines 14 Systems Engineering Technical Processes from ISO/IEC/IEEE 15288 [41] in the Systems Engineering Handbook [12]. These technical processes enable systems engineers to elicit system needs and shape the design across the engineering disciplines to deliver a system that not only meets specifications but the stakeholder needs. These technical processes begin with conceptual design and work through the entire lifecycle of design, development, and operation through disposal. Technical processes enable systems engineers to coordinate between specialists, various engineering disciplines, systems stakeholders, and operators. These processes enable the creation of a sufficient set of system requirements and the resulting system solution that delivers capabilities within the bounds of performance, environment, interfaces, and design constraints. ISO 29148 Systems and Software Engineering Life Cycle Processes details the processes and products involved in engineering requirements throughout the system life cycle. It provides guidance for constructing well-formed requirements in the context of system and software engineering.

Of primary interest to this work are the first three INCOSE technical processes:

1. Business or Mission Analysis
2. Stakeholder Needs and Requirements Definition
3. System Requirements Definition

The purpose of the Business or Mission Analysis is to define the business or mission problem or opportunity and characterize the solution space or potential solution classes that could address the problem or take advantage of the opportunity. In this technical process, the problem domain is

defined, major stakeholders are identified, and the system environment and other major constraints are elaborated. This technical process generates system needs, that are refined into Stakeholder needs. A key activity of this technical process is a translation of the organization's concept of operations (CONOPS) into the development of a system operational concept (OPSCON) and initial system requirements.

The next INCOSE ISO15288 technical process is stakeholder needs and requirements definition. The purpose of this process is to define the stakeholder requirements for a system that can provide the capabilities needed by the stakeholders [41]. Stakeholder requirements govern the system's development and are essential in defining the scope of the development. The main activities in this process include determining stakeholders, defining stakeholder needs, transforming those stakeholder needs into stakeholder requirements, and developing the operational concept (OPSCON).

CONOPS and OPSCON

Often, the activities and terminology of CONOPS and OPSCON development can be confused between the two efforts. ISO 29148 provides guidance for CONOPS and OPSCON Development. The CONOPS describes the assumptions or intent in regard to the overall operation or series of operations of the business using the system to be developed with other existing systems and possible future systems. This is often a strategic or long-range plan. It serves as the basis and elaboration of the overall background and needs for the system to operate in as stakeholder requirements are elicited. ISO 29148 separately defines a System Operation Concept (OPSCON) as a document describing what the system will do (not how it will do it) and why (rationale). The OPSCON describes system characteristics and communicates overall quantitative and qualitative system characteristics to the acquirer, user, supplier, and other organizational elements [20]. The OPSCON contains both the business needs and the stakeholder needs. It is iteratively refined through the process of system requirements definition and the architecture definition. An OPSCON is often based on the use and understanding of current or similar systems the new system is replacing, if those exist. The ISO 29148 gives detailed guidance for the development of a System OPSCON.

An OPSCON should include the following:

- operational policies and constraints
- description of the proposed system
- modes of system operation
- user classes and other involved personnel
- support environment

Operational Policies and Constraints are management decisions regarding the operations of the system, normally in the form of general statements or understandings that guide decision-making activities. Policies limit decision-making freedom but do allow for some discretion. Operational constraints are limitations placed on the operations of the system [1].

The Description of the Proposed System is the lengthiest portion of the OPSCON and often the portion most detailed in the first iteration of an OPSCON development. When no current system exists this section describes the situation that motivates the proposed system. This section should detail the operational environment and its characteristics including major system elements and the interconnection among those elements. Interfaces to external systems or procedures should be described. Capabilities, functions/services, and features of the system should be enumerated. If possible, charts and accompanying descriptions depicting inputs, outputs, data flows, control flows, and manual and automated processes sufficient to understand the system or situation from the user's point of view should be included. Additionally, the cost of system operations, operational risk factors, and performance characteristics, such as speed, throughput, volume, frequency, and workload should be included as available. Any quality attributes available should be included such as availability, correctness, efficiency, expandability, flexibility, interoperability, maintainability, portability, reliability, reusability, supportability, survivability, and usability, along with any other logistics requirements to support the system. Any known considerations about safety, security, privacy, and integrity should be captured. The method proposed in this work helps identify those.

Modes of System Operation describe the various modes of operation for the system or situation (e.g., operational, degraded, maintenance, training, emergency, alternate-site, peacetime, wartime, ground-based, flight, active, and idle modes). All modes that apply to all classes of users should be included. Important modes to include are degraded, backup, and emergency modes, if such exist. This is especially true if these modes involve different geographical sites and equipment that have a significant impact on the operational aspects of the system [20].

User Classes and other Involved Personnel A user class is distinguished by the ways in which users interact with the system. Factors that distinguish a user class include common responsibilities, skill levels, work activities, and modes of interaction with the system. Different user classes may have distinct operational scenarios for their interactions with the system. In this context, a user is anyone who interacts with the existing system, including operational users, data entry personnel, system operators, operational support personnel, software maintainers, and trainers [20].

The Support Environment describes the support concepts and support environment for the current system, including the support agency or agencies, facilities, equipment, support software, repair or replacement criteria, maintenance levels and cycles, and storage, distribution, and supply methods [20].

The OPSCON is a key input and starting point for this work's System Security Goal elaboration. Its information is the formal input that allows for the execution of the method presented in this work. However, it is noted that this work overlaps in helping elaborate items that should be included in a fully detailed system OPSCON, particularly those aspects that are relevant to the development of system security needs and requirements. This work provides utility in iterating and developing an OPSCON further based on stakeholder inputs and the resultant system description and modeling efforts.

Need The Oxford English Dictionary defines a need as a thing that is wanted or required. For a system these are often capabilities or things that are lacking but desired by one or more stakeholders [12]. System Needs are translated and refined into system requirements. Per the INCOSE Handbook, Requirements are formal structured statements that can be verified and validated [12].

The stakeholder needs and requirements definition process is followed by the system requirements definition process. The purpose of this step is to transform the stakeholder view of desired capabilities and requirements into a technical view of the solution with specific system requirements that meet the operational needs of the user [41]. The output of the method in this work would be an input to this process, using the high level security goals as an input to traditional systems security requirements elicitation as a part of the Threat Analysis and Risk Assessment (TARA) activities. System Requirements form the basis for the architecture, design, integration and verification of the system. The system requirements definition process generates a set of system requirements using the stakeholder requirements developed in the stakeholder needs and requirements definition process.

Requirements Engineering

ISO 29148 Systems and Software Engineering - Life Cycle Processes - Requirements Engineering, details the processes and products involved in engineering requirements throughout the system life cycle. It offers a structured approach for the elicitation, analysis, specification, validation, and management of requirements throughout the life cycle of a system. ISO 29148 covers various aspects of requirements engineering, including the identification of stakeholders, the definition of system boundaries, the handling of changes, and the maintenance of requirement traceability.

The standard emphasizes the importance of clear, consistent, and traceable requirements to ensure that the final system meets stakeholder needs and performs as intended. This leads to improved system quality and reduction of risk. This standard is aligned with the INCOSE Systems handbook and the ISO 15288 system technical processes [20].

INCOSE provides further guidance in the INCOSE Requirements Writing Guide [21]. This guide gives practitioner guidance on best methods to eliciting and managing system requirements. It is not security focused but does address security as a system requirement.

1.7.6 Security-Focused Standards and Methods

In the author's search for security requirements engineering literature, one previous survey paper stood out, *A comparison of security requirements engineering methods [42].* This work was the most recent survey the author could find, and it is approaching 15 years old. However, it does an excellent job of providing a detailed summary of the primary methodologies for security requirement engineering as of 2010. Fabian et al. highlight that the security requirement engineering process must support engineers in identifying the security goals of the stakeholders. The paper emphasizes that the process of converting stakeholder needs to requirements is extremely crucial to system success:

> "This process must establish a coherent set of security requirements for the entire system, which is complete and consistent within itself and with the other kinds of requirements that are relevant for the system."

The authors also discuss that security requirements are consequences of the identified threats to the system. And that all security requirements must be done before the design of the system, because of the influence that these requirements may have on the final design. Fabian et al. discuss that in the security requirement engineering community, the distinction between security goals (abstract) and security requirements (more detailed) is often not completely precise. This work was more specific to security engineering and did not consider other aspects of systems engineering relevant to secure system design and requirements elicitation.

System Security Guiding Standards

This section addresses recent and influential system security standards and then surveys relevant security-focused requirements engineering methods.

NIST 800-160 Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems

The NIST 800-160 provides guidance for integrating security into the engineering processes of complex systems. It emphasizes a systems-oriented approach to security, addressing both technical

and organizational aspects. It is perhaps the most impactful document for informing a systems approach to the security of systems beyond traditional IT and computer networks. NIST 800-160, [13], outlines best practices for incorporating security throughout the system development life cycle, from conception and design to deployment and maintenance. It provides insight for developing more resilient and trustworthy systems capable of withstanding a wide range of security threats and vulnerabilities.

ISO/SAE 21434 The ISO/SAE 21434 [1], released in 2021, compiles cybersecurity development guidance for vehicles. It provides vocabulary, objective, requirements and guidelines for cybersecurity engineering. It articulates a process for cybersecurity activities beginning with concept definition, extending into product development, verification, validation, production and operations and maintenance for the system of interest.

A very important subsection of ISO 21434 is the Threat Analysis and Risk Assessment (TARA). The steps of a TARA include: asset identification, threat scenario identification, impact rating, attack path analysis, attack feasibility rating, risk value determination, and risk treatment decision. This process is used to generate requirements in system design and development, but it assumes a level of system architecture is already in place. The challenge this work will address is how to abstract its processes to be able to perform a subset of its steps in the conceptual design phase of system development. For threat scenario identification as a part of the TARA, it encourages the use of threat modeling approaches such as EVITA [43], TVRA [44], PASTA [45], STRIDE (spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege) [46].

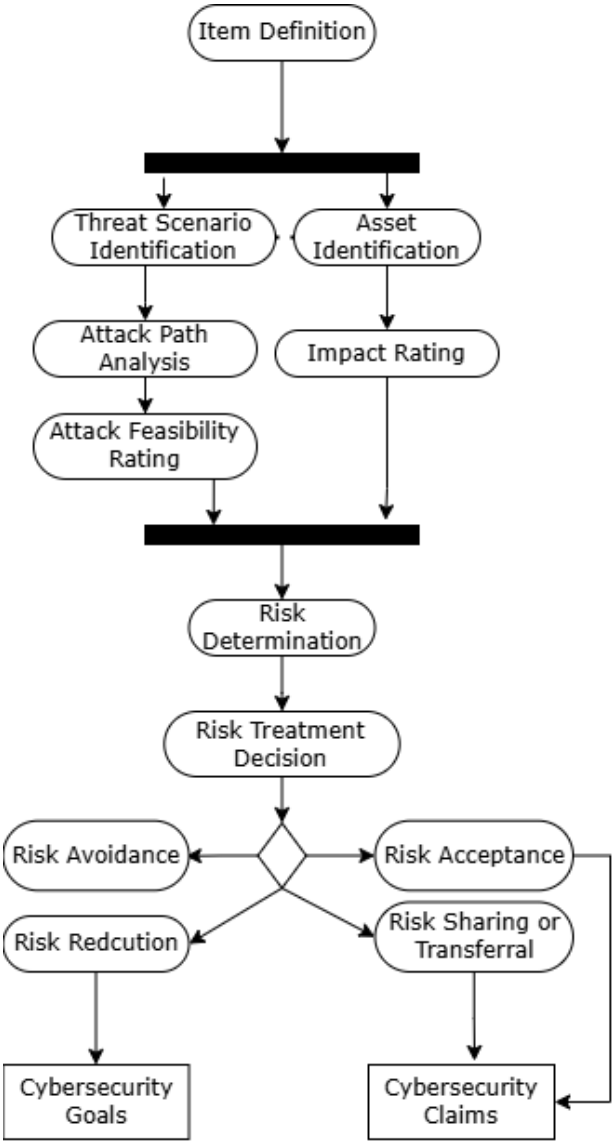

Figure 1.3: Threat Analysis and Risk Assessment (TARA) Workflow Redrawn from ISO21434 [1]

Security Requirements Engineering:

This section surveys Security Requirements Engineering literature for best practices. These methods are in line with the practices recommended in this work, however; previous work was largely tailored to the computer science and software engineering communities and not for cyber-physical system development. The subsequent literature review highlights various notable approaches employed for security requirement engineering (SRE), along with their respective limitations.

Security Requirements Engineering Process (SREP)

The Security Requirements Engineering Process (SREP) [47] is a systematic methodology designed to address the need for the early integration of security considerations into system development. This method is designed to ensure that security requirements are effectively identified, documented, and managed throughout the system development life cycle. SREP involves structured activities like security risk assessments, threat modeling, and the selection of security controls to mitigate identified risks. It is a foundational methodology in the field of Security Requirements Engineering (SRE), and emphasizes the proactive inclusion of security requirements in the development process to foster a security-centric system design. One of the key contributions of SREP is the implementation of a Security Resource Repository, which is based on the idea of reusing the security requirements proposed. The approach aims to build and manage security knowledge, assure the quality of security work, and focus on security early in the requirements stage of development through reuse, generic threats, and requirements from a shared repository. Along with the repository, SREP outlines the following activities to be followed during the development process:

1. *Agree on Definition:* Define stakeholders, security policies, and the organization's vision, forming the foundation for overall security requirements.
2. *Identify Vulnerable and/or Critical Assets:* Identify vulnerable and critical assets (Typically done by requirements engineer).
3. *Identify Security Objectives and Dependencies:* Involves identifying security objectives and dependencies using the predefined repository. Documented through security policies and a Security Objective Document.
4. *Identify Threats and Develop Artifacts:* Focuses on finding threats that could target assets and developing artifacts like misuse cases, attack tree diagrams, and UMLSec use cases.
5. *Risk Assessment:* Estimates security risk based on relevant threats, their probability of occurrence, and negative impacts, with risk impact and likelihood both categorized on a scale of 1-5 in terms of severity.

6. *Elicit Security Requirements:* Corresponding requirements or clusters of requirements are identified for each threat, and additional security-related requirements are discovered.
7. *Categorize and Prioritize Requirements:* Categorizes and prioritizes security requirements based on the impact and likelihood analysis of threats.
8. *Requirements Inspection:* Validates the quality of teamwork and deliverables, assessing the overall security requirements engineering process.
9. *Repository Improvement:* Updates to the repository are made with new and modified elements.

It is important to note that SREP's effectiveness is limited by the expertise and quality of threat modeling conducted, making it potentially challenging for inexperienced practitioners to identify and prioritize security risks adequately.

Security Requirements Engineering: A Framework for Cyber-Physical Systems

This work introduces a security requirements engineering framework for cyber-physical systems called Cyber-Physical Systems - Security Resource Repository (CPS-SRR) [48]. This method is an extension of SREP and is designed to create a secure CPS development process. The purpose of the work is to demonstrate that SREP has many advantages, and is effective when extended to Cyber-Physical Systems. The authors introduce the framework as a centralized approach to security requirements engineering that allows for the storage of key information about the system and its domain and can be easily modified throughout the process of iteration. The framework involves the progression through 11 activities very similar to that of SREP, and it aimed at developing security requirements, specifically for cyber-physical systems:

1. Agree on definitions and gather security goals
2. Identify vulnerable and/or critical assets
3. Define security objectives and dependencies
4. Enumerate threats and develop artifacts
5. Domain analysis and vulnerability identification
6. Risk Assessment

7. Develop Patch Management
8. Elicit Security Requirements
9. Categorize and prioritize requirements
10. Requirements inspection
11. Repository improvement

As the steps in this approach progress towards eliciting security requirements in Step 8, the activities in the initial part of this framework are security goal definition, asset identification, security objective development, and threat and vulnerability identification. This top-down approach is facilitated effectively by beginning with defining security goals in Step 1, whereas other approaches do not begin with them at all. Once the security goals and critical assets are defined, the methodology progresses to identifying threats and vulnerabilities and employs misuse cases and attack trees to develop artifacts. However, modeling is only added during Steps 4 and 5 instead of the onset of goal definition. Establishing a model from the genesis of system development will substantially improve the effectiveness of the model by incorporating traceability into the security design processes.

Security quality requirements engineering methodology (SQUARE)

SQUARE is a comprehensive methodology for security requirement engineering [49]. It is primarily used for software development and provides an organizational framework for carrying out security requirement engineering activities. It is comprised of 9 steps:

1. Agree on Definitions
2. Identify security goals
3. Develop artifacts
4. Perform risk assessment
5. Select elicitation technique
6. Elicit security requirements
7. Categorize requirements
8. Prioritize Requirements

9. Requirements inspection

This framework offers a very effective means of developing security requirements and follows the foundational elements defined by SREP, with validation built in as exit criteria for each step. However, there is an identified lack of resources or approaches with regard to eliciting security goals from stakeholders in an efficient and effective manner (Steps 1 and 2).

Engineering Security Requirements

This work's primary findings state that "Most Requirements engineers are poorly trained to elicit, analyze, and specify security requirements, often confusing them with the architectural security mechanisms that are traditionally used to fulfill them" [50]. They end up specifying design constraints rather than true security requirements. The common problem with specifying security requirements is that they tend to be accidentally replaced with security-specific architectural constraints that may unnecessarily constrain the most appropriate security mechanisms for the system. To combat this problem, Firesmith introduces 12 types of security requirements that are useful for guiding the requirements engineers and the process of eliciting precise security requirements. These 12 include:

1. Identification Requirements
2. Authentication Requirements
3. Authorization Requirements
4. Immunity Requirements
5. Integrity Requirements
6. Intrusion Detection Requirements
7. Nonrepudiation Requirements
8. Privacy Requirements
9. Security Auditing Requirements
10. Survivability Requirements
11. Physical Protection Requirements
12. System Maintenance Security Requirements

The author highlights that often security control mechanisms are selected as requirements and may unnecessarily over-constrain the system implementation. Therefore, the requirements must have clear guidelines and specifications, in order to not lead to any preemptive design decisions.

Security Requirements Engineering: A Framework for Representation and Analysis

Haley et al. [51] claim that Security Requirements must satisfy three criteria: *definition*, *assumptions*, and *satisfaction*. While the work is focused on software security requirements, the need for a systems approach is clearly highlighted. The paper discusses the importance of context in a requirement, i.e. confidentiality could be solved by protecting the actual data physically (locked door). They propose to generate security goals by first enumerating all system assets, then postulating actions that would violate security concerns for the assets, i.e. brainstorm threats and threat vectors. Then, a set of security goals can be generated by listing the actions that need to be taken to prevent (or avoid) the threats to the assets.

Other Relevant Security Work

Misuse cases: UML-based approach

As opposed to the traditional use case approach in systems engineering, UML-based Misuse cases identify behavior not wanted in the system to be developed. Simply put, misuse cases are caused by those who seek to misuse the system. This approach requires the modeling of an abbreviated use case diagram which is composed of use cases and actors, as well as misuse cases and misusers. This approach helps brainstorm and conceptualize other security requirements and goals that must be addressed. The five steps of this iterative method are as follows:

1. Identify critical assets in the system
2. Define security goals for each asset
3. Identify threats for each security goal
4. Identify and analyze risks for the threats
5. Define security requirements

While the utilization of misuse case diagrams is very effective in this stage of system development, the approach does not cover requirement elicitation based on the misuse cases, ensuring all requirements are included, validation, verification, conflicting requirements, or the interplay between security and other non-functional requirements.

STRIDE - Threat Modeling Designing for Security

The STRIDE method [52] was proposed by Microsoft and represents a mnemonic for six different types of security threats that a system may encounter. STRIDE helps identify and categorize threats by focusing on six main dimensions: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. This structured approach helps security professionals analyze and mitigate potential security risks in software and system design by considering these threat categories. Each threat category relates to a specific security theme that is violated. Table 1.1 outlines the relationship between the six threat dimensions and their associated security theme.

Table 1.1: Stride Threat Classification

Security Threat	**Security Theme**
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-Repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

In simple terms, the STRIDE security threats are described in the following manner [46]:

Spoofing: Pretending to be someone or something you are not in the system.

Tampering: Changing or messing with real information.

Repudiation: Denying you did something in the system when you actually did.

Information Disclosure: Unauthorized access to secret data or a data breach.

Denial of Service (DoS): Stopping legitimate users from using a service.

Elevation of privilege: Gaining more access rights in the system when you shouldn't have them.

This framework plays a vital role in systems security engineering and modeling by providing a structured and systematic approach to identifying and categorizing security threats. This framework is particularly valuable for several reasons. Firstly, it helps security professionals and system engineers to comprehensively assess and classify potential risks and vulnerabilities associated with a system. Secondly, STRIDE establishes a common language for discussing and documenting these threats, which is essential for effective communication among various stakeholders involved in system development. Furthermore, it assists in risk assessment, enabling organizations to prioritize and address the most critical security concerns.

1.7.7 Systems Theory Related Work

Systems Thinking

Systems thinking principles are a helpful tool in elaborating system security requirements. A systems thinker views the entire system holistically, in an effort to design out vulnerabilities in CPS implementations, obscure information dependencies, and even emergent system properties which are open to exploitation. [13] provides an example of this application of Systems Thinking to the security problem. Further, incorporating MBSE techniques with this Systems Thinking approach to address security problems in CPS design teams is discussed in [53].

STPA for Security

System Theoretic Process Analysis (STPA) is a widely adopted method for performing safety analysis on systems, viewing them functionally in order to derive control actions (requirements) to bound the system into a safe operating state. STPA's foundation is based upon the System Theoretic Accident Model and Process, STAMP, introduced by Nancy Leveson of MIT in her book, Engineering a Safer World [54]. Leveson's group at MIT hosts an annual conference on the use of STPA to solve complex problems in system design. They created a handbook on the use of STPA [55]. Within the last ten years, STPA has been extended to perform security analyses of

systems. The seminal work in this extension [39] explains the overlap between safety and security design considerations and proposes the value of STPA to generate security requirements for complex system design and development. STPA's application to security has many recent publications on the application to different domains and system types. Many of the authors' previous works demonstrate the utility of STPA in generating security requirements for system design. Initial works [56], [57] proposed a new organization of an STPA for Security, STPA-Sec, to use for security requirements analysis of complex cyber-physical systems. This initial work culminated in a full demonstration of this method applied to conceptual security analysis of an aerial refueling concept [58]. In the years since this initial effort, this concept has been expanded to applications of STPA-Sec in additional domains to include: autonomous space systems [59], small unmanned aerial systems [60], and an application to the DEVSECOPS reference architecture [61]. STPA-Sec's utility to solving the system security requirements elicitation problem is well proven and as such is a key contributor to answering the research questions in this work. A key value of STPA is the lack of concern for a specific current threat. Threats change across a lifecycle, so modeling against the threats of today is not sufficient for a system that will be fielded for 10-50 years.

1.7.8 Use of MBSE for Security Requirements

Recent advancements in MBSE provide a critical development in Systems Engineering and System Design. MBSE's adoption and realization of benefits is a key effort of professional systems engineering organizations such as INCOSE [12]. In parallel, systems modeling and architecture have evolved and developed formalized modeling languages utilized in various modeling methods and tools. According to Campo et al., the systems engineering community views model-based systems engineering favorably based on an analysis of over 60 references discussing the approach. The extensive review of SE literature found a 4:1 positive to negative attribute ratio for MBSE [62]

MBSE Introduction

Systems modeling typically requires three fundamental pillars; A language, a tool, and a method [63]. Systems Modeling Language (SysML) is the widely accepted modeling language

that is associated with MBSE [64]. Created by Object Management Group (OMG) and supported by INCOSE, it is an object-oriented graphical modeling language for building models of complex and multi-disciplinary systems [65]. SysML is built on the Unified Modeling Language, which provides guidance for readily understandable visual and modeling aids such as block definition diagrams and context diagrams. Of note, SysML v2 is currently in the final stages ofdevelopment,t as some portions have been released for Beta testing and adoption. Recently, the Object Modeling Group (OMG) released the Risk Analysis and Assessment Modeling Language (RAAML). RAAML defines formal extensions to SysML to support safety and reliability analysis, some of which are also being used for security analysis and requirements generation. Dassault Systèmes company created a Systems Cybersecurity plugin [66] largely based on RAAML to assist systems engineers in developing security requirements and completing threat assessments and hazard analysis for their system using Model-Based Systems Engineering. Other semi-formal modeling languages exist such as the Specification and Description Language (SDL), and Architecture and Analysis Description Language (AADL) [67], but are not as common in the systems engineering field.

The following are some commonly used tools for implementing SysML; Enterprise Architect, Cameo, Rhapsody, and TTool. Above supporting the standard SysML diagrams, these tools all maintain the capability of requirement expression and traceability, use case analysis, and model simulation [67].

MBSE Methods

In a recent work, D. Mazeika et al. [68] explored the model based systems engineering approaches for security analysis and identified the most prominent methods for using MBSE for secure design. Based on the conclusions in the Mazeika work, only the methods shown in Table 1.2 are highlighed. Note, this work is not intended to be an exhaustive survey or full comparison of MBSE Security methods.

UML Sec

Table 1.2: MBSE Security Methods

Method	**Primary Usage**
UML Sec [69]	Uses security classes, associations, and constraints to model security policies, access control mechanisms, and security protocols
Secure UML [70]	Models Role-based Access Control in UML
SysML-Sec [71]	Characterizes threats and attacks through attack trees and ending with hardware and software partitioning
CHASSIS [72]	Uses UML-based diagrams and text-based techniques to facilitate its three main steps: Eliciting functional requirements, eliciting safety/security requirements, and specifying safety/security requirements
MBSE-Sec [68]	MBSE-Sec outlines the activities and steps for designing secure systems with its own SysML security profile

UML Sec is an extension of the Unified Modeling Language (UML) that provides a set of security-related constructs, such as security classes, associations, and constraints. It is designed to help analysts and developers model the security aspects of a system, such as security policies, access control mechanisms, and security protocols [69]. UML Sec uses standard UML diagrams to specify security requirements and attack scenarios. UML Sec is in concept a great idea to use an MBSE extension for secure system development, but again it fails in that it assumes a known system architecture at some level and is based on known threats. It, like many other methods, starts with an assumption that a set of security requirements already exist.

Secure UML

In the work, SecureUML: A UML-Based Modeling Language for Model-Driven Security [70], Loggersmith et. alium. present a modeling language for the model-driven development of secure, distributed systems based on UML. The approach is based on role-based access control. They show how UML can be used to specify information related to access control in the overall design of an application. This work is the least like the others reviewed for MBSE based security requirements elicitation but is relevant with respect to access control.

SysML-Sec

SysML-Sec is an extension of SysML that focuses on security analysis. The SysML-Sec methodology aims to facilitate the collaboration and communication between experts in system design and experts in system security [71]. It is development focused and captures requirements through a goal-oriented approach, and captures architecture through a model-oriented view. SysML is very helpful for modeling security requirements, threats, and architectures. The SysML process involves developing high-level security requirements, followed by characterizing threats and attacks through attack trees and ending with hardware and software partitioning.

CHASSIS

The Combined Harm Assessment of Safety and Security for Information Systems (CHASSIS) is a method for requirement generation that relies on use cases and sequence diagrams [72]. CHASSIS relies on UML-based diagrams and text-based techniques to facilitate its three main steps: Eliciting functional requirements, eliciting safety/security requirements, and specifying safety/security requirements [68]. CHASSIS emphasizes the benefit of a combined safety and security analysis process including a broader context than a separate analysis might use.

MBSE-Sec

The MBSEsec method is the most promising of the MBSE methods surveyed. The author's previous work utilizes MBSE-Sec and assesses its utility to improve secure system design [40]. MBSE-Sec outlines the activities and steps for designing secure systems with the SysML security profile. The method involves four phases that provide the appropriate techniques and diagrams to be used. The first phase involves identifying and capturing security requirements in a SysML requirements diagram. The second phase involves defining the structure of the system by capturing and allocating assets using the MBSEsec profile. The third phase outlines the process for modeling behavioral and structural threats and risks through misuse case, activity, and threat definition diagrams. Lastly, the fourth phase defines security control objectives and controls. This is done with the use of activity and definition diagrams. The MBSEsec method introduces an efficient process for creating secure systems while respecting ISO/IEC 27001 requirements [68]. This method demonstrates value with clear and distinguished phases of design and the perceived effectiveness

of iteration throughout the method. While some of the other methods focused heavily on requirements or partitioning, MBSEsec presented the most holistic and balanced approach to developing a secure system.

1.8 Summary

This introduction and literature review captures the current state of the body of knowledge relevant to answering the research questions of this dissertation:

1.8.1 Summary of Literature Review for Research Question 1:

What recommendations can improve Cyber-Physical System (CPS) Design Teams with respect to Security?

- Key Terminology - Safety and Security have an overlapping relationship such that security includes all safety-driven loss prevention design considerations but also includes malicious intent loss considerations.
- The CPS Design Team Challenge - Cyber-Physical System Design Teams often lack both Systems Thinking and system security knowledge
- Systems Thinking has merit for solving complex problems, yet has not been applied extensively to CPS design, or the design team composition itself.

A need exists to apply Systems Thinking approaches to improving the CPS design team composition, knowledge, skills, and abilities to accomplish secure by design CPS development.

1.8.2 Summary of Literature Review for Research Question 2:

Hypothesis: Secure system design can be improved through early system security goal elicitation.

- Identifies a need and gap in the SE Body of Knowledge for security design considerations before a preliminary system architecture is in place. Security needs to be a functional requirement, but a gap exists for eliciting security requirements in early system design.

- Security focused standards and methods - Standards provide clarity on the types of security considerations to address but all methods and tools surveyed did not address how to elicit security requirements without a system architecture. In fact most threat and hazards analysis techniques needed detailed system design knowledge to be performed.

- Systems Theory related work provides principles that guide secure design considerations. Specifically, STPA provides a promising loss driven approach, demonstrating utility for requirements and goal elicitation system design.

- MBSE is widely used for requirement elicitation and definition in system design, but few methodologies apply it specifically to security. Those that are available were surveyed and while demonstrating some utility, they did not enable security requirement elicitation before the system architecture was defined.

A clear need and gap in the literature exists for eliciting system security goals in early system design before a preliminary system architecture is defined.

Chapter 2

Improving Cyber-Physical System (CPS) Design Teams

2.1 Introduction

In this chapter, Systems Thinking and Model Based Systems Engineering (MBSE) is used to solve complex organizational problems in a case study of a CPS design team. SE and MBSE with SysML and Causal Loop Diagrams (CLD's) are used to demonstrate needed changes in the mental model of a CPS design team to evolve as cyber conscious CPS design team. This chapter answers research question one: *What recommendations can improve Cyber-Physical System (CPS) Design Teams with respect to Security?*

Two papers were published answering this research question and form the majority of the content in this chapter. The first publication was at the IEEE International Systems Engineering Conference, ISSE 2022, titled *Systems Thinking and Model Based Systems Engineering's Utility to Solve Complex Organizational Problems - Cyber-Physical System Design Teams* [73]. The second publication is *Considerations for Cyber-Physical Design Teams Tasked with Engineering Safe and Secure Systems for a Notional Electrified Aircraft Concept* which was published at the IEEE Systems Conference, SYSCON 2023 [53].

2.2 A Characterization of the Problem Space Using Systems Thinking Models

This section demonstrates the use of Systems Thinking, both principles and models, to characterize a complex organizational problem. It begins with an analysis of the CPS design team problem concerning several Systems Thinking principles. Next, an iceberg model is used to illustrate the CPS design team problem. Finally, a causal loop diagram is presented that captures the

complexity of interactions in the current problem space of insecure system design practices in CPS design Teams.

2.2.1 Root Cause Analysis of the CPS Design Team Cybersecurity Problem Space

This section explores the complexity of the CPS design team challenge using the classic Iceberg model [26]. The Iceberg model is a simple tool for gaining a deeper understanding of a problem by seeing beyond the obvious trigger event(s) (e.g., CPS outages caused by successful cyber attacks). The iceberg model guides an understanding of the buried root cause by identifying patterns of behavior associated with the event, uncovering the underlying system structure responsible for creating those patterns, and Uncovering the underlying mental model – in multiple levels of detail – to better understand the underlying structures which lead to the undesired, hazardous, or harmful event.

Figure 2.1 details a root cause analysis of traditional CPS security design team failings. This multilevel analysis reveals a number of critical systems and security shortcomings that must be addressed in order to design safe and secure CPSs.

Events

"Events" are observable unwanted, undesired, or potentially hazardous occurrences. They do not necessarily have to be malicious or harmful but are merely an indication of a problem which needs to be further investigated. As indicated by the "Iceberg" analogy, often times the actual problem is much larger (and more complex) than is initially observable, and most of the challenge is actually hidden from plain sight. In the case of CPS design teams, the "event" is witnessed through successful cyber attacks which occur against CPSs of all scope, scale, and industry. Recent world events demonstrate the ability of malicious actors to continue to exploit not only IT networks but critical CPS devices and CPS-enabled System of Systems (SoS) [74]. Too often, CPS devices fall victim to cyber attacks, resulting in unwanted disruption, delays, denial of services, or worse; thus, the analysis continues.

Patterns

At this level of the root cause analysis, patterns of behavior are reoccurring unwanted activities. In this analysis, patterns represent shortcomings of traditional CPS designs and teams that result in vulnerabilities and cyber attacks. For example, system specifications that simply state: 'the system shall be secure' fail to understand the importance of cybersecurity for modern connected systems. Poor CPS security patterns are evidenced by the increasing number of vulnerable CPS devices being developed and the widespread exploitation of everything from simple home-monitoring video cameras to advanced automobiles in the popular media [6].

Observed patterns within CPS design teams may include: pressure to quickly bring new devices to market; an over-narrow focus on desired functionality without sufficient concern for security requirements and/or privacy considerations; treating security as an afterthought; reliance on IT personnel to address system-level security issues; trusting in IT-based checklist security approaches; the ability to patch CPS devices as required in the future to fix problems; a limited understanding or refinement of stakeholder needs for safety or security.

Perhaps most importantly, key systems thinking tenets of holism and emergence are often not considered. However, attackers view CPS holistically and seek out novel and unique ways to exploit any vulnerabilities, not just the ones the designers plan for. Historically, ignoring emergent behaviors can result in exploitable vulnerabilities of which malicious actors can take advantage [5]. Thus, security is an emergent property where vulnerabilities often result from unanticipated behaviors resulting from interactions between and within elements and users of the System of Interest (SoI). These interactions must be modeled and studied, allowing the CPS design team to gain a more complete understanding of the CPS security problem.

Systems Structure

The totality of the systems structure can be wide in scope. The focus is on the design teams charged with securing new CPS systems. Typical structural concerns include entities such as: CPS companies and their design teams; national policies and regulations; external operational context and existing infrastructure; and even social structures and personal motivations are all contributing

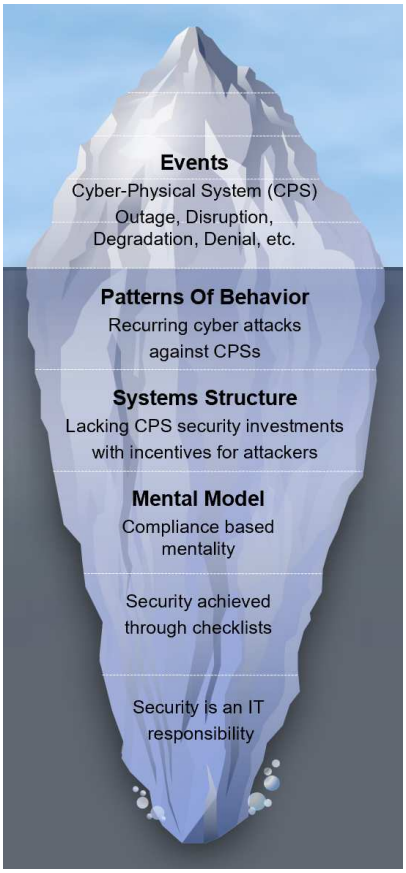

Figure 2.1: Iceberg Model For CPS Design Team Cybersecurity Challenges.

factors to CPS vulnerabilities and attacks. In most cases, the structural incentive for CPS attacks has largely gone unmatched by regulatory policy or individual company investments in security. For example, Ransomware attacks continue to occur over a broad range of industries [75].

Because of these structures, CPS design teams often do not regularly employ systems thinking nor plan for system evolution or long system lifecycles. Thus, unwanted behaviors often go undetected in early testing or fielding but emerge in delayed feedback loops (often realized as zero-day exploits). Likewise, when vulnerabilities emerge, these systems are left open and vulnerable during much of their operational lifetime because there is little incentive to fix the problem(s).

Mental Model

The foundation on which Iceberg events, patterns of behavior, and systems structure rest is the mental model for CPS design. The mental model represents the mindset and worldview of current

CPS design teams. This mindset contributes to and builds all the layers above. For example, the way a software developer views security and privacy issues directly influences how they implement the SoI. Likewise, the way the team manager views cybersecurity concerns will strongly influence the end deliverable. The details of the mental model will be explored in the following section.

A reductionist approach to solving complex problems is typically not successful, and that is certainly the case for secure CPS design. Securing individual Internet of Things (IoT) devices within a CPS with current IT security practices has proven unsuccessful, as it fails to consider the emergent properties of the system. Ironically, those who seek to exploit CPS use a holistic systems approach to determine vulnerabilities in the system based on the complexity of interactions between the system elements and the environment. The CPS design teams must employ a holistic systems approach utilizing systems thinking principles to protect against known and unknown cyber exploits. Threat and exploit tactics continuously evolve. Thus, the design must approach security proactively and invest significant effort into 'Security by Design'.

2.2.2 Application of Systems Principles to the CPS Design Team Problem

The following are four systems principles from [30] with a description of how the current CPS design does not meet the stated systems principle.

- Holism: Cybersecurity is not viewed holistically by the CPS design team. On the contrary, attackers view CPS holistically to exploit the vulnerabilities.
- Evolution: Current CPS does not accommodate system evolution, but the attacks evolve.
- Emergence: Security is an emergent property, and often vulnerabilities result from emergent behavior that results from the interactions between elements of the system and the environment.
- Feedback: Vulnerabilities emerge as unwanted system behaviors, often not apparent in early design, but emerge from feedback loops with delays.

The current state of system security efforts applies a checklist approach to patching known security vulnerabilities. This approach provides some security. However, it is insufficient to address unknown vulnerabilities induced by the complexity of interactions and a variation in operating conditions and environments. These foundational problems are the base of the iceberg and those that must be addressed to create changes to behavior patterns and events.

2.3 Architecture Models and Systems Thinking

Architecture models are one of the most effective systems thinking tools, as they help define the structure and interrelationships of systems with their environment and within their internal subsystems and components.

2.3.1 MBSE as Architecture Modeling Tool

This section is adapted from the author's book chapter in a new Systems Thinking book titled "Turbocharge Your Future in a Complex World with Systems Thinking" [30]. MBSE is a critical development in advancing Systems Engineering and System Design. Its implementation and proliferation have been widely adopted across industry and government as a best practice for complex system design and development. Systems Modeling and Architecture have evolved, with formalized languages bringing standardization to this effort. Specifically, MBSE utilizes the SysML modeling language, which is built on the Unified Modeling Language (UML) foundations.

MBSE helps you rigorously navigate development, standardization, and communication through a single unified system representation, termed a "model". This model presents different viewpoints of a system and defines a system's architecture.

Model-Based Systems Engineering is best used to help answer questions and bring organization to the complexity of system design. Modeling for the purpose of modeling is no more than generating pretty pictures. A structured language and formal methodology help address this, but systems thinking is critical to architectural modeling efforts to ensure the model accurately represents reality and provides utility to the design effort.

This section will focus specifically on MBSE architectural models, specifically system architecture's utility for systems thinking.

System architecture is defined as an abstract representation of a complex system/entity specified in terms of:

- *Structure*–how is the system organized; where do important interfaces occur?
- *Behavior*–how do individual pieces and parts interact, and how do they collaborate to produce the overall behaviors that implement the functionality required?
- *Rules*–what are the conventions (e.g., standards and requirements) that must be adhered to in order to achieve a useful and satisfactory solution?

System Architecture and MBSE helps with:

- Understanding and capturing the system's purpose and requirements
- Ensuring that the structure, behavior, and configuration of the solution deliver the needs established in the system requirements
- Applying systems thinking principles in a rigorous method when developing a solution to a complex problem
- Adding rigor, machinery, language, tools, and process (framework)

MBSE provides inherent rigor, consistency, efficiency, and fidelity to systems thinking principles [30].

MBSE with Systems Thinking

MBSE models, particularly those developed in any tools that have advanced analysis features (ie Cameo, Rhapsody, Enterprise Architect, Capella, MATLAB/Simulink, etc.), provide a robust framework for defining and analyzing system architecture.

Common System Architectural Models:

- Context Diagrams - present a structural view of a system - looking inward at subsystems and components. They also can model outward: illustrating the systems thinking 3 systems principle stating every system operates in at least 3's: the system of interest, its environment, and in conjunction with other systems. Figures 2.2, and 2.3 are examples of context diagrams showing both an internal and external look at the SoI, in this case, the Space Traffic Management System (STMS). Context diagrams demonstrate that the STMS SoI both has internal subsystems and external systems with complex interactions worth defining and understanding in the system design process.

- Use Case Diagrams: capture the key functionality of a system. They model the system actors, both operators and other systems interacting directly with the key functionality. Figure 2.4 is an example of a use case diagram for the STMS SoI. Use case diagrams are an effective tool for stakeholder alignment on the key system functionality.

- Activity Diagrams: are functional models illustrating a sequence of actions that occur to fulfill intended functionality. These are very useful for a high-level overview of functions needed for a given system use case. They assist in systems thinking by modeling condition-based relationships and the flow of activities within a system's intended behavior. Figure 2.5 is an example of an activity diagram showing the functional flow of executing the *Track Object* function of the STMS system.

Example Architectural Models This section presents samples of architectural models representing a notional space traffic management system. This system would be responsible for tracking, identifying, and predicting space objects in order to provide efficient space utilization and prevent collisions.

A *context diagram* highlights the system environment and other systems it interacts within that environment. It is an excellent and simple way to visualize the 3 systems principle. Figure 2.2 shows the external systems and actors that the system design must account for. It models the relevant environmental considerations and various user types that will interact with the system as

well. In Figure 2.2 you can see the system of interest is the Space Traffic Management System (STMS). It operates within the three systems principle with Space as its environment and earth as an enabling system, considering the gravitational pull and laws of physics that apply to orbital mechanics. This also illustrates the human principle of capturing human actors that will impact or be impacted by the system.

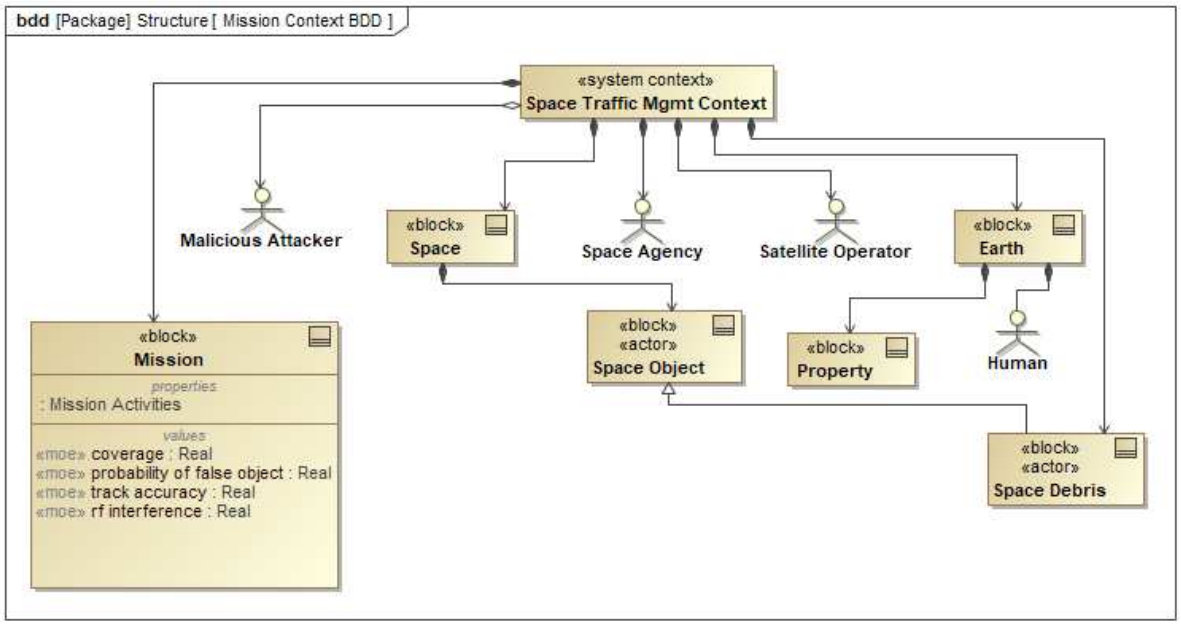

Figure 2.2: Context Diagram - External

An internal view of a context diagram, a system structural block definition diagram, as shown in Figure 2.3, illustrates the holism principle. Here, the space traffic management system as a system of interest is shown as a composition of various elements and interconnects that interact to achieve a given purpose.

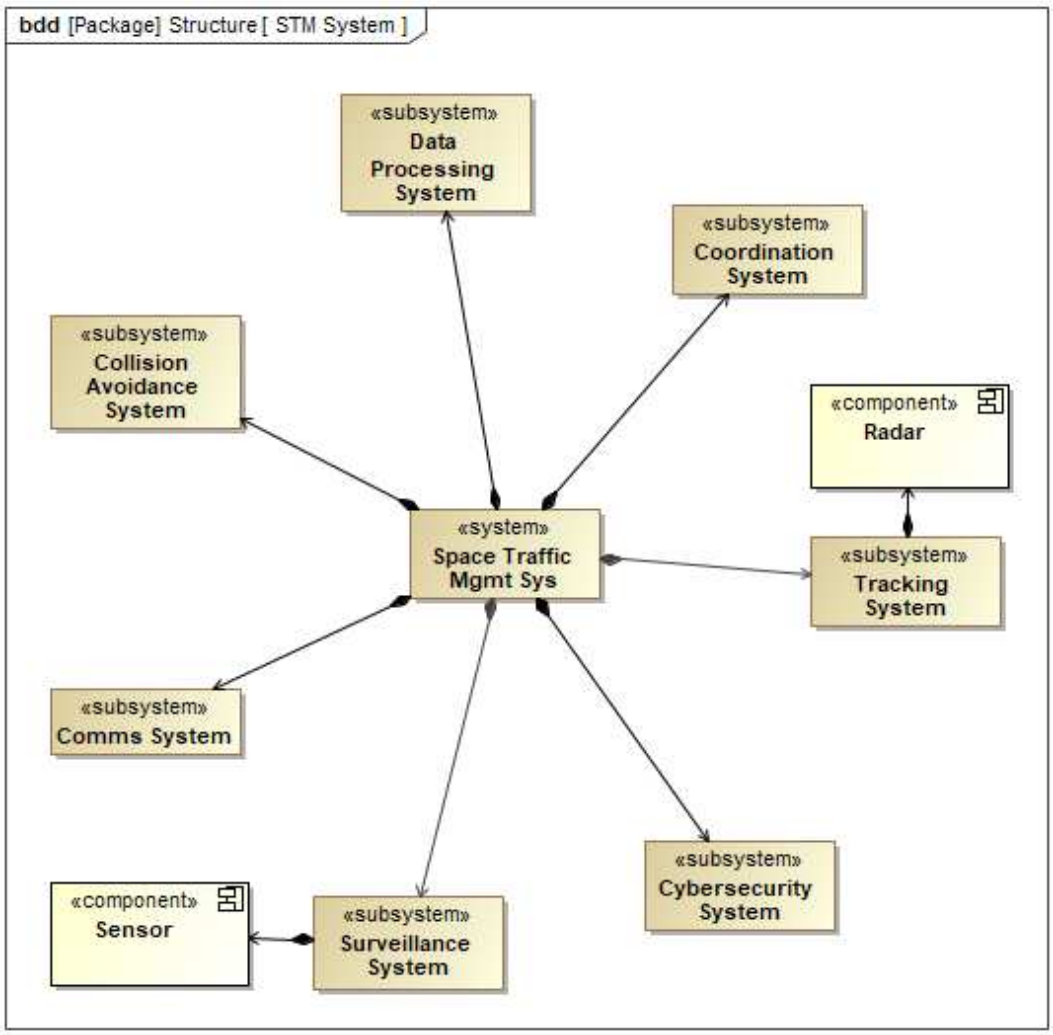

Figure 2.3: System Structure - Internal - Showing Composition

The use case diagram for the space traffic management system, shown in Figure 2.4, illustrates the actors and primary functions the system is responsible for executing. It also shows relevant external actors that interact as a part of the primary functions. Here the architecture principle is shown, the space traffic management system, having functional interconnections with space debris and also with the two types of human actors, the operator and the space agency users. It is more than just structural relationships, but also behavior relationships to achieve the key functions of Track Object, Identity Objects, and Predict Orbits.

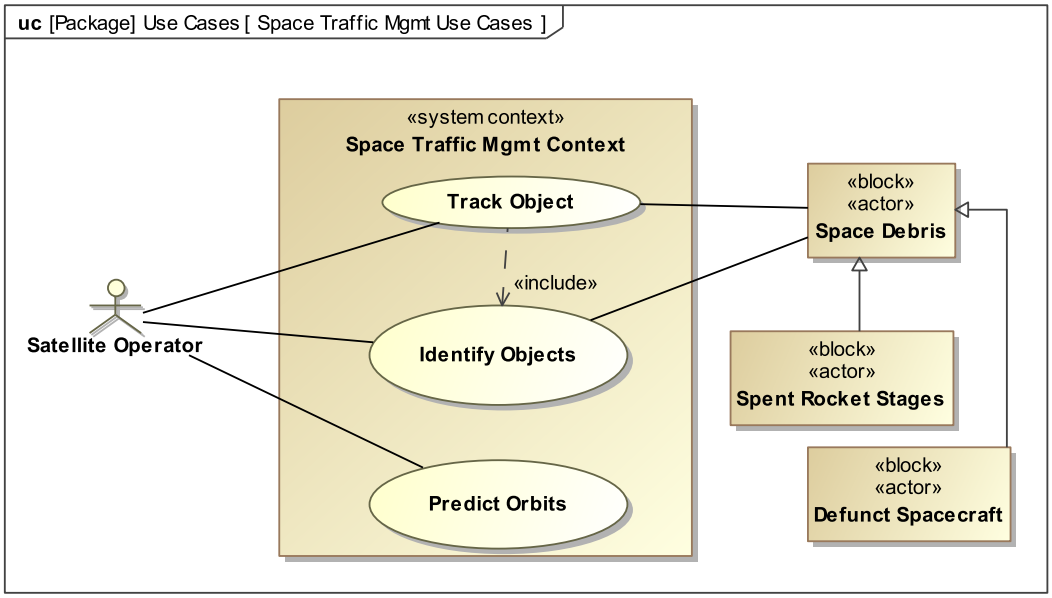

Figure 2.4: Use Case Diagram

Activity diagrams show the flow of activities required to execute a system use case, such as Tracking Objects, shown in Figure 2.4. They model the conditional logic required to execute the given activity and allocate actions to a given actor. In this case, the actions are separated between the space traffic management system itself and the user. Activity diagrams provide a functional viewpoint to utilize the emergence principle in considering the complexity of interactions impact on the system. It shows the flow of events and conditional logic of actions across various system elements and their interconnectedness to achieve a desired purpose.

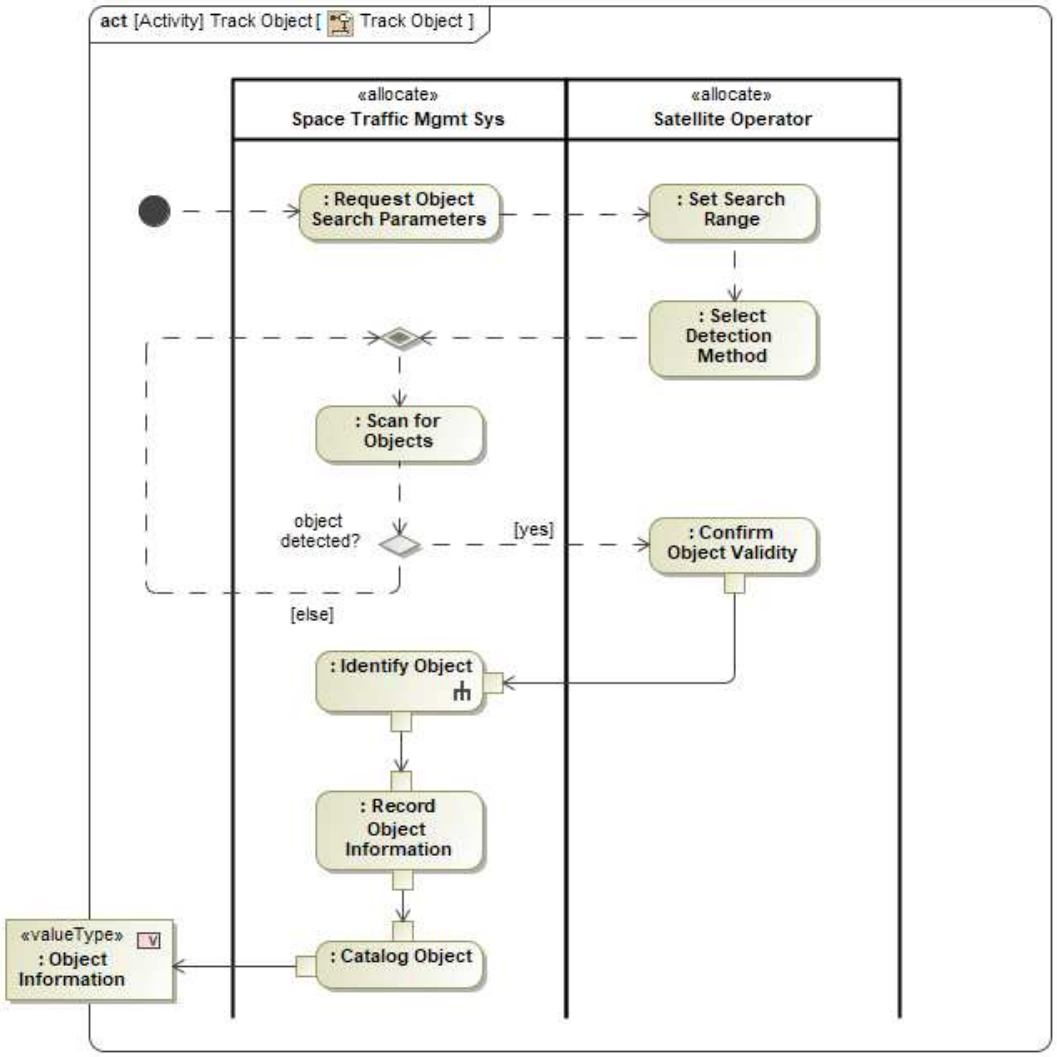

Figure 2.5: Track Object Activity Diagram

The examples above are presented to demonstrate the modeling of architectural modeling as a rigorous tool for systems thinking. They are a potential tool to help you understand and visualize the complex interactions going on in your system. They are excellent at demonstrating the interconnectedness of complex systems all around us, both man-made and natural. When used in a modern tool, they provide a digital thread across a system architecture, which is exceptionally useful for identifying dependencies within system relationships and digesting complexity into manageable chunks.

2.3.2 Using System Models to Characterize the Problem Space

Generating system models for a System-of-Interest (SoI) is an excellent technique to better characterize the complexity of interactions contributing to the undesirable behavior. Casual loop diagrams (CLD), a tool of systems thinking and system dynamics, are used along with a few SysML Diagrams popular in MBSE efforts. Our SoI, the CPS design team, aims to design CPS to deliver a system for a company or organization that meets functional requirements within the cost and schedule parameters. Figure 2.6 is an adapted SysML Block Definition Diagram (BDD) that depicts the System of Interest, Enabling Systems, and the Environment. It also describes the system elements and the interconnections. CPS design teams are composed of the discipline engineers as pictured and interact with other functional areas within the organization.

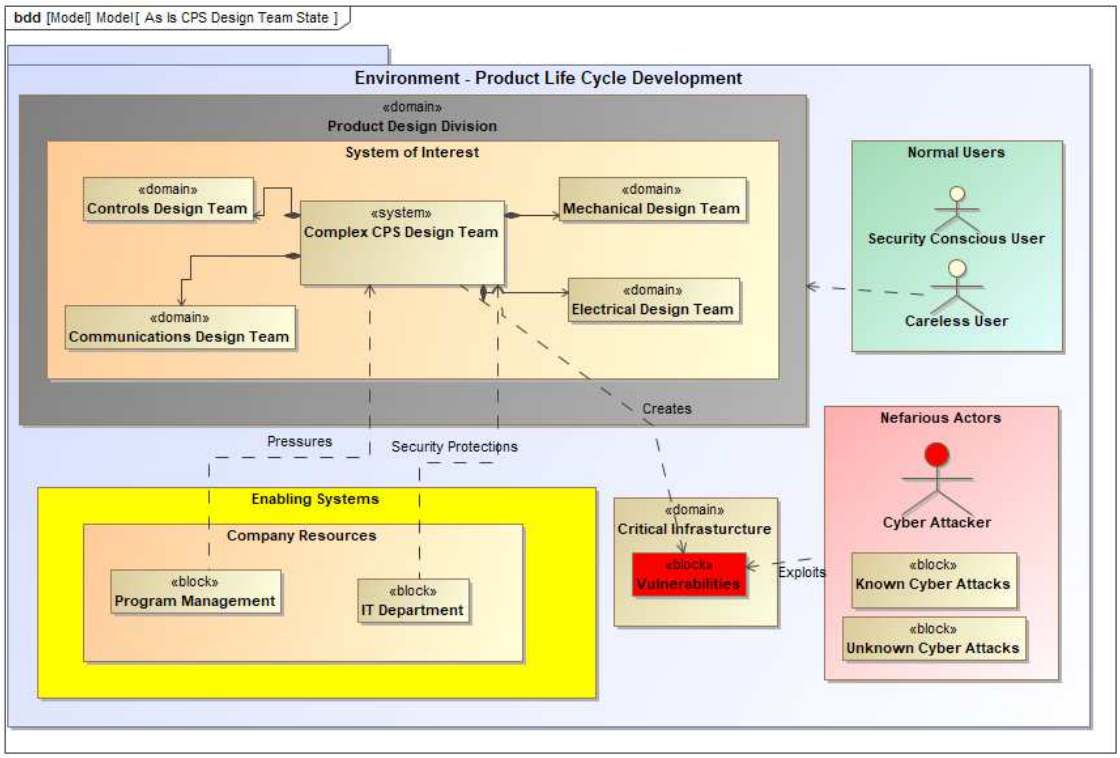

Figure 2.6: SysML Block Definition Diagram (BDD) Modeling CPS Current State

Causal Loop Diagrams (CLD), shown in Figures 2.7, 2.9, and 2.10, are a modeling technique used to understand the variables impacting a complex system along with their impact as either re-

inforcing (R) or balancing (B) behaviors. Figure 2.7 represents the impact of limited cybersecurity awareness of CPS design team, the evolution of attackers, and the impact of quick IT security. The R loop indicates a reinforcing effect.

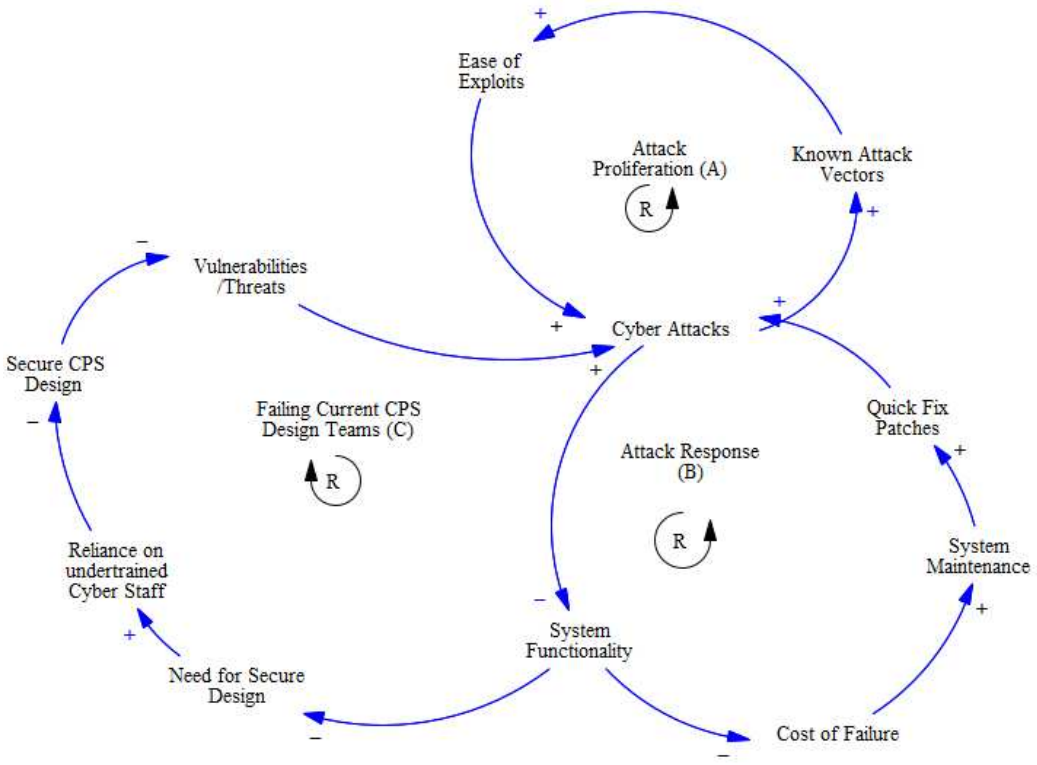

Figure 2.7: Causal Loop Diagram Showing Complexity of CPS Design Team and CPS Security

The Attack Proliferation (A) loop represents the reinforcing effect of cyberattacks on CPS. Attackers gain knowledge from cyberattacks and evolve with new techniques to exploit the vulnerabilities, causing increased cyberattacks. The Attack Response (B) loop represents the reinforcing effect of corporate response to cyberattacks. Cyberattacks have a negative effect on the system's functionality. A broken system increases the cost of failure, as its normal functionality needs to be restored. The cost of failures affect the system's maintenance cost and schedule. Due to increased cost and limited Operational Technology (OT) security awareness, the corporation often creates quick IT security patches as a response to the cyberattacks on the CPS. Quick IT security patches

introduce new vulnerabilities and thus make the CPS more prone to increased cyberattacks. The failing current CPS design team (C) loop represents the reinforcing effect of the failure of CPS design teams to address cyberattacks. The need for secure design is overlooked, and the organization depends on the CPS design team, which is not well-trained in cybersecurity. Generally, CPS design teams focus on system concepts as parts instead of holistically viewing the entire system over its life cycle. For instance, the control software team focuses on the design of software using software modeling tools. The electrical team focuses on electrical design via electrical design tools such as AutoCAD electrical drawings. The mechanical design team focuses on the mechanical aspects of the system through mechanical design modeling tools. Thus, each discipline focuses on its own sub-system without taking cybersecurity into consideration. This creates a gap in the holistic design of the entire system with little or no cybersecurity consideration, creating security vulnerabilities and threats in the CPS. Undetected vulnerabilities and threats are eventually exploited by attackers, continuing these negative reinforcing loops of increasing cyberattacks.

2.4 Using Systems Thinking Models

This section details a proposed solution to the problem of poor CPS design team composition through updated models and a discussion of unknowns. This illustration demonstrates the utility of Systems Thinking and MBSE to evaluate potential solutions to complex organizational problems.

Figure 2.8 is an adapted SysML Block Definition Diagram showing the incorporation of cybersecurity conscious engineers and the inclusion of the specialty discipline of system security engineering within the CPS design team. In addition, this analysismodels a healthier relationship between the corporation's enabling systems and the design team, SoI. Program management empowers the CPS team, and the corporation's IT staff, and infrastructure provide resources and support to the CPS team. This solution could be achieved either through training of current staff, the more likely solution, or through hiring efforts of new design engineers with a skill set that includes some cyber design appreciation. Our ideal CPS design team treats cybersecurity on equal

footing with traditional system functionality requirements. The design minimizes vulnerabilities, making it more difficult to exploit by cyberattack.

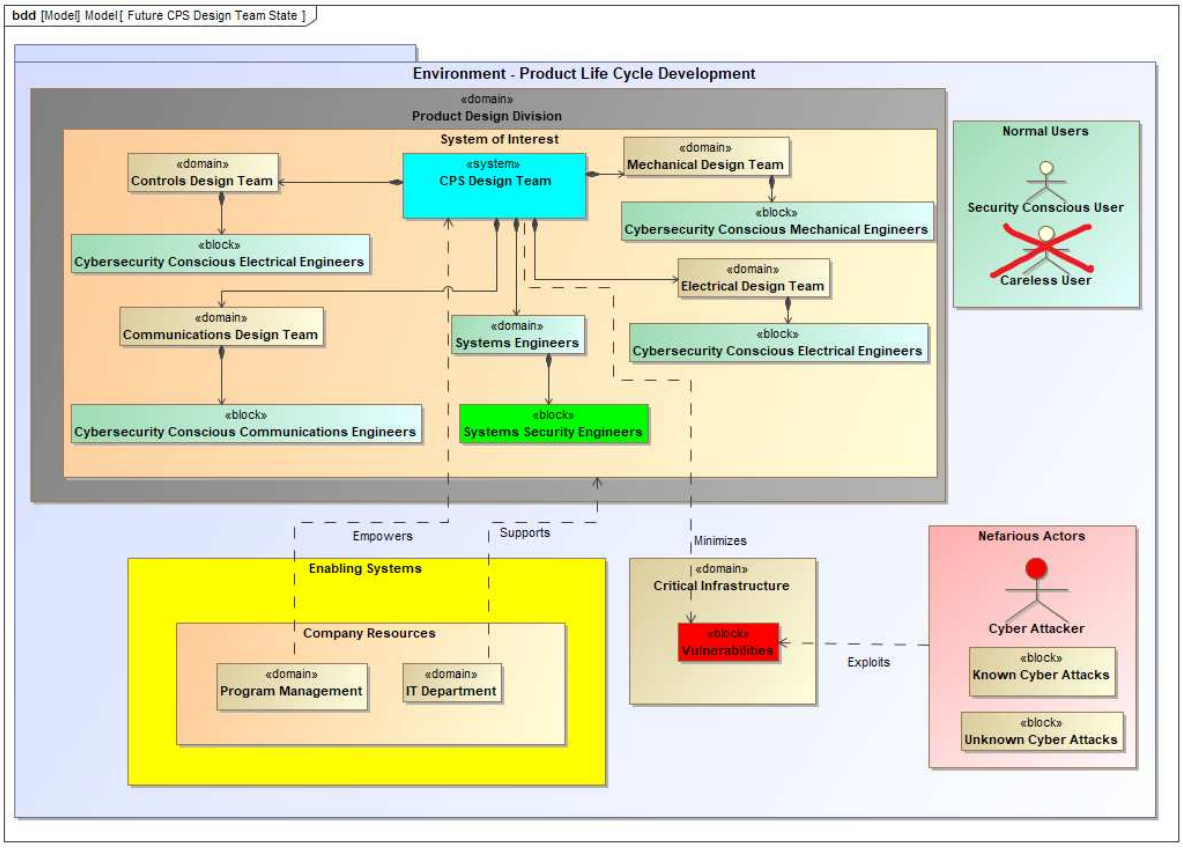

Figure 2.8: SysML BDD Of Proposed Design Team Solution

Our improved CPS design team now implements an enhanced feedback loop leveraging mentoring, feedback, and lessons learned to apply cybersecurity to the system design and evaluate changes impact on cybersecurity with expert input.

Figure 2.9 represents the proposed solution to change the mental model that improves the cybersecurity awareness of the CPS design team. The Cybersecurity Awareness (CA) loop represents the reinforcing effect of training the CPS design team. The CPS design team initially gets trained on cybersecurity by external system security experts. The cybersecurity awareness of the CPS design team improves through knowledge sharing and expert mentoring. The Risk Analysis (RA) loop represents the reinforcing effect of security evaluation and threat modeling. The

Cybersecurity Conscious CPS Design Team performs security risk assessments to evaluate cybersecurity requirements. Then, it verifies the cybersecurity assessments through threat modeling and re-evaluates the security requirements based on the threat modeling results. After security evaluation and threat modeling, the Cybersecurity Conscious CPS Design Team incorporates security by design by implementing security techniques. The entire process of a security evaluation, threat modeling, and security by design improve the mental model of the Cybersecurity Conscious CPS Design Team, extending its knowledge and experience of cybersecurity.

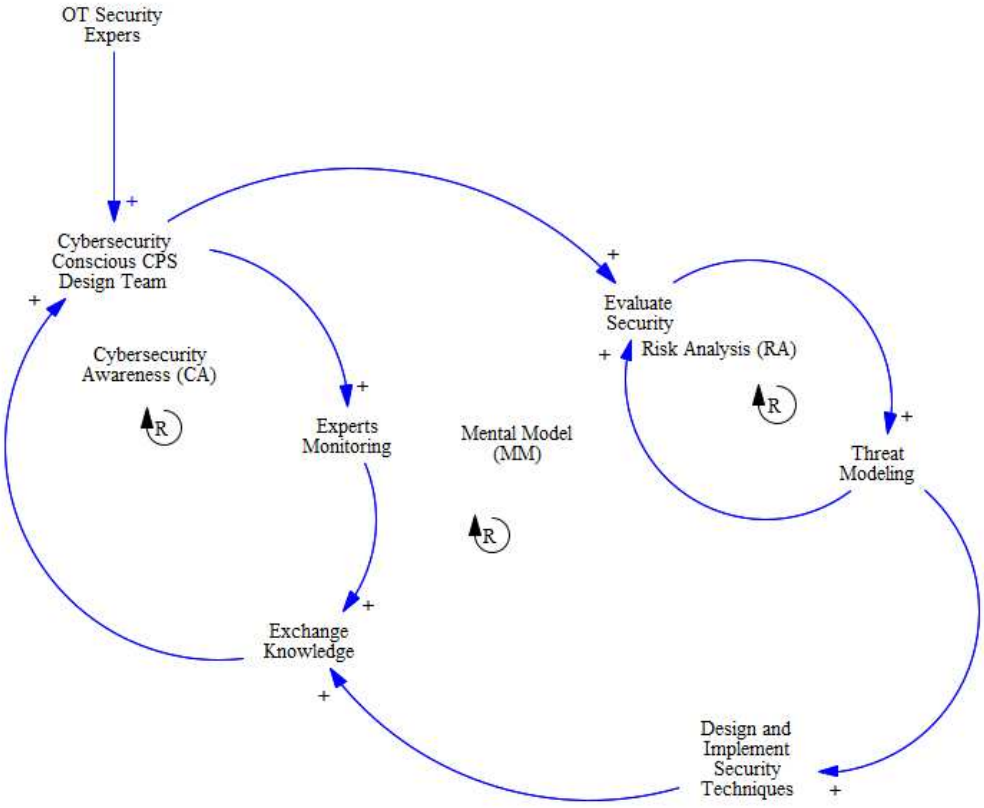

Figure 2.9: Causal Loops For Secured CPS Design Through Cybersecurity Conscious CPS Design Team

Figure 2.10 represents the integration of the proposed solution to the complexity of the problem, addressing the issues discussed in Section 2.3.2. The proposed solution helps develop the

cybersecurity conscious design team that can now incorporate security in the early design iterations.

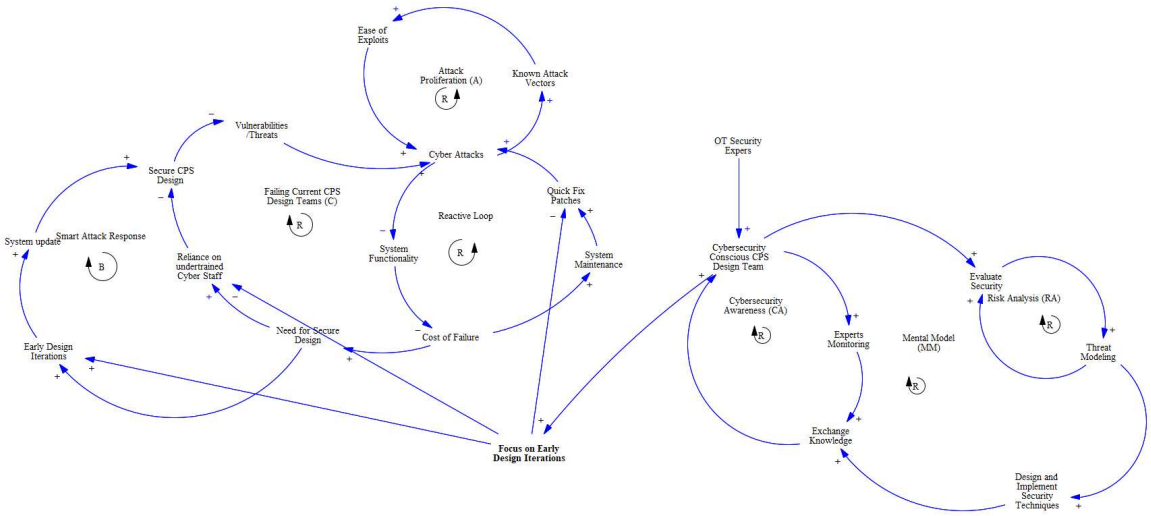

Figure 2.10: Integration of solution with the problem CLD

Focusing on early design iterations reduces the possibility of IT quick fixes in the reactive loop as the possible security threats are addressed in the design phase. In turn, this reduces the possibility of introducing new vulnerabilities, thus reducing the possibility of cyberattacks. The security through early design iterations reduces the dependency on untrained cyber staff in the Failing Current CPS Design Loop (C). It also improves the security system design, which reduces the possible vulnerabilities/threats in the system, thus reducing the possible cyberattacks. The proposed solution facilitates smart attack response in the case of cyberattacks, as represented in the Smart Attack Response loop. The cybersecurity conscious design team re-evaluates the security of the CPS under attack. It performs the system update to address the attacks instead of installing quick IT security patches. The security evaluation and system update process improves the secure system design, reducing possible vulnerabilities and cyberattacks. Thus, developing a cyber-conscious CPS design can improve system design by incorporating security in the early phases of the system development, making the system more resilient to cyberattacks.

2.4.1 Discovery of Unknowns

This section evaluates CPS design teams along with the proposed solution using the iceberg model elements, Figure 2.1, where events, patterns, structure, and the mental model are assessed. Potential solutions to complex problems can introduce certain unwanted emergent behaviors within a complex system, as shown in Figure 2.9. The following bulleted list illustrates the value of the iceberg model in analyzing the current state, unwanted emergent behaviors due to the proposed solution, and actions to address those behaviors.

- **Events**
  - **Current culture/emergent behaviors due to proposed actions**:
    1. The Design Team is focused on CPS design and relies on IT for security.
    2. The design team has little to no knowledge of CPS vulnerabilities and possible threats.
  - **Proposed actions to address unwanted emergent behaviors**:
    1. Hire Systems Security Engineering (SSE) and Operational Technology (OT) experts to train the CPS design team to become cyber-conscious and implement security by design.
- **Patterns of behaviors or trends over the time**
  - **Current culture/emergent behaviors due to proposed actions**:
    1. Resistance to change and reluctance to learn new concepts (outside their domain) and new tools design engineers leaving the organization.
  - **Proposed actions to address unwanted emergent behaviors**:
    1. The organization should inform its teams of recent cyberattacks and their consequences for CPSs.
    2. A visual framework may assist in helping the design team understand their significance in securing CPS design, with how and where they fit.

- **Structure**
  - **Current culture/emergent behaviors due to proposed actions**:
    1. The CPS design team is an expert in the design and development of CPS, but not in security. IT support for security enabling systems is limited and the environment is driving pressure focused on cost and schedule.
  - **Proposed actions to address unwanted emergent behaviors**:
    1. Train the CPS design team on cybersecurity, detection of vulnerabilities, threat modeling, and risk analysis.
    2. Promote knowledge sharing through structured mentoring programs, webinars, and seminars on real-world attacks and consequences.
- **Mental Model**
  - **Current culture/emergent behaviors due to proposed actions**:
    1. The CPS design team is not valued for its expertise.
    2. Inefficient CPS development within cost and schedule.
    3. Security is outside the area of responsibility.
  - **Proposed actions to address unwanted emergent behaviors**:
    1. The gradual transition of the current mindset to adopt security by design.
    2. Implement stakeholders' encouragement programs. Conduct surveys to evaluate all stakeholders' feedback.

The injection of cyber-conscious engineers into the design team with the addition of systems security engineers should provide the right skill sets to bring about 'Security by Design' for CPS. This solution addresses the current problem of CPS being designed without regard for the complexities of interactions and emergence within the current design. Cyber appreciation and these specific skill sets should not introduce additional adverse side effects. However, it is likely the

additional design considerations for cybersecurity may increase the cost and schedule for their implementation in design. If done properly, the cybersecurity requirements will be generated early in the system life cycle and will afford design trade-off decisions alongside traditional functional and safety design considerations. In this optimal implementation, additional cost and schedule concerns should be mitigated.

Unfortunately, a negative emergent property is likely that not all traditional design engineers will fully embrace systems thinking and cybersecurity in their design methodology. It is challenging for individuals to realign their mental model and escape their current habit patterns. Changing the mental model requires deliberate education, training, and reflection, as detailed throughout this section of work. Through the types of actions proposed above, one can spark a change in the mental model of their CPS design teams.

Next, this work will illustrate how Systems Thinking models can be used to evaluate the impacts of potential solutions. In the CPS design team case, the proposed changes to the CPS design team along with the mitigation of the unknowns discussed here should instill teams that can truly achieve 'Security by Design'.

2.5 Additional Considerations for CPS Design Teams

This section presents considerations for improvements to CPS design teams to more effectively field secure CPS systems by embracing 'Security by Design'. The recommendations presented in this section identify potential ways to change the problematic mental models assessed as the root cause of the CPS cybersecurity problem space characterized in the Iceberg Model described in Section 2.2.1. These considerations are discussed using an Electrified Aircraft concept as shown in Figure 2.11.

Figure 2.11: Operational Concept Diagram for Notional Electrified Aircraft. This concept is derived from the innovative work of Joby Aviation [2].

The current state of system security efforts too often applies a checklist approach to patching known security vulnerabilities and is often relegated to IT personnel and executed too late in the system design and life cycle. This approach provides some security. However, it is insufficient to address unknown vulnerabilities induced by the complexity of interactions and a variation in operating conditions and environments.

2.5.1 CPS Design Team Structure

Figure 2.12 is an adapted SysML Block Definition Diagram (BDD) that depicts the CPS design team for an electrified aircraft system with enabling systems and its environment. It describes the system elements and the interconnections. CPS design teams are composed of the discipline engineers interacting with other functional areas within the organization. The CPS design team aims to design safe and secure CPSs for a company that meets functional requirements within the cost and schedule parameters.

Figure 2.12 illustrates a proposed solution to the problem of poor CPS design team composition through an injection of cybersecurity and systems thinking conscious design engineers with additional security specialization knowledge being added to the design team. In addition, a health-

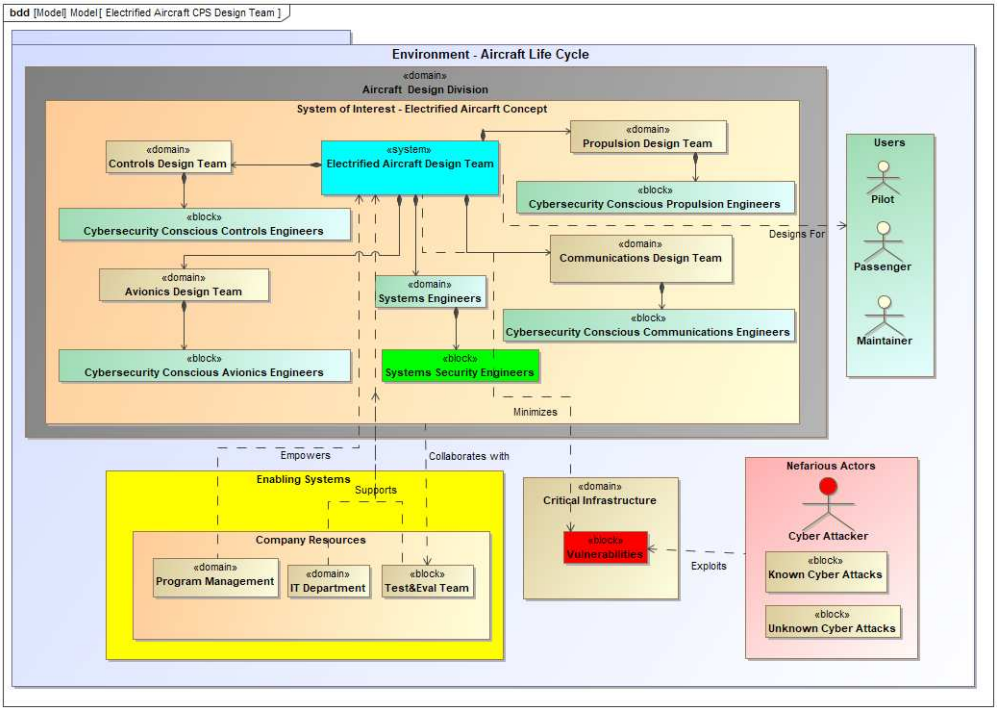

Figure 2.12: Proposed Human Design Team Structure for an Electrified Aircraft Concept.

ier relationship is modeled between the corporation's enabling systems and an electrified aircraft design team, SoI. Program management empowers the CPS team, and the corporation's IT staff, test and evaluation team, and additional infrastructure provide resources and support to the CPS team. The CPS team collaborates with these resources and ensures the design captures diverse stakeholder needs including those of the users as shown in the model. This improved CPS design team could be achieved either through training of current staff, the more likely solution, or through hiring efforts of new design engineers with a skill set that includes some cybersecurity design appreciation and skill.

2.5.2 CPS Design Team Fundamentals

Embracing 'Security By Design' requires a holistic approach with focused effort across the entire system development lifecycle, especially during early conceptual analysis activities. This effort starts with understanding the SoI's intended purpose or mission. In this example, the notional

electrified aircraft is intended to provide safe and secure passenger transport. Further, the team must be able to understand and apply pertinent cyber threat reports.

Shown in Figure 2.13, an environment in which an electrified aircraft might expect to operate is presented. The environment includes a number of actors, entities, services, and existing infrastructure considerations. In contrast to conventional aircraft transportation (which is well understood), new and emerging concepts like vertical takeoff and landing in urban environments require a renewed focus on understanding essential functions and dependencies – both internal and external dependencies. For example, the CPS design team should seek to more fully understand key stakeholder needs and associated requirements in addition to meeting regulatory safety compliance. Likewise, it greatly benefits the CPS design team to identify – as early as possible – essential functionality, information flows, and control loops [57] before they are realized in software, hardware, and/or firmware. Lastly, external cyber dependencies play a critical part in designing safe and secure systems as each communications port provides access to potential cyber attackers.

Moreover, it is worth noting that because of rapidly changing cyber capabilities and recent automation advancements, it is absolutely critical to consider Human-Machine trust and teaming issues. For example, vehicles (ground or airborne) are no longer mechanically controlled systems. Vertical takeoff/landing aircraft are likely to be realized with complex feedback loops incorporating dozens of computers and processors all "controlled" by a pilot. Thus, system design and development must be able to accommodate, anticipate, withstand, recover from, and adapt to unexpected inputs, cyber attacks, and service disruptions in addition to other expected adverse conditions. While sounding insurmountable, this type of multidisciplinary concept analysis work is regularly accomplished through approaches such as Systems-Theoretic Process Analysis [58–61].

2.5.3 CPS Design Team Core Knowledge and Experiences

There is a shortage of qualified and talented systems and security engineering personnel with specialties in system security. Thus, the required workforce must be created. This section briefly

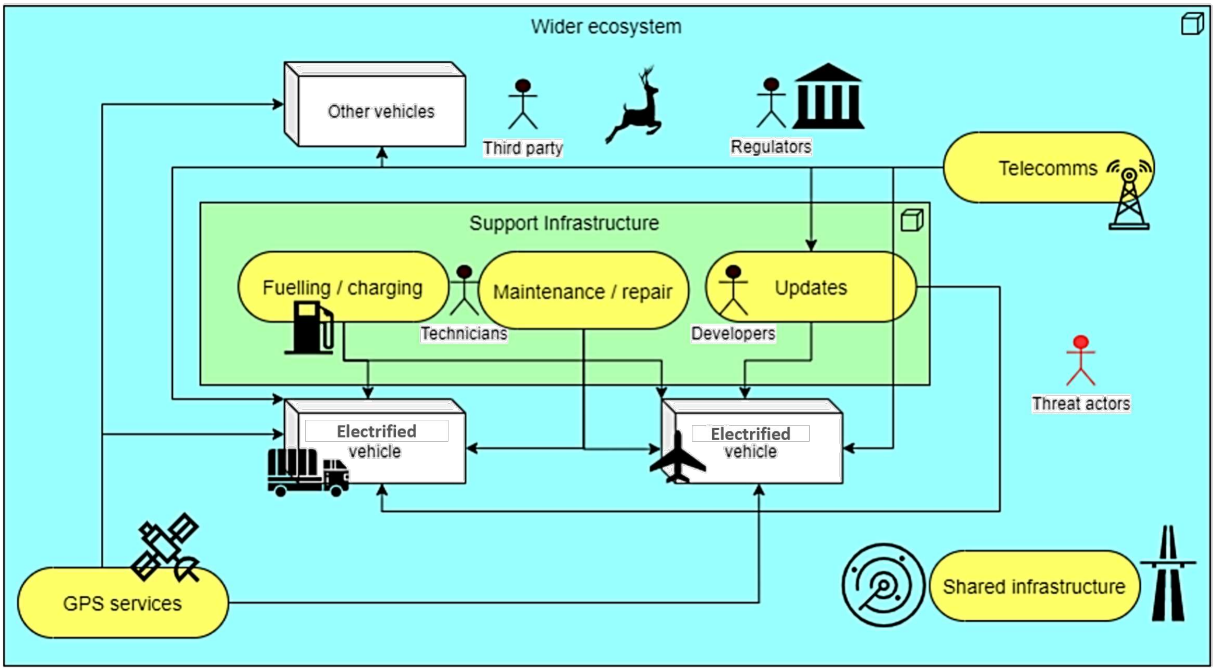

Figure 2.13: Electrified Vehicle Operational Environment & Threat Diagram adapted from [3].

identifies and comments on the wide breadth of experiences and prerequisite knowledge to develop safe and secure CPSs.

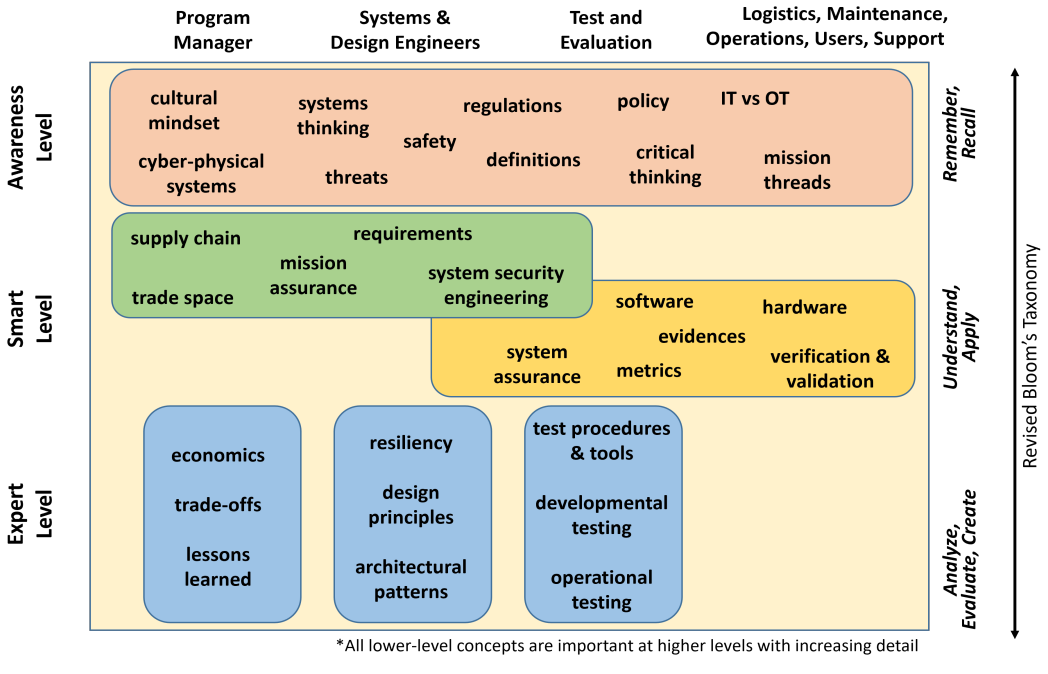

Figure 2.14: Experience and Knowledge Required for Safe and Secure CPS Design Teams.

Shown in Figure 2.14, CPS design team required experiences and knowledge are mapped across the revised Bloom's taxonomy [76]. This mapping was accomplished as part of an effort to train some 50,000 program managers, engineers, and other personnel in systems security engineering. These knowledge areas should be applied to teams as a whole and individuals at multiple levels of detail:

- (i) Awareness - Remember and Recall
- (ii) Smart - Understand and Apply
- (iii) Expert - Analyze, Evaluate and Create

New engineers are expected to first learn and understand these concepts, while journeymen are expected to effectively apply these standards, policies, and frameworks. As personnel gain experience, they should participate in and contribute towards higher level strategic planning and refinement. Unfortunately, there is a national shortage of people with these experiences and key knowledge; thus, a breakdown is provided to help articulate the need to train, educate, and develop personnel with the necessary knowledge and skills to successfully apply cyber-physical security concepts to technical design efforts.

2.5.4 CPS Design Team Necessary Skills and Abilities

In addition to pointing out essential knowledge, Figure 2.14 is also helpful for identifying necessary skills and abilities team members must have in order to meet CPS's security needs. In addition to considering security requirements across the CPS's entire lifecycle (the Concept, Development, Operation, Maintenance, and Retirement phases), this work elaborates on the most interconnected systems security engineering processes and discusses them in the context of the Electrified Aircraft Concept [77]:

- Business/Mission Analysis - Consider the stakeholder's security needs across the system's set of operational concepts (transporting people or goods), operational environments (urban, suburban, or rural), operators (experienced or novice pilots).

- System Requirements Definition - Write “SMART” (Specific, Measurable, Achievable, Realistic, and Timely) security requirements like ensuring aircraft digital communications are encrypted and/or authenticated whenever feasible.

- Architecture Definition - Assess competing architectures (both functional and physical architectures) and prioritize potential solutions with consideration for long-term feasibility. This activity should be done not only for the internal SoI’s architecture but with consideration for the existing infrastructure the aircraft must utilize and depends upon.

- Design Definition - While defining the SoI, it is important for CPS design teams to leverage best practices captured in applicable CPS cybersecurity frameworks to ensure ‘Security by Design’ for personnel, processes, and technological solutions are indeed designed in.

- Implementation/Integration - Historically, most security features are bypassed due to poor implementations rather than broken. Careful implementation and integration is necessary to deliver defensible systems.

- Verification - Performing low level test and evaluation of the Electrified Aircraft includes performing input fuzzing on each port and data input, hardware reliability analysis, and holistic cyber red-teaming to look for non-obvious vulnerabilities.

- Validation - Assessing the cybersecurity posture of the fielded system in a realistic operational environment includes performing and document cybersecurity activities in appropriate artifacts for inclusion in system security and risk management processes.

Skills for the CPS design team are important, but so are the team’s abilities (or collective attributes). For example, the ability to act with integrity and humility is essential for establishing trust within the team. Each member of the design team must be free to raise issues and bring challenges to the team’s attention, even if the consequence negatively impacts the budget or delivery schedule. Likewise, humility acknowledges that the team may not have all the answers, and may need to

seek a different approach or opinion to understand and solve a specific issue (whether technical or human).

There is an adage that says, "Anyone can design a security system to keep themselves out." However, a good CPS team needs to design a system to keep everyone out and since security as a system property is dynamic and emergent, the job of the team is never really over. The practical consequence of this recognition is that the design team itself must undergo continuous improvement, it must learn and adapt. The CPS team will inherently shift focus from concept to preliminary design, to detailed design, production, utilization, and retirement to cover the complete system lifecycle.

Lastly, the Knowledge Skills and Abilities (KSAs) advocated herein are aspirational in nature; most organizations will not be able to realize them. In those cases, leadership must recognize the importance of bringing in carefully selected support services or investing in education for the existing workforce. Incentive structures should be built to reinforce secure design pursuits and discourage apathy toward cybersecurity. Likewise, cross-organizational awareness should be emphasized, and 'stove pipe' development structures should be eliminated whenever possible. Stove piping engineers fails to achieve holistic systems thinking approaches.

2.5.5 The CPS Design Team Developmental Lifecycle

The lifecycle of the SoI, the Electrified Aircraft CPS Design Team, is shown in Figure 2.15. This figure depicts the current state of design team representation in some percentage of effort across a traditional system life cycle. The life cycle shows a few of the problems in the current SoI. There is minimal system engineering involvement, and IT is solely responsible for security, yet brought in after most of the design, and Testing and Evaluation (T&E) is finished.

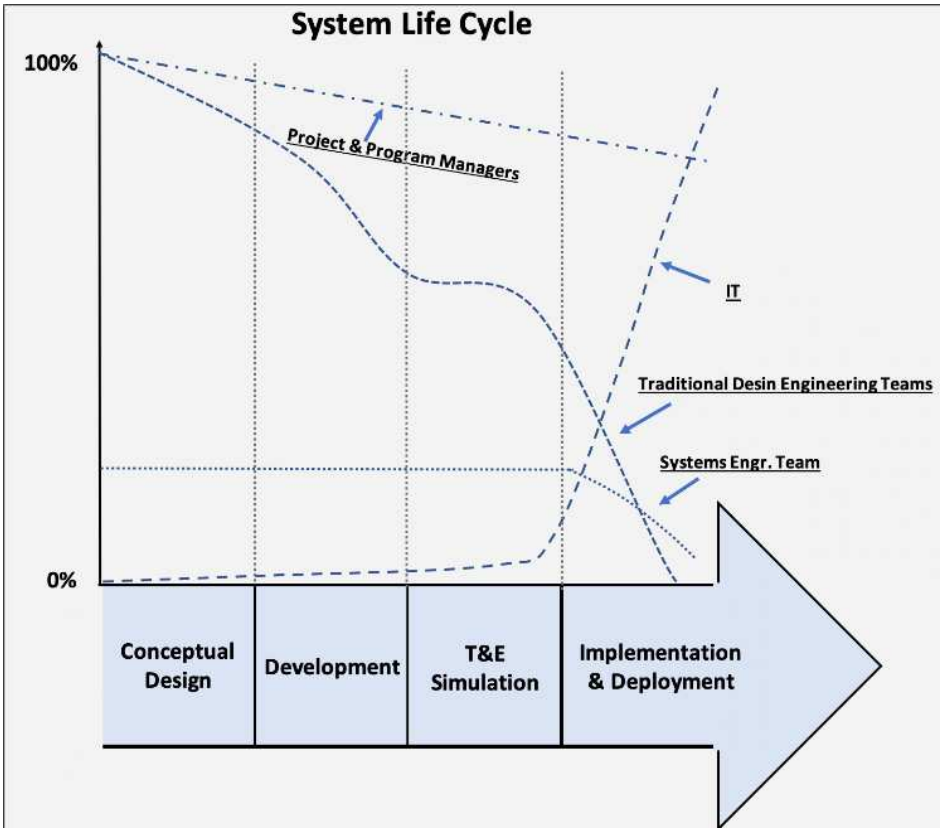

Figure 2.15: Typical CPS Life Cycle.

Our ideal CPS design team treats cybersecurity on equal footing with traditional system functionality requirements. This is achieved through earlier and additional Systems Engineering, OT, and Systems Security involvement. Earlier involvement of these specializations allows for earlier design iteration to address systems and security considerations. This should minimize vulnerabilities, making it more difficult to exploit by cyberattacks. An implementation of the desired state system life cycle model is shown in Figure 2.16 infused with systems and security engineering as illustrated in Figure 2.12. For a CPS design team to achieve the stated objective of 'Security by Design', Operation Technology (OT) experts and cyber-conscious engineers should be included in the system development process and a greater amount of System Engineering, on the whole, should be included. The updated percentages of effort across the life cycle in Figure 2.15 demonstrates this desired balance of team disciplines and the optimal timing for their involvement in the system development efforts.

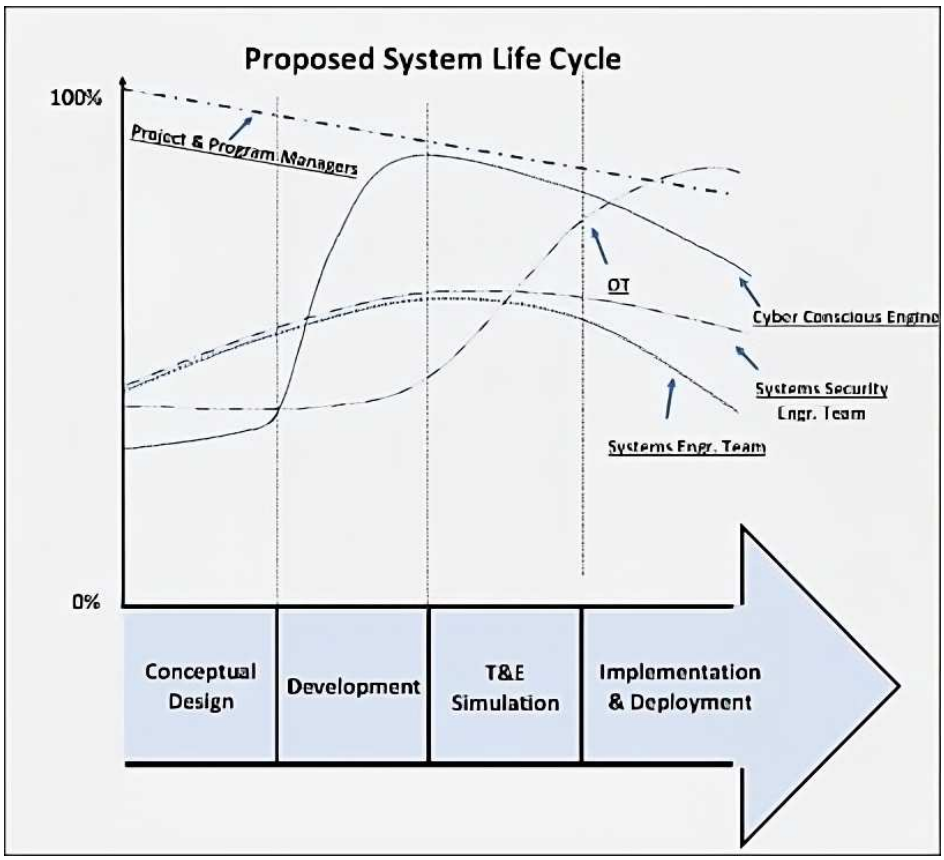

Figure 2.16: Life Cycle Model for a Security Smart CPS Design Team.

2.6 Chapter Conclusion

This chapter answers research question one by providing novel recommendations for the improvement of CPS design teams to field secure systems. This chapter illustrates the utility of Systems Thinking and MBSE in solving complex organizational problems. Using Systems Thinking principles, iceberg models, block definition diagrams, and causal loop diagrams, an analysis of current CPS design teams is presented systematically identifying the underlying causes of weak security design. The analysis extends beyond system design itself, leveraging systems thinking to uncover the problems in the organizational structure and the mental models that result in system loss events and the pattern of weak security design. CLD's and a SysML BDD are used to model the negatively reinforcing current behaviors and propose better models showing the changed behavior and its impact on security. A systematic analysis of the potential issues in execution is

provided, along with suggestions to reduce the implementation risks of a proposed solution. Many current CPS design teams lack a holistic systems approach specifically as applied to the challenge of securing the system, while ironically those that seek to exploit the system use systems thinking to engineer the complexity of interactions to their benefit in attacking the system.

Several important issues such as the CPS design team construct, prioritization of effort, essential knowledge, necessary skills and team attributes, and design team composition over a system life cycle are discussed presenting recommendations on their implementation and examples as applied to an electrified aircraft concept. CPS design teams must embrace a holistic systems approach infused with systems security engineering to design and field secure systems.

While this work does not present a case study of a solution as implemented in an actual CPS System Design Team, it provides the necessary foundational modeling to justify proposed changes to an organization seeking improvement. Systems Modeling, specifically the CLD's in this work, provides a low-cost method of demonstrating the value of investing in Systems Thinking and Systems Security Engineering training and new hires required prior to investing significant time and monetary resources in its implementation.

The investigation and research in answering question one resulted in the conclusion that training people is effective, but not a universal solution to improving CPS design with respect to security. One clear area where training and hiring proper security expertise is not widely adoptable is for small companies or small teams that cannot afford the training or dedicated personnel hires. The next chapter will introduce research question two and a methodology developed for universal application to address the challenge of 'Security by Design'.

Chapter 3

EGRESS: Eliciting Goals for Requirement Engineering of Secure Systems

3.1 Security Goal Elicitation Method Introduction

This section introduces a new method for security goal elicitation, named EGRESS: Eliciting Goals for Requirement Engineering of Secure Systems, shown in Figure 3.1. This method incorporates best practices from the works surveyed in Chapter 1. In Chapter 4 the EGRESS method is applied to various systems of interest, demonstrating its utility.

The EGRESS method is intended to be used in the conceptual design phase of complex cyber-physical system design. It is important to note that this method is intended to be used ***before*** a preliminary architecture is available for the system of interest (SoI) as shown in Figure 3.2. Methods and techniques exist for completing security analysis on systems with some architecture definition, as outlined in the literature review. However, a gap was identified in methods for eliciting early system security requirements, or goals, when architectural features were not available. This method addresses the INCOSE Vision 2035 need for security to be as foundational of a requirement in system design as functionality and safety. EGRESS is a best practice based approach for eliciting security goals in early complex CPS design.

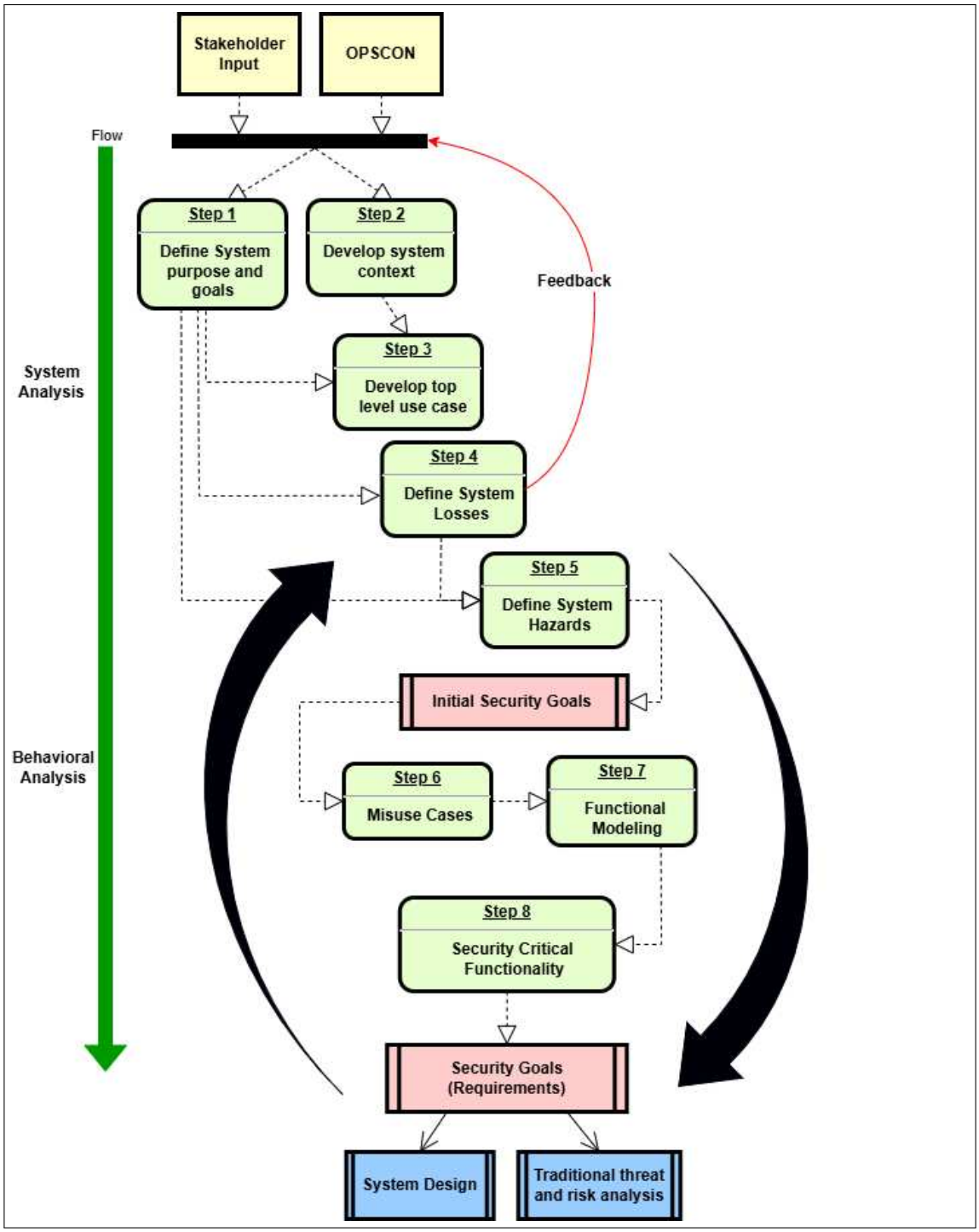

Figure 3.1: Eliciting Goals for Requirement Engineering of Secure Systems (EGRESS) Method

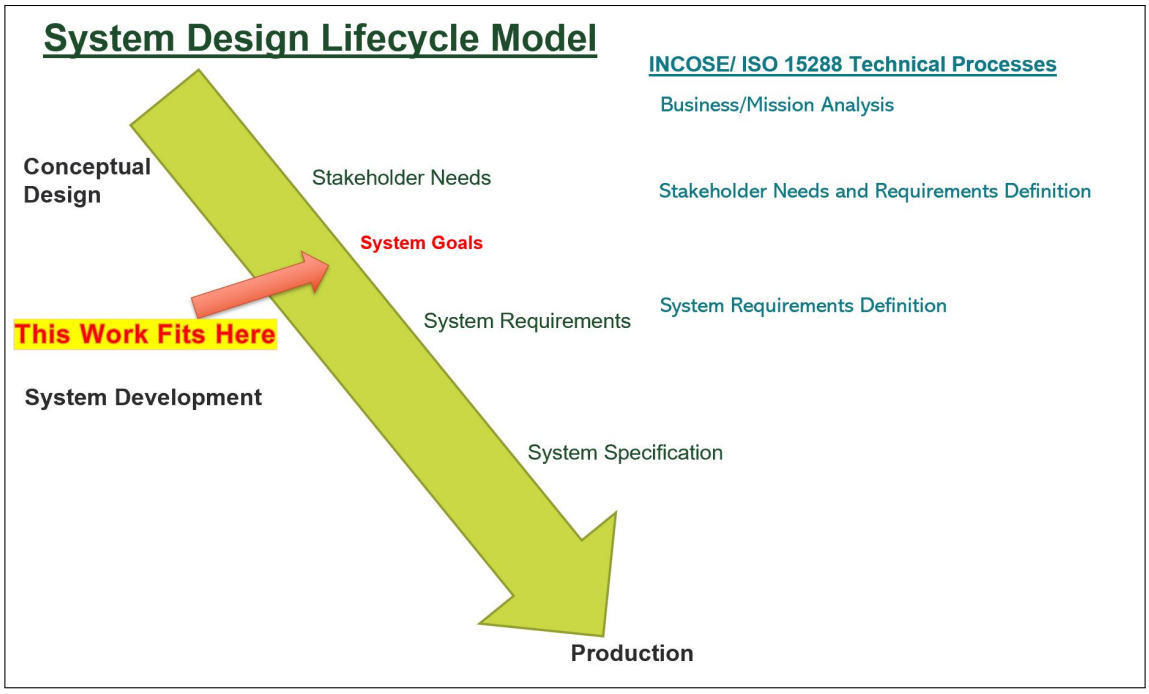

Figure 3.2: Location in System Design Process

3.1.1 Inputs Required to EGRESS

The minimum input needed to begin the EGRESS method is stakeholder input. This can either be in the form of a stakeholder working group or with a well-formed Concept of Operations (OPSCON/CONOPS) that details the intended system use and mission. Stakeholder participation is highly valuable and encouraged in conducting EGRESS. Stakeholders are any party with an interest in the SoI and includes (but is not limited to): system owners, users, program management, engineering discipline leads, and SEs.

Stakeholder Alignment is a key need for initiating a successful system design. This objective is highlighted in the INCOSE Future of Systems Engineering, FuSE, Objectives list [78]. EGRESS can help build the common security vision and knowledge among the stakeholders addressing this FuSE objective need. It is not immune from the same challenges in traditional system design for security. The misalignment of stakeholder incentives is particularly challenging for system security [79]. Stakeholders are able to comprehend and quantify direct impacts from insufficient performance or the cost of safety and performance based shortcomings. However, security is often seen

as unnecessary or interfering with performance due to added complexity or 'inconvenience'. Also, the cost of security related shortcomings in system design is not easily quantifiable, which further weakens secure design considerations. Together, these challenges lead to weak or omitted security design in many CPS's. EGRESS provides a mission-centric analysis, eliciting security goals focused on ensuring the mission capabilities and preventing system-level unacceptable losses.

EGRESS combines structural analysis along with behavioral modeling and analysis in an iterative approach to elicit security focused goals to inform system design.

This section introduces the steps of EGRESS as a whole, independent of implementation methods and tools. Section 4 details each step and demonstrates examples of executing the method on various SoI's.

3.1.2 System Analysis

The first steps of the EGRESS method begin with a system analysis focused on understanding the purpose and goal along with the system context. The first step is to define the system's purpose and goals: the what, why, and how. In step two, the system context captures relevant portions of the operating environment and supporting systems. It is important to capture the system boundary for the system of interest in its larger operational context. The last steps of the system analysis for this method are determining system losses and hazardous states. The stakeholders should agree on what they cannot afford the system to do. A system loss is a stakeholder-defined unacceptable outcome or effect of system operation. After a baseline of these losses are agreed upon, then hazardous system states are identified. These hazardous states are modes/operating states that when combined with worst-case environmental conditions can lead to a system loss. A common pitfall is identifying environmental conditions as system hazards. For instance, poor visibility or bad weather is not a system hazard, operating with visual only guidance would be the hazardous system state that when paired with poor visibility could result in a system loss.

Based on the results of the system analysis, a first pass at initial security goals for the system is able to be generated. These system-level security goals are constraints to prevent the system from entering the hazardous states identified in Step 4 in Figure 3.1.

3.1.3 Behavior Analysis

After establishing the system's purpose and goals, and constructing a context diagram for the system, the next steps of EGRESS transition to behavioral analysis of the system of interest (SoI). A SysML use case diagram serves as a means of establishing behavioral goals for the system. It is important to note this step should be completed without a preliminary architecture in mind, and should not introduce any design constraints. Once this use case diagram is completed, further elaboration of security goals can be completed by creating a misuse case diagram in step six. This misuse case diagram adds an attacker to the use case diagram and captures a high-level brainstorming activity of how an attacker may exploit the system use cases. Exploits are categorized in terms of confidentiality, integrity, and availability - the CIA security triad. This misuse case diagram is not intended to be an exhaustive threat modeling activity, but instead, a thought-provoking exercise to capture some key high-level potential exploits. Next, in step seven, high-level activity diagrams are elaborated for each use case. These are captured in a system model, which creates links to each activity diagram in top-level use case diagrams.

3.1.4 Iteration

Misuse cases and functional modeling refine the SoI's initial security goals. EGRESS iterations are likely to uncover security goals, hazards, and losses that need modifications.

3.1.5 Security Critical Functionality and Iteration

The final activity of EGRESS is another security brainstorming activity in which potential security critical functionality are identified in the activity diagrams completed in step seven. As with the misuse case activity, this is not an exhaustive listing, but a thought provoking exercise to highlight a few critical actions across the activity diagrams. These security critical functions

are used to validate the initial security goals. Attempting to map these security critical functions to security goals identifies new security goals that need to be captured. At this point, update the security goals and revisit our losses and hazards to confirm if they need updating as well.

3.1.6 Output - Initial System Goals for Security

After the completion of this iteration, the team should have a working set of security focused system goals to guide further system design and systems modeling activities, forming the foundation upon which SMART security requirements can be elicited.

3.2 EGRESS variant focused on alignment with the RAAML standard and minimizing the use of custom stereotypes

As a part of this research work, a variant of EGRESS was modified and simplified to maximize alignment with the Object Management Group (OMG) Risk Analysis and Assessment Modeling Language (RAAML) standard. This was an intermediary version of EGRESS and not the final product described at the beginning of this chapter. This variant was accepted for publication as a standalone work at the IEEE AEROCONF 2025, citation pending. It is worth of inclusion for additional discussion on maximizing the alignment with existing standards. The final product of EGRESS accepts the trade-off of additional custom stereotypes for the additional clarity provided by their use.

The Object Management Group (OMG) Systems Modeling Language (SysML) is a general purpose graphical modeling language for representing systems in a digital environment intended to support the specification, design, analysis, and verification of systems [80]. The Risk Analysis and Assessment Modeling Language (RAAML) is an extension of SysML that leverages existing SysML elements and constructs to support security, safety, and reliability analyses [81]. This variant of EGRESS uses the System-Theoretic Process Analysis (STPA) and Goal Structure Notation (GSN) portions of the RAAML specification to accomplish the majority of the EGRESS method. STPA is a hazards analysis technique based on an extended understanding of accident causation

by asserting mishaps can stem from unsafe interactions of system components without a single point of failure [55]. GSN is a graphical notation that represents engineering arguments and the relationships between components of the overall goal structure [82].

The following variant of the EGRESS methodology in Figure 3.3 is different from Figure 3.1 in that it omits the security-critical functionality in step 8 and reduces the use of custom stereotypes in the same effort. Chapter 4 applies this variant of EGRESS to a Space Traffic Management System (STMS) concept. See Chapter 4 for a discussion of the trade-offs of utilizing this variant in minimizing custom stereotypes.

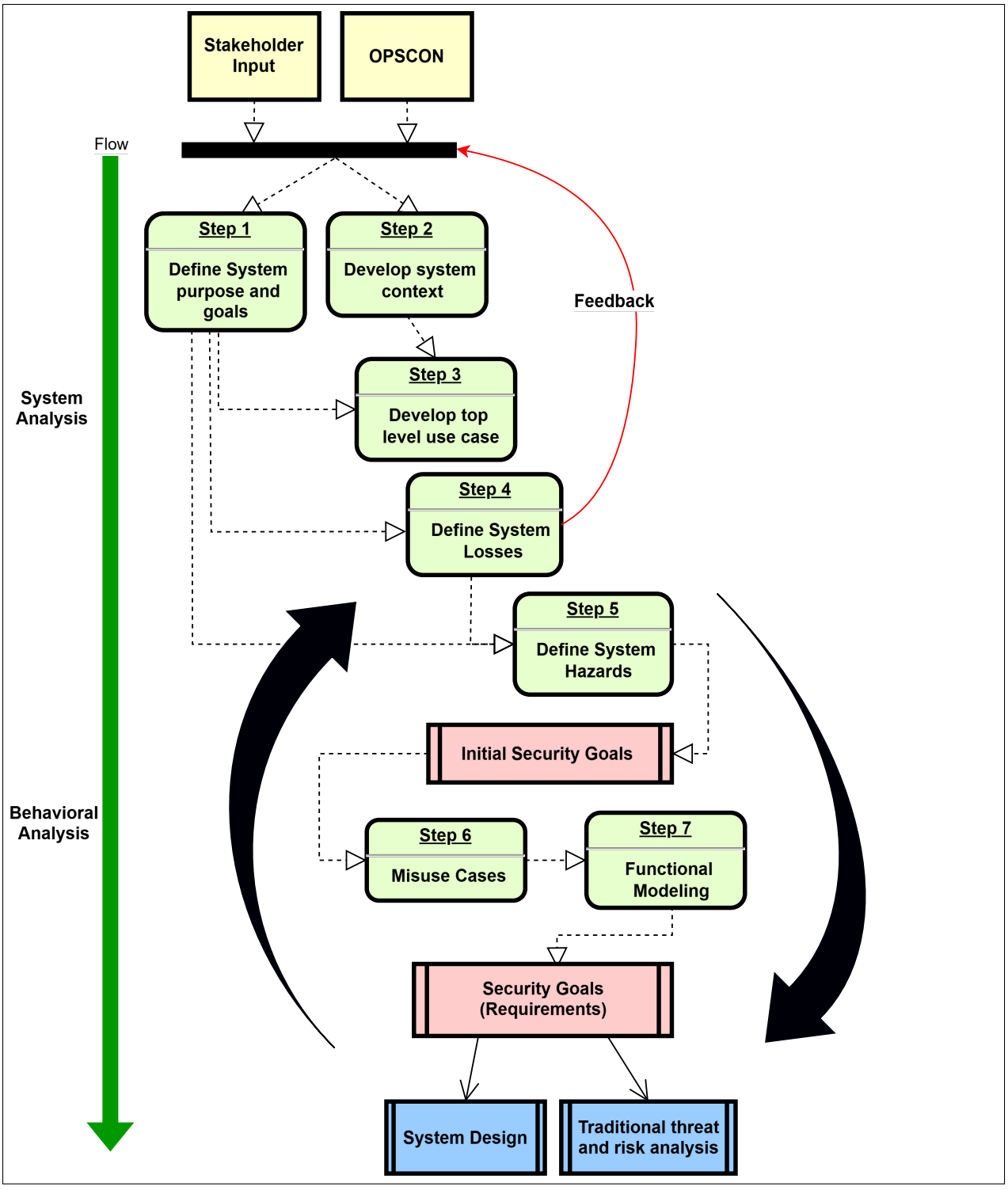

Figure 3.3: Eliciting Goals for Requirement Engineering of Secure Systems (EGRESS) Tailored Method to Maximize Alignment with RAAML

Chapter 4

EGRESS Applications

4.1 EGRESS Method Demonstrated for a Space Traffic Management System (STMS) Concept

To demonstrate the utility of EGRESS, the next sections apply the method to a fictitious system concept without architecture definition. The SoI in the following example is a space traffic management system (STMS) with the purpose of managing space traffic. The SoI is meant to identify space objects, track the objects, and predict orbits and movements. High-level STMS architecture and mission context were loosely derived from the open-source spacecraft model found on github.com that accompanies the book by Freidenthal and Oster [80]. Excerpts from a sample CONOPS of the system are shown in Figure 4.1.

4.1.1 Define System Purpose and Goals

The first step of the EGRESS method is to develop clear system purpose and goal statements. These statements are designed to guide the rest of the modeling process conceptually and should be taken directly from stakeholders and any system documentation available, ideally a CONOPS or OPSCON. High-Level Stakeholder needs, system-level CONOPS requirements, should be captured so that traceability to the source can be demonstrated to further bolster the security goals output from this method. Figure 4.1 shows the top-level CONOPS-based requirements captured for the STMS.

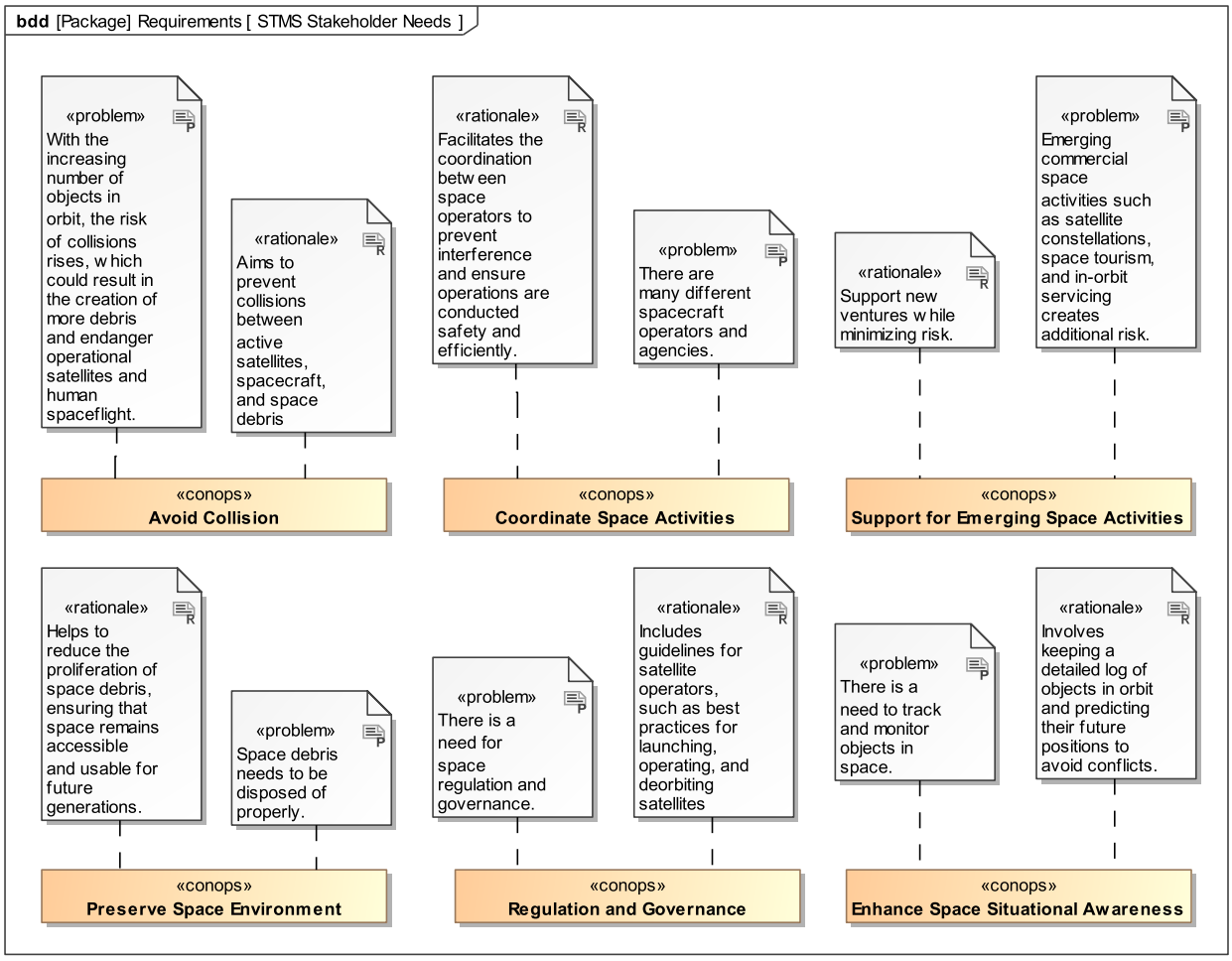

Figure 4.1: CONOPS Based Requirements

Purpose and goals are the highest level of abstraction of design and constructed after collecting stakeholder input regarding the mission and vision of the system's intended use. Adapted from STPA-Sec [83], the system purpose captures the primary mission (i.e., the What, or *Focus*). The Strategy represents design methods and identifies the activities the system must complete to achieve its purpose (i.e., the How). Lastly, the higher level mission (i.e., the in order to) is captured as the overarching system goals. The process of defining system purpose and goals has proven effective throughout the authors' previous work related to STPA-Sec [57], [58], [60], [61]. Defining the system mission allows for prioritization of security goals and trade-offs in design.

To comply with the RAAML specification, stereotypes from the standard were leveraged. The GSN profile supplies the *Goal* and *Strategy* stereotypes, while *Focus* originates from Unified Modeling Language (UML). This diagram is a tool for stakeholder alignment, to ensure that all parties establish consensus on the definitions. Figure 4.2 shows a UML Class diagram with the Space Traffic Management System (STMS) *Focus* (i.e., mission), *Strategy* (i.e., method), and *Goals*. Figure 4.2 further shows the STMS Focus as *Manage Space Traffic* using the strategies of *Identify, Predict,* and *Track* to achieve the goals of *Ensure integrity of space assets* and *Make efficient use of space domain.*

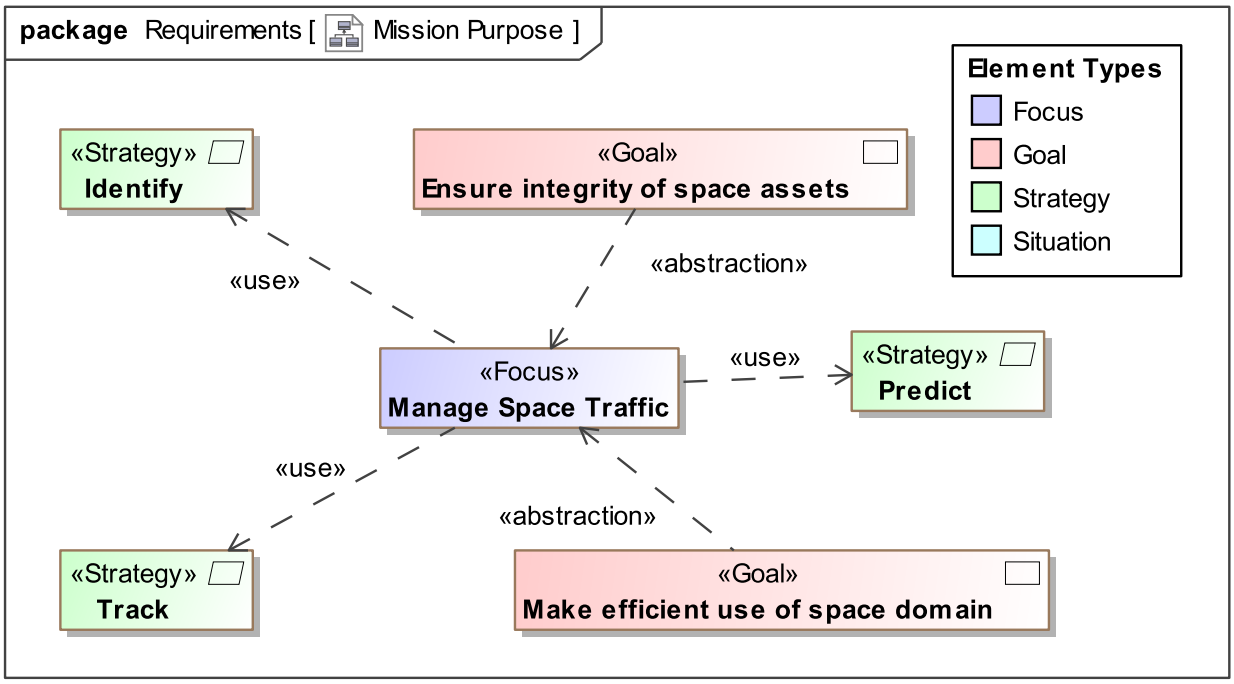

Figure 4.2: Space Traffic Management System (STMS) Purpose and Goals

4.1.2 Develop System Context

By developing the STMS context, the components of the intended operational environment are captured. Stakeholders consider the operational domain and external entities that interact with the system. The objective of this process is to provide an outline of the system domain, clarifying the system boundary. This phase culminates in a structural representation of elements outside of the

context scope. Figure 4.3 is a visual representation of the STMS mission context using a SysML Block Definition Diagram (BDD)

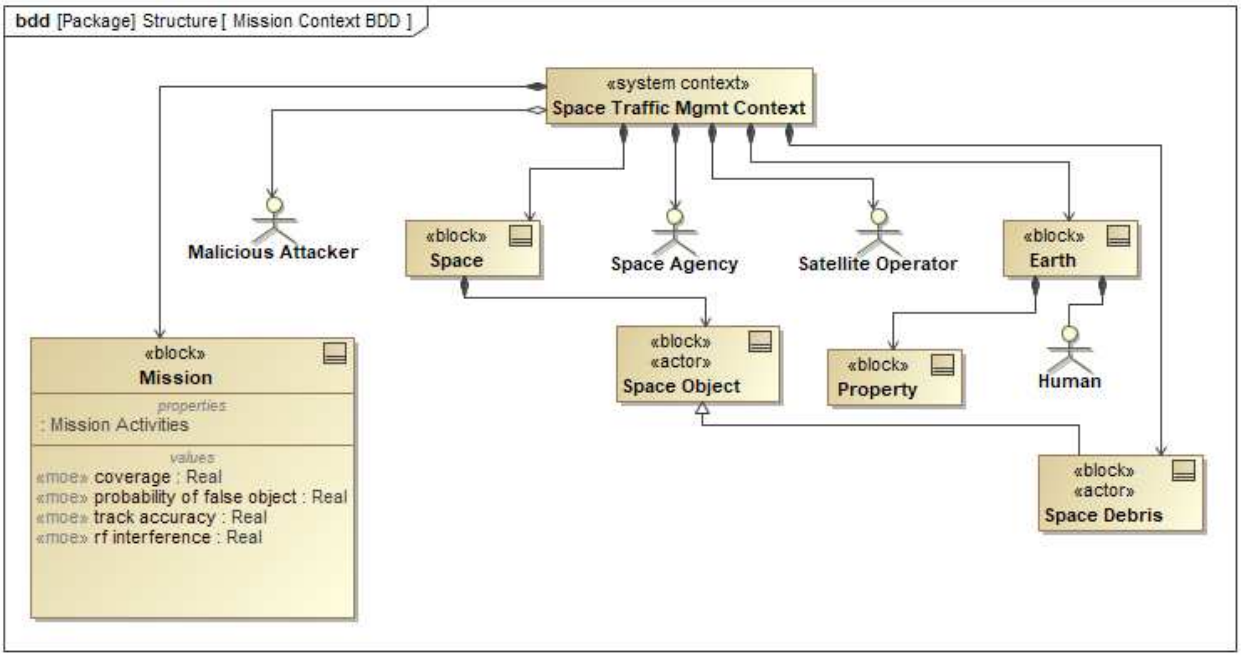

Figure 4.3: Space Traffic Management System Mission Context - Reprinted from Figure 2.2 for Convenience

Figure (Figure 4.3) illustrates the elements that compose the SoI operational environment. The STMS context includes the mission, space, space debris, and earth.

Figure 4.4 is intended to be iterative and is expected to change throughout the system design and development phases. The view is conducive to understanding the potential structure of the SoI.

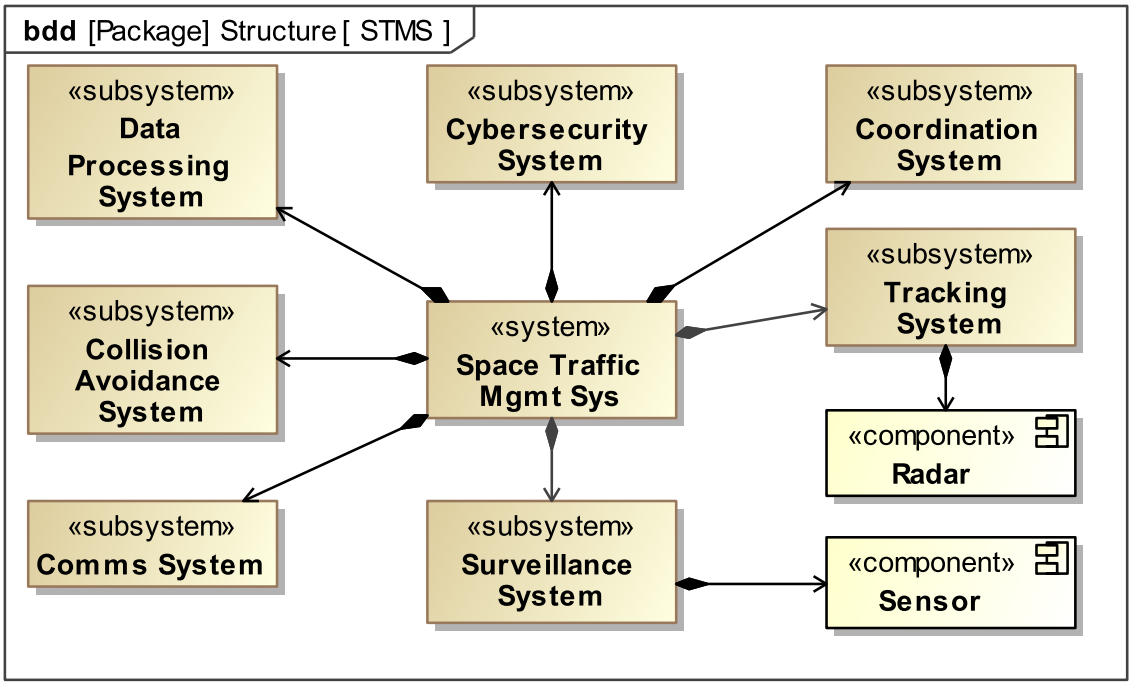

Figure 4.4: STMS System Composition - Reprinted from Figure 2.3 for Convenience

4.1.3 Elaborate Use Cases of SoI

Step 3 of EGRESS utilizes SysML use case diagrams to depict the primary objectives of the SoI from the perspective of its users [84]. This provides a broader understanding of STMS functional aspects and outlines high-level behavioral goals that the system must fulfill.

Building upon the STMS purpose, goals, and context of the SoI, EGRESS recommends a SysML use case diagram to establish behavioral goals for the system without a preliminary architecture to introduce any design constraints. The STMS SysML use case diagram depicted by Figure 4.5 reflects the critical interactions of the SoI with entities external to the system boundary.

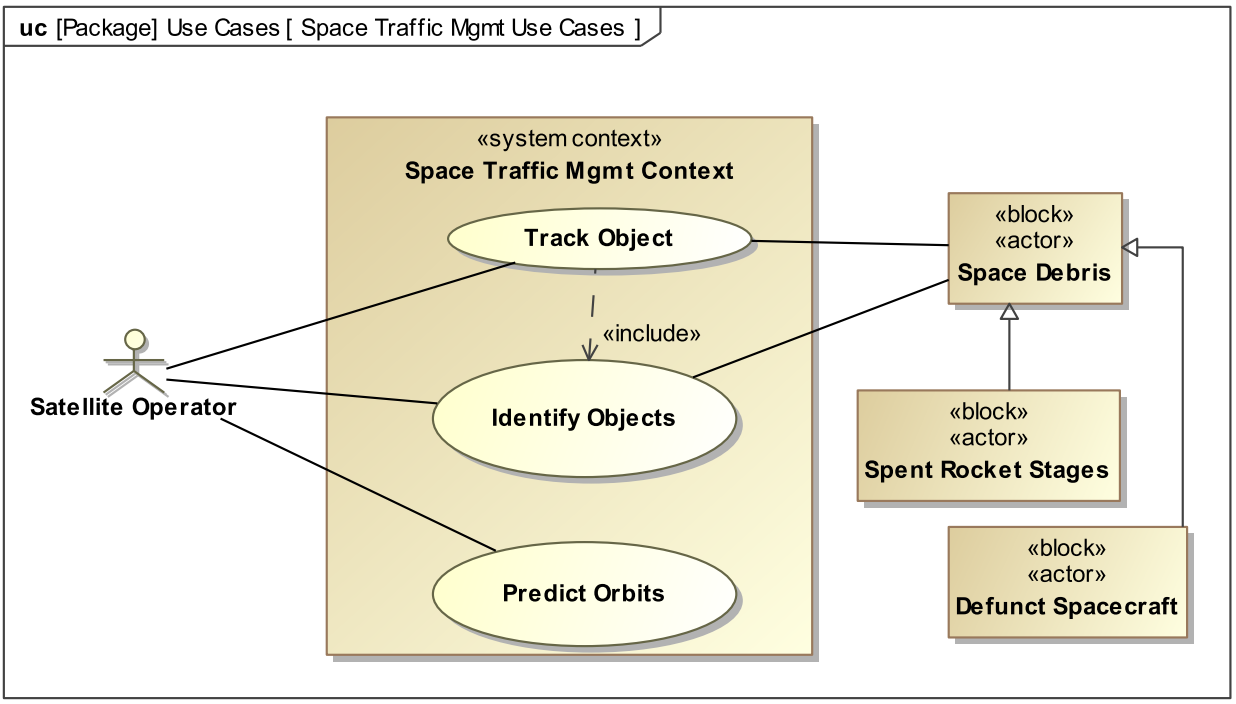

Figure 4.5: Space Traffic Management System SysML Use Case Diagram - Reprinted from Figure 2.4 for Convenience

As illustrated in Figure 4.5, the use cases of the STMS are *Identify Objects*, *Track Objects*, and *Predict Object Paths*. The key actors are *Space Agency and Satellite Operator*, and external actors to consider are *Space Debris*. Further behavior of the SoI is identified by elaborating base use cases with SysML Activity diagrams, which is completed in Step 7 of EGRESS.

4.1.4 Define System Unacceptable Losses and Hazardous States

This step involves the systematic definition of unacceptable losses and the identification of hazardous states that may lead to these losses. Unacceptable losses are specific, undesirable outcomes as defined by the stakeholders. They should identify what is of the highest value to the stakeholders and differentiate between what is desired and what is required. Unacceptable losses can be associated with mission, personnel, or equipment. A concerning outcome of potential system behavior should be captured and clearly understood. For example, loss of reputation or unauthorized disclosure of critical data are considered unacceptable [57]. Hazardous system states describe con-

ditions that can lead to an unacceptable loss when paired with a worst-case set of environmental conditions. As a general rule, hazards of the SoI should be abstracted to the highest level possible. Identifying hazards enables refinement and clarification of unacceptable losses. Each hazard should be mapped to one or more unacceptable losses, otherwise, it is likely a loss is missing and need to be captured to map to the hazardous state.

Mapping hazards to losses is shown with a traceability matrix, Figure 4.19.

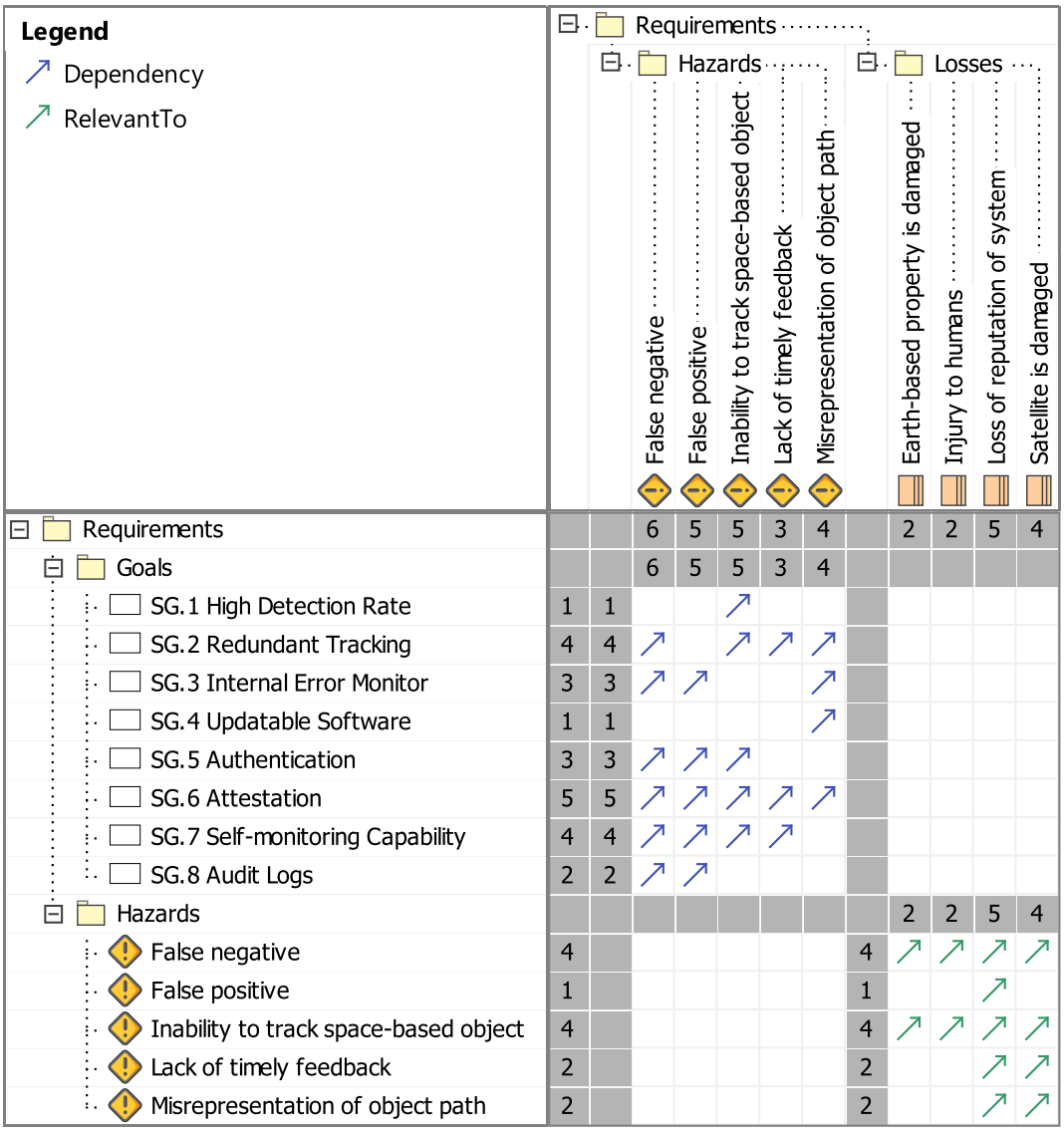

Figure 4.6: STMS Losses and Hazards Matrix

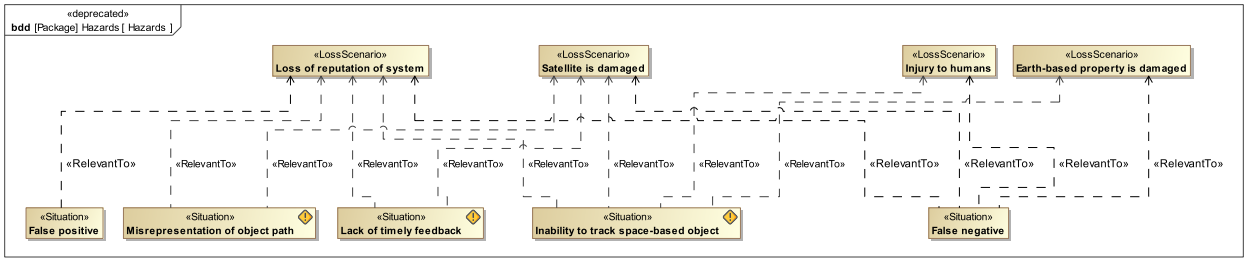

Figure 4.7: Hazards Traced to Losses Attempting to Comply with RAAML 'Situation' Stereotype for Hazards

Figure 4.7 shows an attempt to comply with the RAAML standard for capturing Hazards and Losses. RAAML has a stereotype for LossScenario but calls Hazards, 'Situations' which is very ineffectively undescriptive. Additionally, the specified RAAML relationship of 'Relevant To' also lacks desired descriptiveness of this relationship. 'Causes' would be much more appropriate and simple.

As shown in the EGRESS Methodology, Figure 3.1, this step provides an opportunity to iterate on initial EGRESS outputs and update the system OPSCON as needed. The stakeholder alignment and system understanding gained by implementing EGRESS before system architecture definition provide the basis for design efforts.

4.1.5 Initial Security Goals

Initial security-focused goals for the STMS can be elaborated by identifying design constraints that prevent the system from entering the hazardous states defined in Figure 4.19. Constraints are restrictions placed on the system to be further refined into security requirements. They are implemented via the system architecture to ensure SoI operation operates within acceptable parameters. This encourages stakeholders to discuss and define what is "secure enough" for a given SoI [85]. These security goals are defined using a table, Figure 4.8, and then using an MBSE tool are allocated to corresponding system hazards which ensures that each system Hazard has been 'prevented' by security goals. These security goals bound functionality, constraining the system from entering the hazardous system states. These goals provide the foundation of future security requirements and constraints. This allows additional iterations to further specify bounds as

the system matures. Establishing goals before architecture definition enables simultaneous consideration of system security on equal footing with functionality considerations in the design and development process.

Goal #	Title	Description of Goal
SG.1	Detection Rate	The system shall have a high detection rate to be further specified in design
SG.2	Redundant Tracking	The system should have redundant object and tracking systems or capabilities - not necessarily for reliability, but for security mitigation of hacking one tracking system
SG.3	Internal Error Monitoring	The system shall be able to predict its error, provide on a confidence on a track/collision potential
SG.4	Updatable Software	The system shall have updatable algorithms/software for tracking - creates potential hazard of unverified software update

Figure 4.8: Initial STMS Security Goals

4.1.6 Misuse Cases

A best practice based on previous SE-based security work, [86], includes creating misuse cases for the SoI, and identifying high-level attack scenarios and the involved malicious actors. This is not intended to replace a complete vulnerability or attack path assessment. The misuse case diagram is used as a brainstorming tool to consider high-level potential attack vectors independent of a candidate system architecture.

The goals of this step are twofold:

1. This step identifies how hazardous states are entered from a security perspective and is used as a technique for identifying those not previously considered.

2. Misuse Cases provide a check of the initial security goals is performed by reviewing how goals in Figure 4.8 prevent misuse cases.

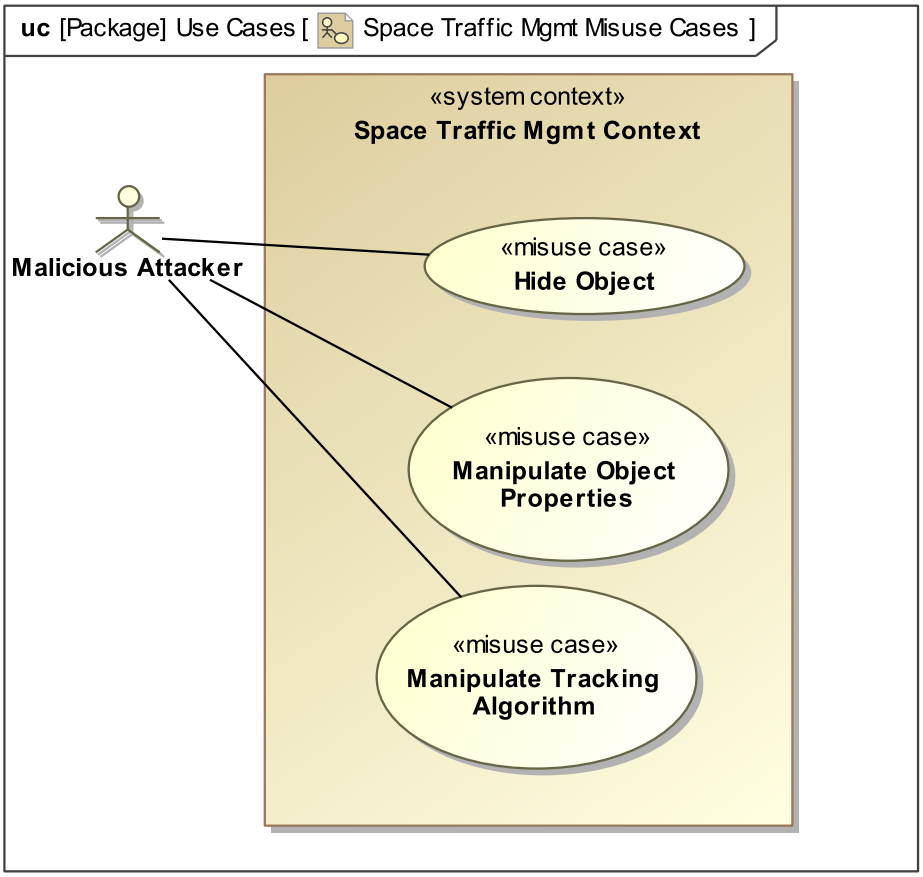

Figure 4.9: STMS Misuse Cases

Figure 4.9 shows the STMS misuse cases of *Hide Object, Manipulate Object Properties,* and *Manipulate the Tracking Algorithm.*

4.1.7 Functional Modeling

The STMS use cases identified in Figure 4.2.3 are elaborated with SysML Activity diagrams. These views detail and allocate behavior to the entities responsible for each behavior. They also

capture conditional logic required to execute the use case. Figure 4.10 shows the sequence of activities for the *Track Object* use case.

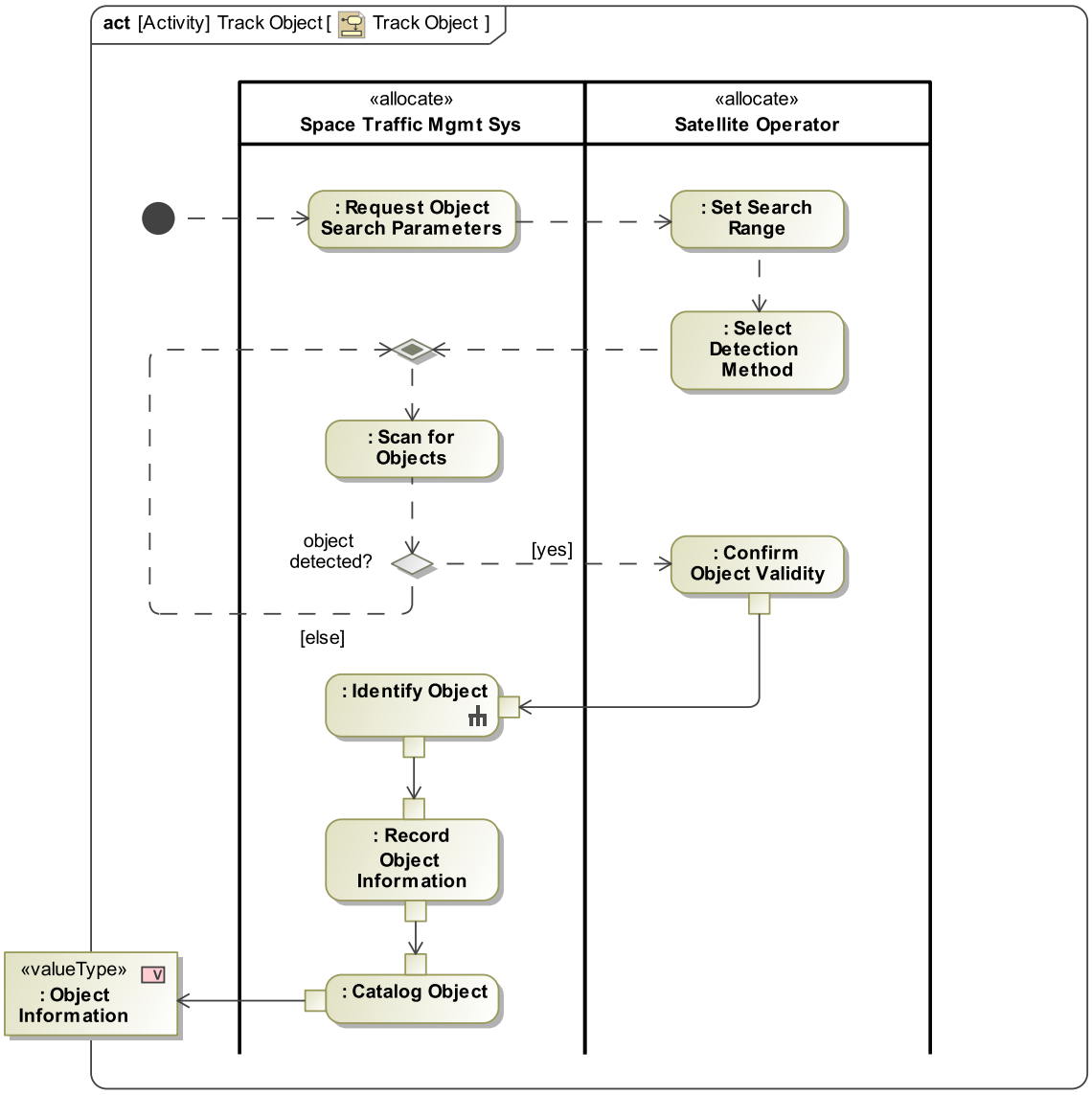

Figure 4.10: SysML Activity Diagram for *Track Object Behavior* - Reprinted From Figure 2.5

The *Object Information* value type represents the data being transferred between the STMS and the Satellite Operator. This item is an input to the *Identify Object* SysML Activity diagram, shown as a parameter on the diagram of Figure 4.11 frame. Once the data has been confirmed and logged, it is an output of the *Track Object* behavior.

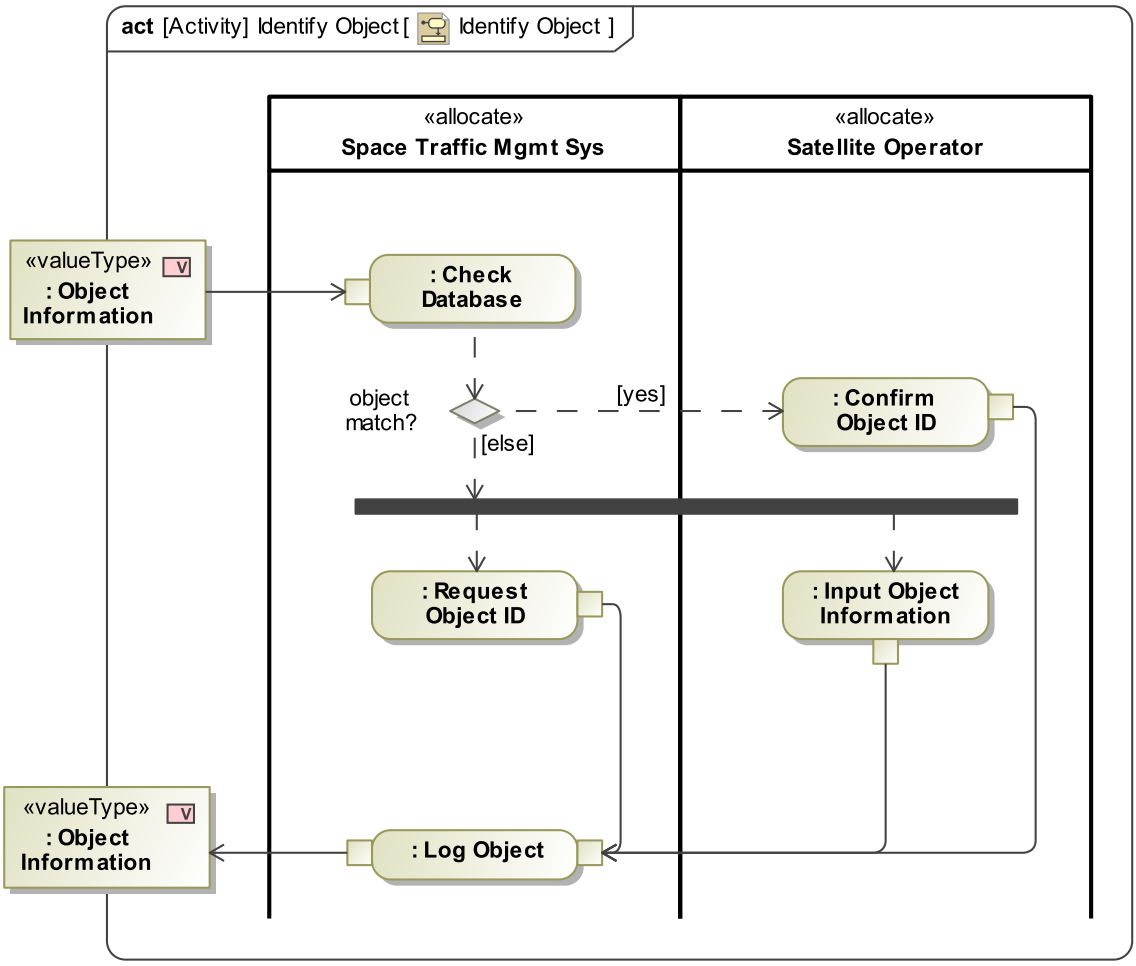

Figure 4.11: SysML Activity Diagram for *Identify Object Behavior*

4.1.8 Goal Verification

Dependency matrices created within the model are used to verify all SoI hazards are linked to a security goal, as shown in Figure 4.12. In the event a hazard is not associated with a goal, additional security measures must be considered.

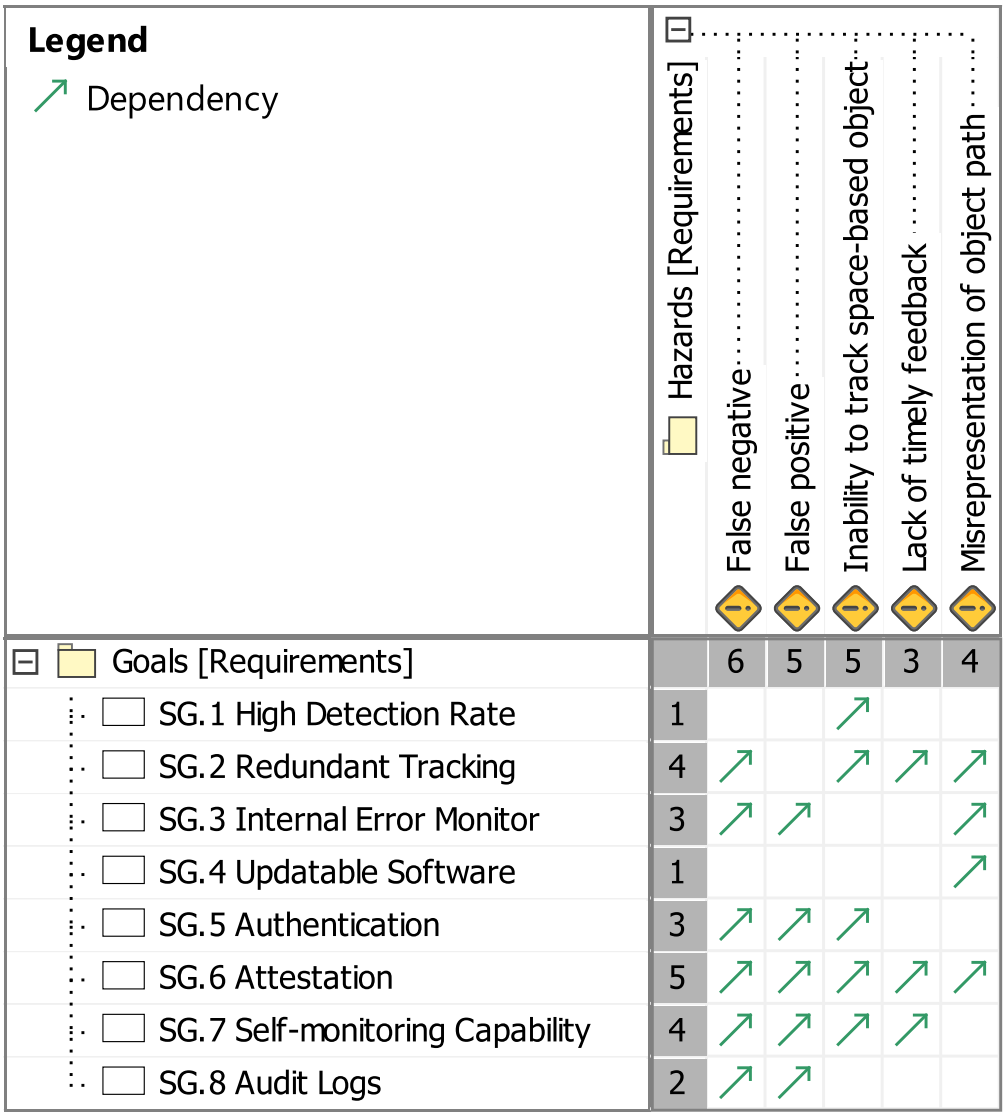

Figure 4.12: STMS Goals Traced to Hazards

An example of results from evaluating misuse cases from Step 6 in Figure 3.1 identified missing security goals to prevent the execution of those attack vectors. Specifically, the lack of user authentication and attestation was evident and elicited additional security goals, SG.5 and SG.6. The *Manipulate Tracking Algorithm* use case also highlighted the need for the creation of SG.7 and SG.8 to identify degraded capability in the system and have an audit report of system changes and the user implementing those.

4.1.9 Output - Initial System Goals for Security

Figure 4.13 shows the security goals elicited from the EGRESS method for an STMS. These represent the evolution of system-level security goals and will be used as the foundation for deriving measurable requirements. This example shows how goals were updated to reflect additional information obtained through the EGRESS process.

Goal #	Title	Description of Goal
SG.1	Detection Rate	The system shall have a high detection rate to be further specified in design
SG.2	Redundant Tracking	The system should have redundant object and tracking systems or capabilities - not necessarily for reliability, but for security mitigation of hacking one tracking system
SG.3	Internal Error Monitoring	The system shall be able to predict its error, provide on a confidence on a track/collision potential
SG.4	Updatable Software	The system shall have updatable algorithms/software for tracking - creates potential hazard of unverified software update
SG.5	Authentication	The system shall have authentication of users and trusted data sources
SG.6	Attestation	The system shall have attestation of users and data sources
SG.7	Self Monitoring of Capability	The system should self-report degraded capabilities
SG.8	Audit Logs	The system should have auditable logs

Figure 4.13: Output - Initial Security Goals for STMS

4.1.10 Traceability

A key benefit of the EGRESS approach is traceability of security goals to stakeholder-defined system goals, shown in figure 4.14.

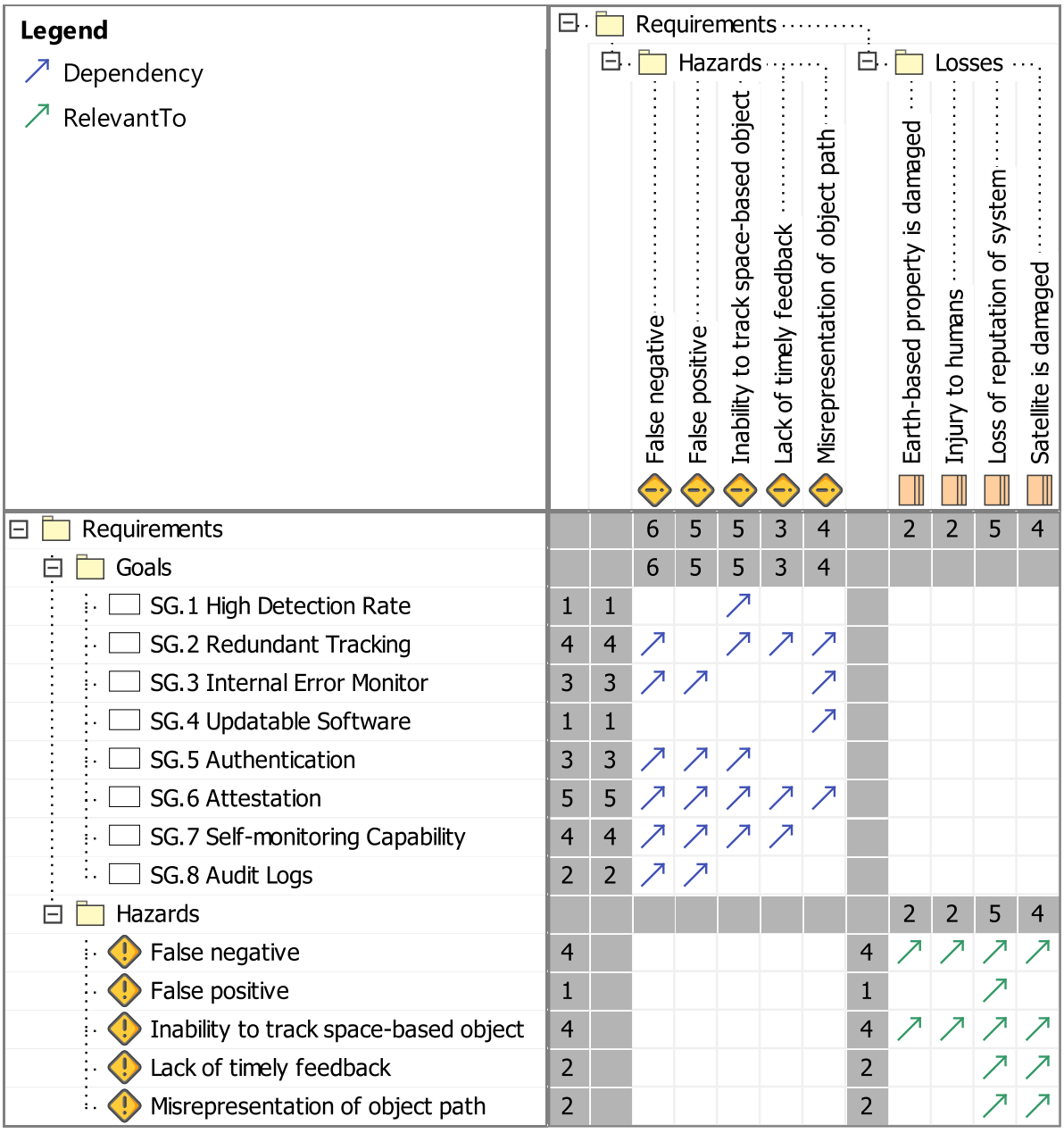

Figure 4.14: STMS Traceability matrix for goals, hazards, and losses.

Matrices produced from the model give decision-makers a visual mapping regarding the origins of required system security and functionality. This presents visual evidence to combat the mindset of security as an inconvenience or an unnecessary cost. The traceability becomes even more valuable as future iterations elicit security requirements of the SoI.

4.2 EGRESS Method Demonstrated for an Off-Road Navigation System Concept

To further demonstrate the utility of EGRESS, in this section, EGRESS is extended and applied to another fictitious system of interest. This system can be described as an 'Off-Road Navigation System,' and its proposed purpose is to provide reliable and user-friendly navigation guidance through outdoor natural terrain that does not have pre-existing paths. The system aims to provide users with reliable directions, real-time updates, and essential information to ensure safe and efficient navigation through these untamed landscapes. To accomplish this, the system will offer intuitive and personalized navigation instructions leveraging onboard sensors, GPS communication, and mapping algorithms. It will also consider the distinctive features of the terrain, encompassing factors such as topography, vegetation, and natural landmarks.

4.2.1 Define System Purpose and Goals

The first step of the EGRESS method is to develop clear system purpose and system goal statements. These statements are designed to guide the rest of the modeling process in a conceptual manner and should be taken directly from stakeholders and any system documentation available, ideally a CONOPS or OPSCON. Purpose and goals statements are at the highest level of abstraction for the system and ensure stakeholder agreement on the mission and vision of the system's intended use. The system purpose should capture the primary mission of the system in a few words (i.e., the What). The method, "by means of", identifies the activities or processes the system uses to achieve its purpose (i.e., the How). Lastly, the goal "in order to" identifies what mission the system contributes to (i.e., the Why). This process of defining system purpose and goals has proven

effective throughout the authors' previous work related to STPA-Sec [57], [58], [60], [61]. Accurately defining the desired system's purpose and goal requires involvement from key stakeholders such as mission owners, operators, and users. Moreover, correctly defining the mission provides a baseline for prioritizing and performing security tradeoffs based on documented stakeholder needs.

It is recommended to document the system purpose and goals through a custom SysML Block Definition Diagram that characterizes the elements and defines their relationship to one another. Custom block stereotypes *Goal*, *Purpose*, and *Method* are created to allow the user to clearly label each block. Custom relationships of *In order to*, and *By means of* define the relationship between the elements. These were implemented to demonstrate how the elements relate to one another, for example, the *Purpose* of the system must be achieved *In order to* accomplish the system *Goal*. The system *Purpose* is completed *by means of* the *Method(s)*. This diagram is a tool for stakeholder alignment, to ensure that all parties agree on the definitions.

In the case of the Off-road Navigation System, a purpose and goal diagram captures the highest level of abstraction of the system. These statements bring the stakeholders in agreement and can be documented both in the model and the system OPSCON to guide the security goals elicitation process. Figure 4.15 illustrates the use of the purpose and goal diagram.

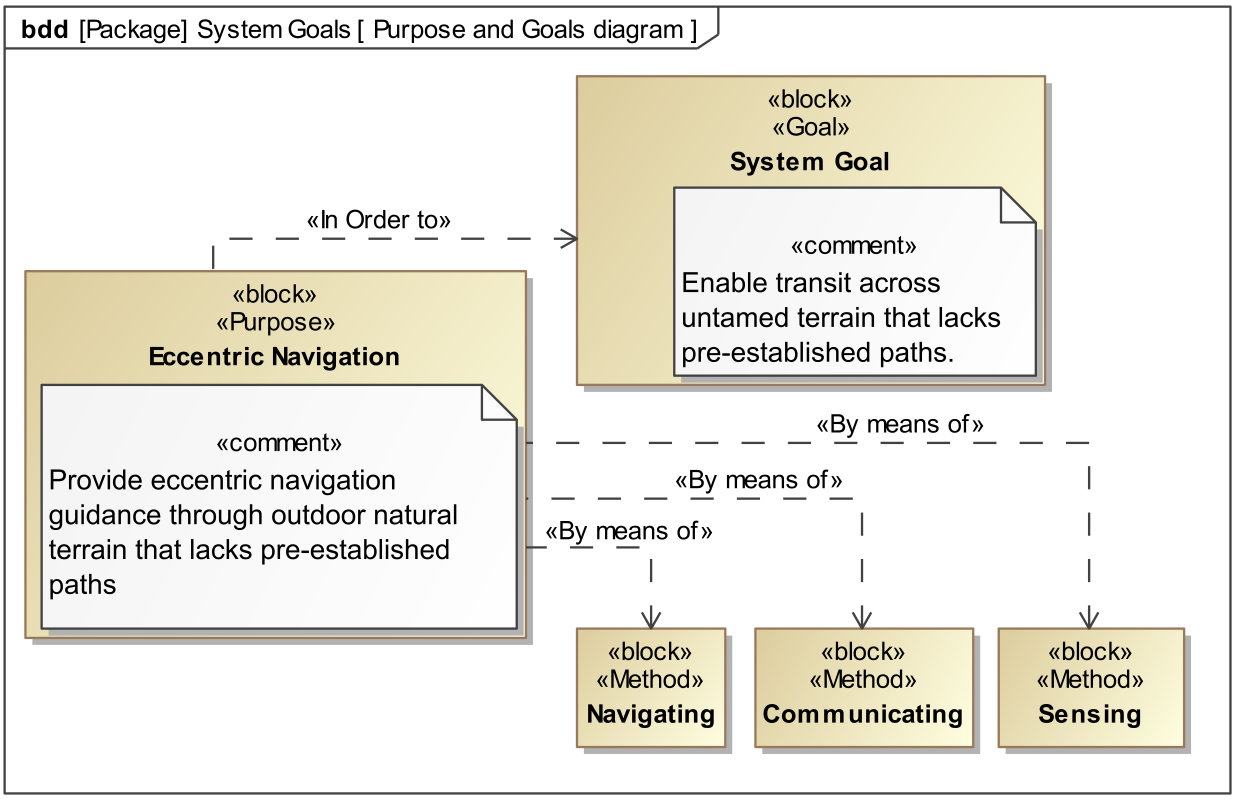

Figure 4.15: Off-Road Navigation System Purpose and Goals Diagram

As shown in Figure 4.15, the three block types of *purpose*, *goal*, and *method* are elaborated and the relationship between each one is clear.

4.2.2 Develop System Context

The purpose of this step is to capture the context in which the system will operate. In this step, system engineers and stakeholders consider the domain in which the system operates, including external actors of interest that may either directly or indirectly interact with the system. This step can be considered 'complete' when all relevant elements external to the system have been identified. The objective of this step is to provide a clear outline of the system domain, to provide all parties involved an understanding of what is 'in bounds' for the system design and what is a relevant environmental concern, but not a part of the system design. It is important to note that this phase does not focus on elaborating the internal parts or behavior of the system, rather it is a

structural representation of all external elements of the system around the SoI. This step is captured in a SysML block definition diagram (BDD) showing all the objects and elements using system blocks.

In the case of an Off-road Navigation System, the majority of the elements pertain to the environment in which it operates. This diagram frames the 'big-picture' context of the system of interest through top-level blocks that represent key elements within the environment.

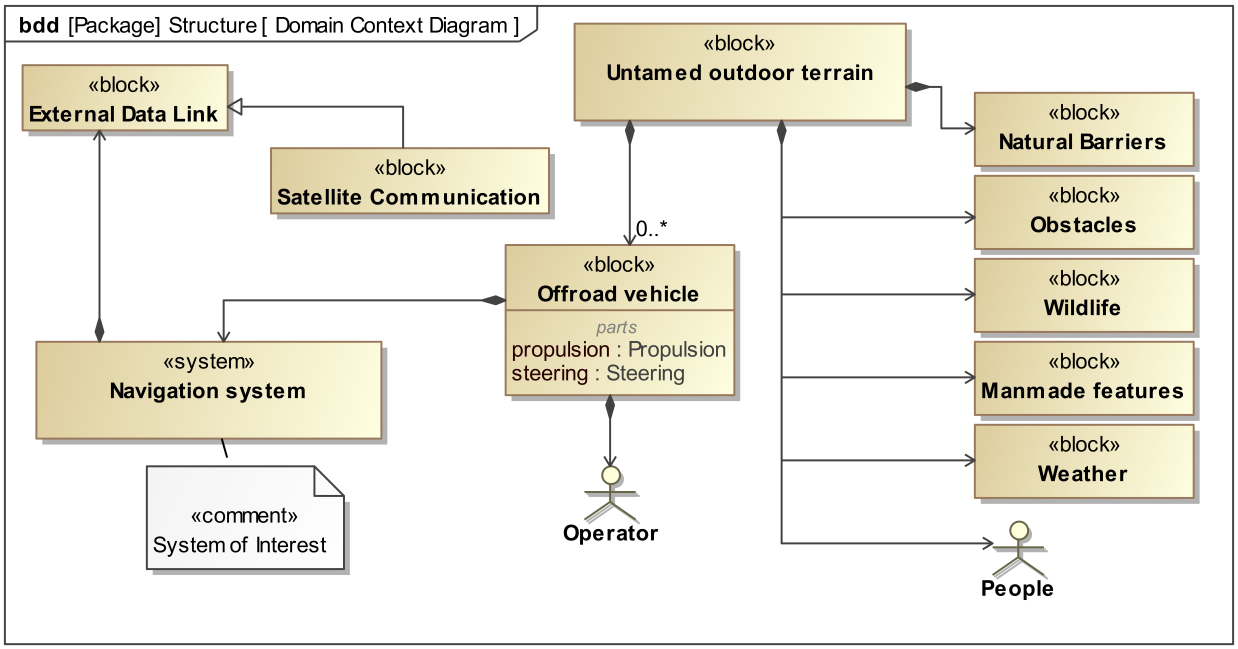

Figure 4.16: Off-Road Navigation System Context Diagram

The context diagram (Figure 4.16) illustrates the elements that the system of interest will interact with. As can be seen in the diagram, the navigation system is modeled as a 'component' of the Off-road vehicle and is highlighted as the system of interest. The elements within the environment such as *Natural Barriers* and *Weather* represent objects that the system of interest will physically interact with, whereas the *External Data Link* represents an external actor that the navigation system will interact with for data and communication purposes. It is important to note that this SoI does NOT include the off-road vehicle itself, only the navigation system. This is further shown in the Asset Definition diagram of Figure 4.17.

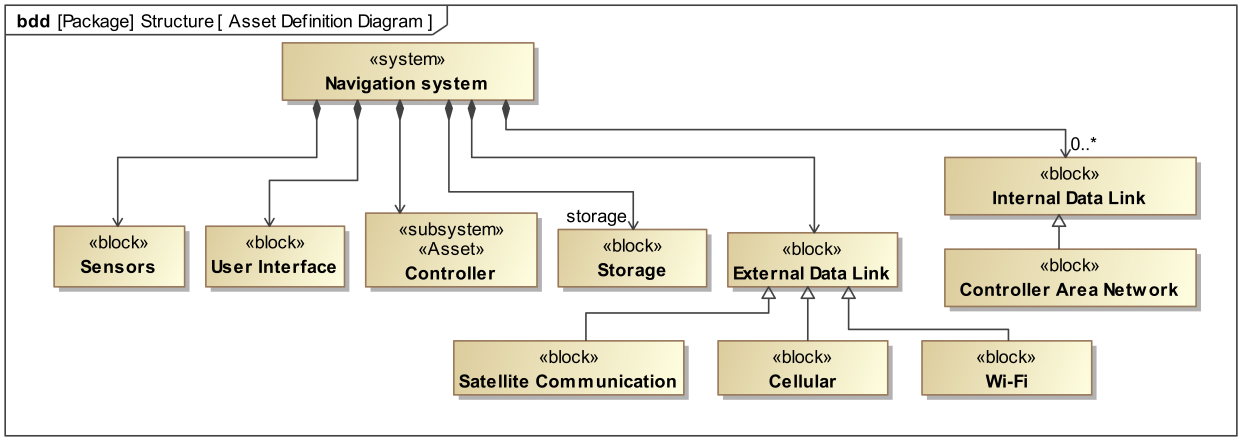

Figure 4.17: Off-Road Navigation System Asset Definition Diagram

4.2.3 Elaborate Use Cases for SoI

Step 3 of EGRESS involves developing a top-level use case diagram for the system of interest (SoI). The primary purpose of this diagram is to depict the system's primary objectives from the perspective of its users [84]. These objectives are defined in terms of the functions the system needs to support and capture key external entities or actors interacting with the system. This provides a broader context for understanding the system's functional aspects and outlines high-level behavioral goals that the system must fulfill.

Developing this diagram serves several important purposes. Firstly, it identifies key actors involved in the system and ensures that the primary use cases encompass critical behavioral elements of the system. Additionally, this diagram acts as a valuable tool for brainstorming and facilitates reaching stakeholder agreement. Furthermore, it aids in establishing and refining the system's context and boundaries as the development process progresses.

Building upon the system's purpose and goals, and context diagram for the system, the use case diagram serves as a means of establishing behavioral goals for the system. It is important to note that this step should be completed without a preliminary architecture in mind, and should not introduce any design constraints. The Off-road Navigation System use case diagram (Figure 4.18)

reflects the key use cases for this system. In the diagram, the user is represented on the left, and any external actors would be represented on the right.

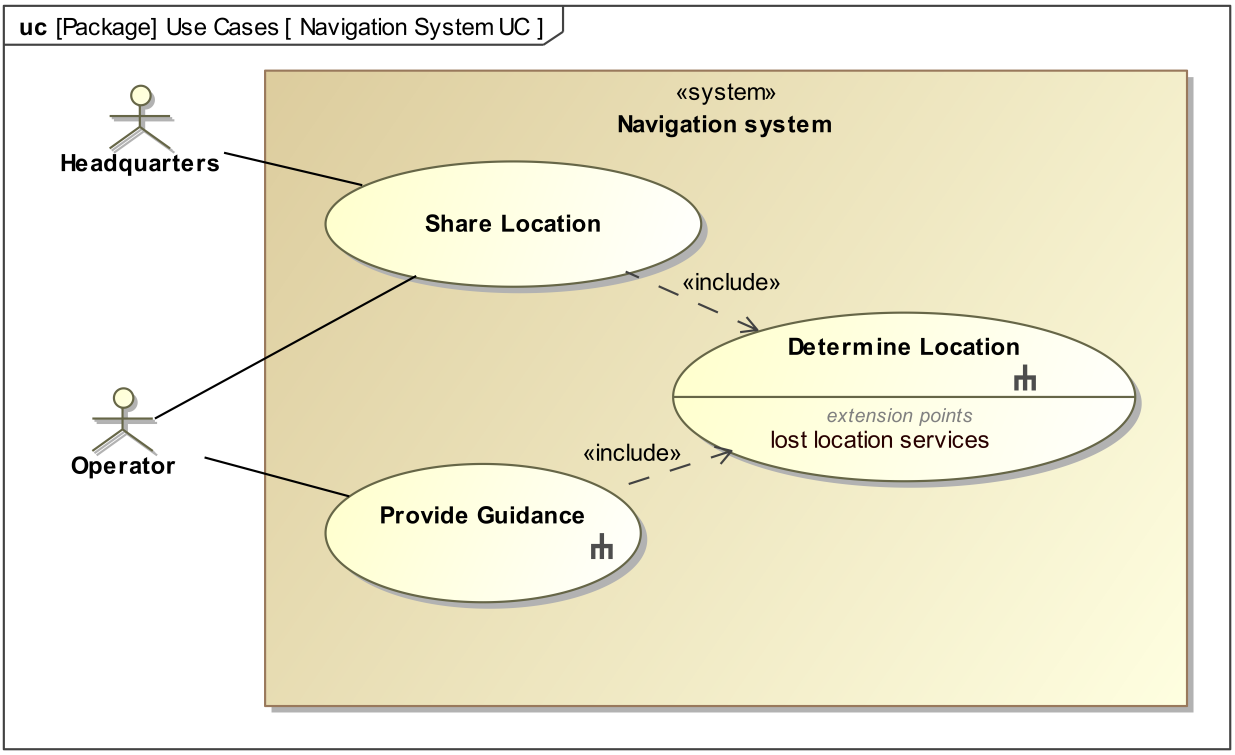

Figure 4.18: Off-Road Navigation System Use Case Diagram

As illustrated in Figure 4.18, the two primary use cases interacting with the user of the system are *Share Location*, and *Provide Guidance*. These Use Cases represent the two primary functions that the system must accomplish with user input, and both include the use case *Determine Location* to accomplish the intended functionality.

4.2.4 Define System Unacceptable Losses and Hazardous States

This step involves the systematic definition of system unacceptable losses and the identification of hazardous states that may lead to these losses. Unacceptable losses are specific, undesirable outcomes as defined by the key stakeholders. System losses should identify what is of highest value to the stakeholders and differentiate from what is nice to have/desired. Unacceptable losses can be

mission, personnel, or equipment losses, with common unacceptable losses including loss of life or mission essential equipment. In a general sense, any outcome that a key stakeholder is concerned about should be captured and clearly understood. For example, loss of reputation or loss of critical data are examples of other potentially important unacceptable losses [57]. Hazardous system states are a set of system conditions, that when paired with a worst-case set of environmental conditions, can lead to an unacceptable loss. Note, hazards are not environmental conditions or external actors. As a general rule, hazards should be abstracted up to the highest level possible, and in most cases, the list of hazards should be fewer than ten. Identifying hazards can also serve to refine and clarify the list of unacceptable losses, as each hazard should be mapped to one or more unacceptable losses (otherwise it is not a hazard or the list of unacceptable losses is incomplete). See [55] as a useful source with examples of defining system losses and hazards.

This step is captured by a simple block definition diagram used to document the hazard and loss elements, as well as trace the relationship between the two. Two new stereotypes, *Loss* and *Hazard* are introduced for this diagram, which enables the clear distinction between the types of elements in the diagram. These stereotypes also enable simple and easy traceability later in the model.

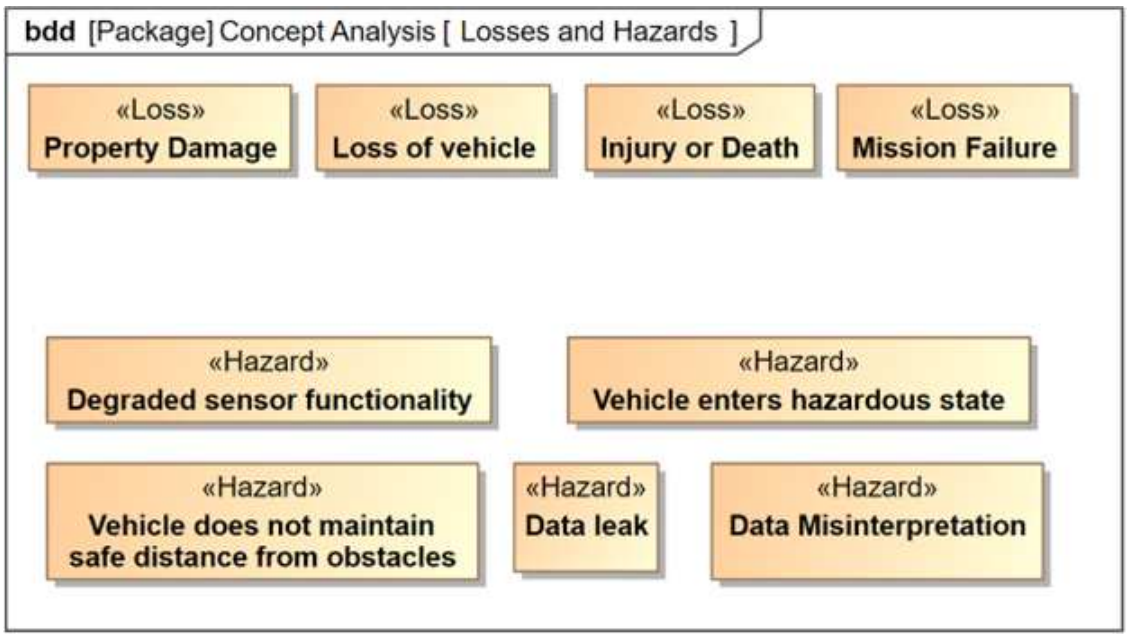

Figure 4.19: Off Road Navigation System Losses and Hazards Diagram

As shown in Figure 3.1, this step provides a good opportunity to iterate on the first few steps of EGRESS and update the System OPSCON as needed. The stakeholder alignment and system understanding gained through the activities of defining the system purpose, elaborating its context, top level behavioral functionality, and unacceptable losses is non-negligible and provides a solid foundation for the system design efforts.

4.2.5 Initial Security Goals

Initial security-focused system goals for the Off-road navigation system can be elaborated from writing system constraints to prevent the system from entering the hazardous states defined in Section 4.2.4. These functional-level system security goals prevent the SoI from entering one of the previously identified hazardous states (i.e., constraints). These constraints are restrictions placed on the system to be further refined into security requirements implemented via the security architecture to bind the SoI's operation within acceptable parameters. In particular, this step forces key stakeholders and engineers to discuss and define what is "secure enough" for a given SoI [85]. These security goals are defined using a standard SysML requirements table and are allocated

to corresponding system hazards with a custom *prevents* relationship, which ensures that each system Hazard has been addressed by security goals. As shown in the table, column one defines the security goal, column two describes the security goal, and column three allocates the goal to the appropriate Hazard(s) that it is designed to prevent. Figure 4.20 illustrates the initial security goals defined and allocated to the appropriate system hazard(s).

#	Name	Text	△ Id	Hazard
1	1 Sensor input validation	The system shall validate sensor inputs against available terrain maps	1	Misinterpretation of terrain Vehicle enters uncontrollable or unrecoverable state
2	2 Secure comms	The system shall not transmit mission critical information over unsecure methods	2	Exposure of sensitive information under the wrong conditions
3	3 Redundant Sensors	The system shall have redundant safety critical sensors	3	Degraded sensor functionality Vehicle enters uncontrollable or unrecoverable state
4	3.1 Critical Sensor ID	Critical sensors for navigation and terrain mapping must be identified	3.1	Degraded sensor functionality
5	4 Safe separation	The system shall maintain minimum user defined separation limits	4	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Vehicle enters uncontrollable or unrecoverable state
6	4.1	The system shall allow users to input minimum separation distances	4.1	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles
7	5 Selectable position sharing	The system shall have selectable position sharing capabilities	5	Exposure of sensitive information under the wrong conditions
8	6 Display and warning	The system shall display the potential hazards and obstacles along route	6	Vehicle enters uncontrollable or unrecoverable state Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles
9	6.1 Audible and visible warning	The system shall have both audible and visual warning indicators for hazards along route	6.1	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Vehicle enters uncontrollable or unrecoverable state
10	7 Changes to destination	The user must be notified and accept any change to target destination	7	Vehicle enters uncontrollable or unrecoverable state

Figure 4.20: Off-Road Navigation System Initial System Security Goals

For example, Security goal 3.1 titled *Critical Sensor ID*, is attributed to the system hazardous state *Degraded sensor functionality*.

4.2.6 Misuse Cases

This step of the EGRESS method involves further elaboration of the previously established use case diagram. A Misuse case diagram is an extension of the use case diagram that focuses on detailing the high-level attack scenarios and the involved malicious actors. This step is accomplished by adding an *attacker* actor to the use case diagram and brainstorming possible vulnerabilities and attack vectors a malicious actor may attempt to exploit. This step is NOT intended to be or replace a complete vulnerability or attack path assessment. This misuse case diagram is used as a brain-

storming tool to consider high-level potential attack vectors independent of a candidate system architecture.

The goals of this step are threefold:

1. This step brainstorms how to enter hazardous states from a security perspective, and can be used as a tool for identifying any hazardous states not previously considered before this step.

2. Developing a more detailed understanding of the potential attack vectors and potential vulnerabilities to the system.

3. A validation of the initial security goals is performed by reviewing if the goals in Figure 4.20 address preventing misuse cases identified in this step.

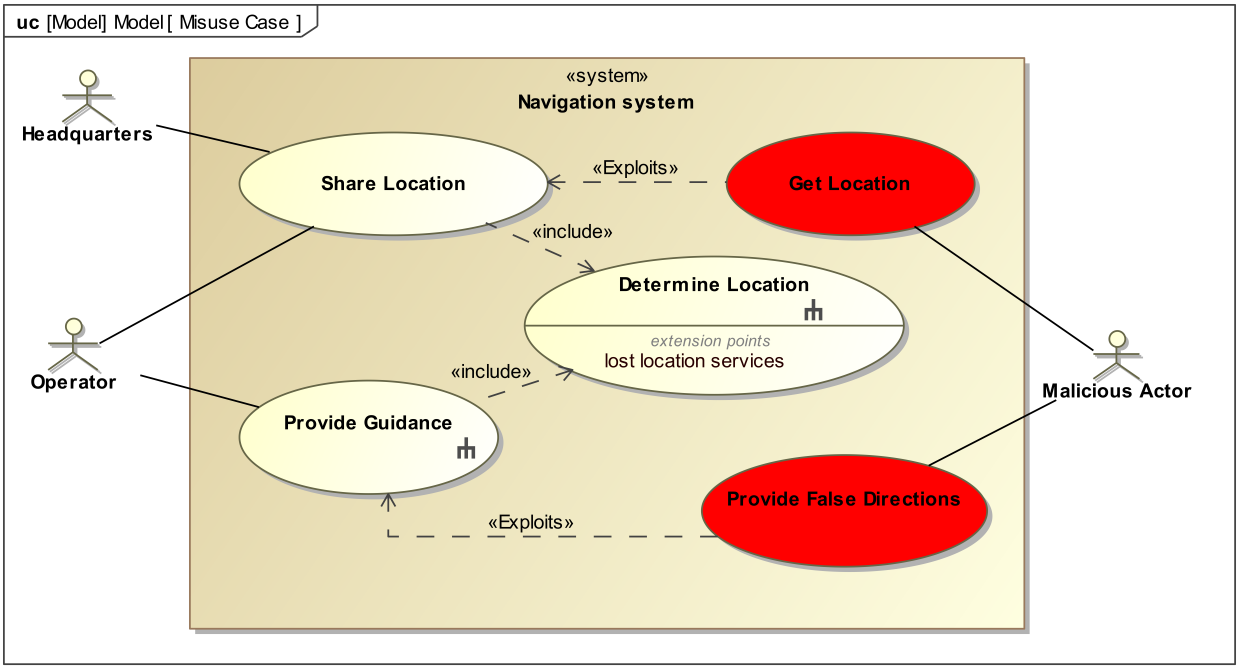

Figure 4.21: Off-Road Navigation System Misuse Case Diagram

Figure 4.21 shows the Misuse case for this example Off-Road Navigation System. An attacker element is added, and the use cases are reviewed for their susceptibility to CIA attacks. Once a susceptibility is defined, a misuse case is added to the diagram, and an activity diagram detailing

the sequence of the attack is built to broadly describe the attack sequence. As an example, for the Off-road navigation system, *providing false directions* is a potential threat. A misuse case diagram is a useful tool for brainstorming security goals by understanding, at a high level, potential attack vectors. These would be explored in much more detail via traditional Threat Analysis and Hazard Assessments later in the system design and development.

4.2.7 Functional Modeling and Security Critical Functions

In this step of the modeling process, the top-level use cases identified in 4.2.3 are elaborated through activity diagrams that describe the key behaviors of the system. It is important to note that the behavior diagrams are not intended to introduce design constraints, but rather outline the expected high-level behavior from the system. As a new contribution, this work proposes identifying important activities in each activity diagram as security-critical functions as a custom stereotype *security critical function*. The behaviors identified as a *security critical function* are then used to ensure that security goals have been produced to protect these actions. This enables goal verification and traceability. This helps guide the identification process and incorporates security principles into the identification process.

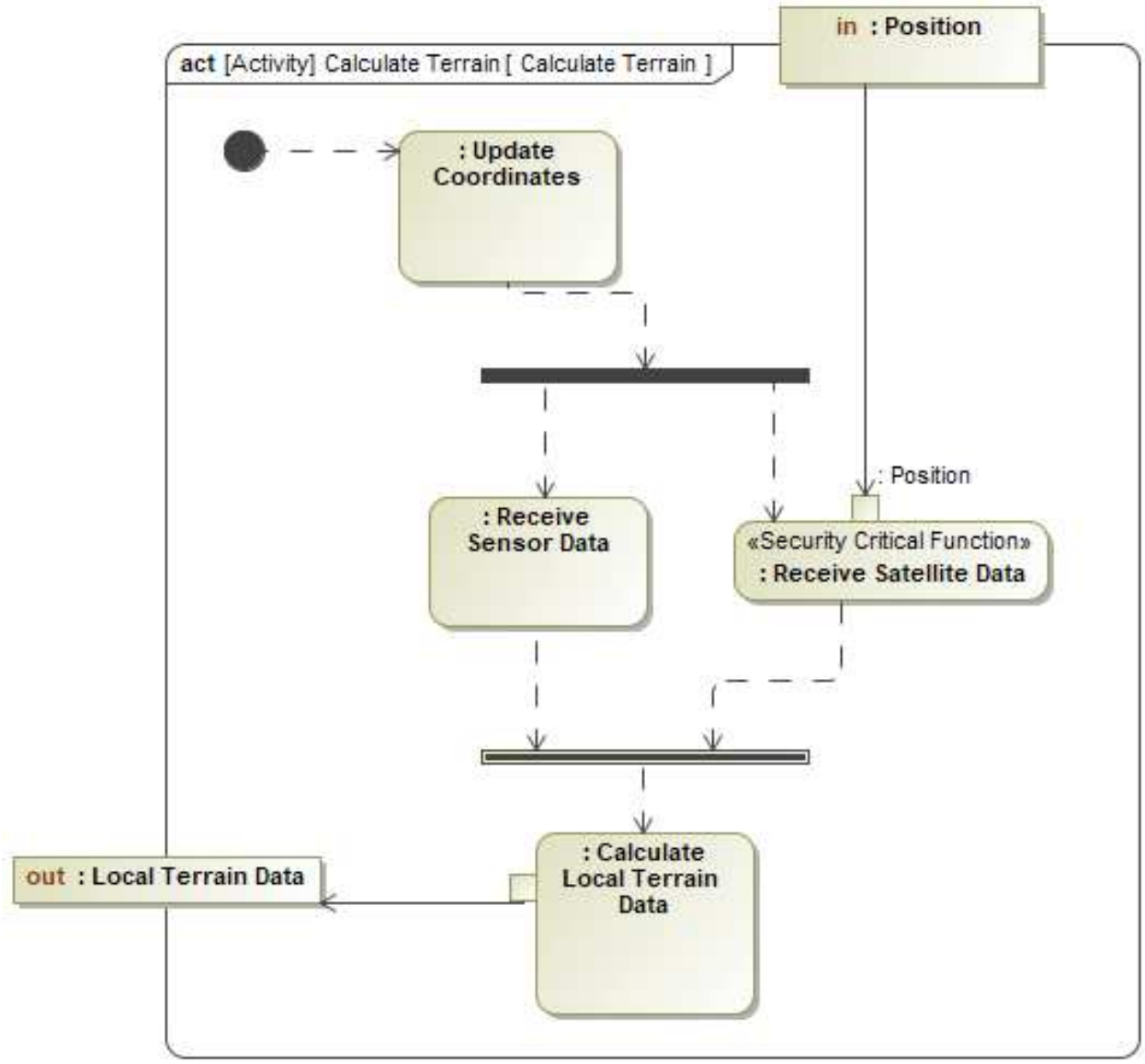

Figure 4.22: Security critical ID for 'Calculate Terrain' use case

An example for the Off-Road Navigation system is shown in Figure 4.22. In the case of this activity diagram, the Off-road navigation system is tasked with updating its coordinates, acquiring data from internal sensors, and satellites, and calculating the local terrain and elevation maps. Since receiving data from the satellite requires external communication, this activity was identified as a *security critical function*.

The objective of this step is not to enumerate all security critical functionality, but it is a brainstorming tool to highlight select functionality that is security critical in an effort to validate the security goals as described in the next step of this work.

4.2.8 Goal Verification

Once the behavioral analysis is complete, the initial security goals can be evaluated for their completeness and accuracy. This is done by developing allocation matrices, and cross-checking that all previously identified security-critical functions can be satisfied by at least one system goal. If gaps are identified, then new security goals are captured and assigned to system hazards and functions for traceability.

First, an allocation matrix relating security critical functionality to the current security goals is created. If each security critical function is not currently addressed by a security goal, then it is determined if a new security goal is likely needed to address the security of the system functionality.

Figure 4.23 is an activity diagram of the Off-road navigation system highlighting the key activities that the system is expected to accomplish as it obtains the user parameters before providing navigation. The activity titled *Validate Authenticity of External Connections* was identified as security critical functionality and did not have a corresponding security goal to address this security need. Therefore, a new security goal, *External Connection Authentication and Authorization* was defined as Security Goal number 11 and added in Figure 4.24 to ensure that all users are authenticated and authorized before use.

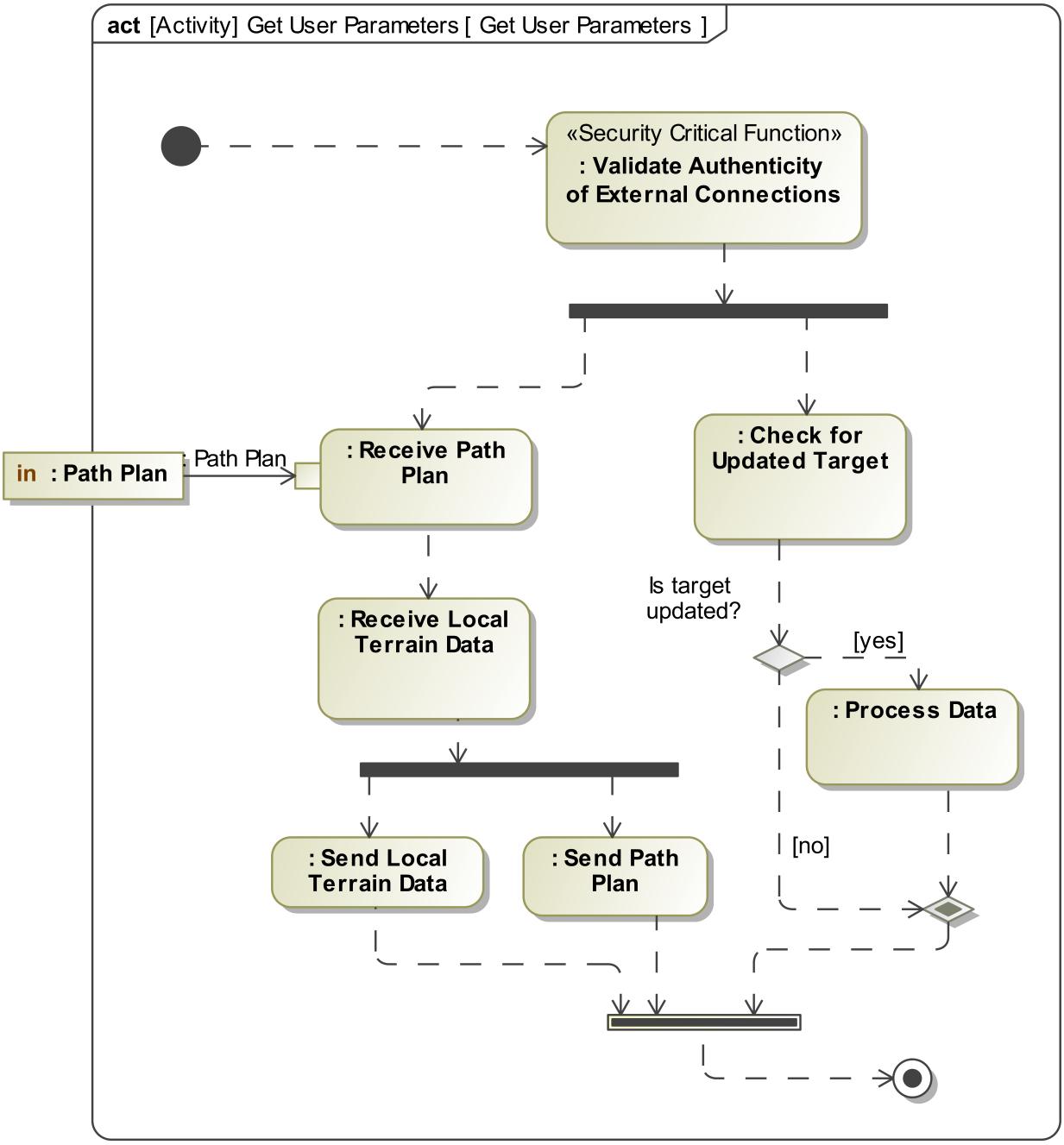

Figure 4.23: Security critical ID for 'Get User Parameters'

4.2.9 Output - Initial System Goals for Security

Figure 4.24 shows the resulting security goals elicited from the EGRESS method for an Off-Road Navigation System. These goals are system level goals elicited from a security lens. They

are NOT system requirements, instead they form a foundation in which specific measurable system requirements can be elicited as a part of the system design process using existing available best practice methods available in the community of practice.

#	Name	Text	Id	Hazard
1	1 Sensor input validation	The system should validate sensor inputs against available terrain maps	1	Misinterpretation of terrain Vehicle enters uncontrollable or unrecoverable state
2	2 Secure comms	The system should not transmit mission critical information over unsecure methods	2	Exposure of sensitive information under the wrong conditions
3	3 Redundant Sensors	The system should have redundant safety critical sensors	3	Degraded sensor functionality Vehicle enters uncontrollable or unrecoverable state
4	3.1 Critical Sensor ID	Critical sensors for navigation and terrain mapping must be identified	3.1	Degraded sensor functionality
5	4 Safe separation	The system should maintain minimum user defined separation limits	4	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Vehicle enters uncontrollable or unrecoverable state
6	4.1	The system should allow users to input minimum separation distances	4.1	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles
7	5 Selectable position sharing	The system should have selectable position sharing capabilities	5	Exposure of sensitive information under the wrong conditions
8	6 Display and warning	The system should display the potential hazards and obstacles along route	6	Vehicle enters uncontrollable or unrecoverable state Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles
9	6.1 Audible and visible warnings	The system should have both audible and visual warning indicators for hazards along route	6.1	Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Vehicle enters uncontrollable or unrecoverable state
10	7 Changes to destination	The user should be notified and accept any change to target destination	7	Vehicle enters uncontrollable or unrecoverable state
11	8 External Connection Authentication and Authorization	The system should validate the authenticity of external connections (Satellites, other vehicles, C2) and limit external inputs to authorized entities only (white listing, role based access control) - Derrived from reviewing security critical functionality in activity diagrams	8	Exposure of sensitive information under the wrong conditions Misinterpretation of terrain
12	9 User Authentication and Authorization	The system should implement role based access control to ensure only valid users can input system parameters.	9	Vehicle enters uncontrollable or unrecoverable state Vehicle does not maintain safe distance from vehicles, terrain, and other obstacles Exposure of sensitive information under the wrong conditions

Figure 4.24: Off-Road Navigation System Goals Table with Added Constraints

4.2.10 Traceability

A key benefit of the EGRESS approach is traceability of security goals back to stakeholder defined system objectives, captured in the form of unacceptable losses and system hazardous states. MBSE tools enable this traceability mapping as shown in Figure 4.25. This figure shows system requirements mapped to hazardous states.

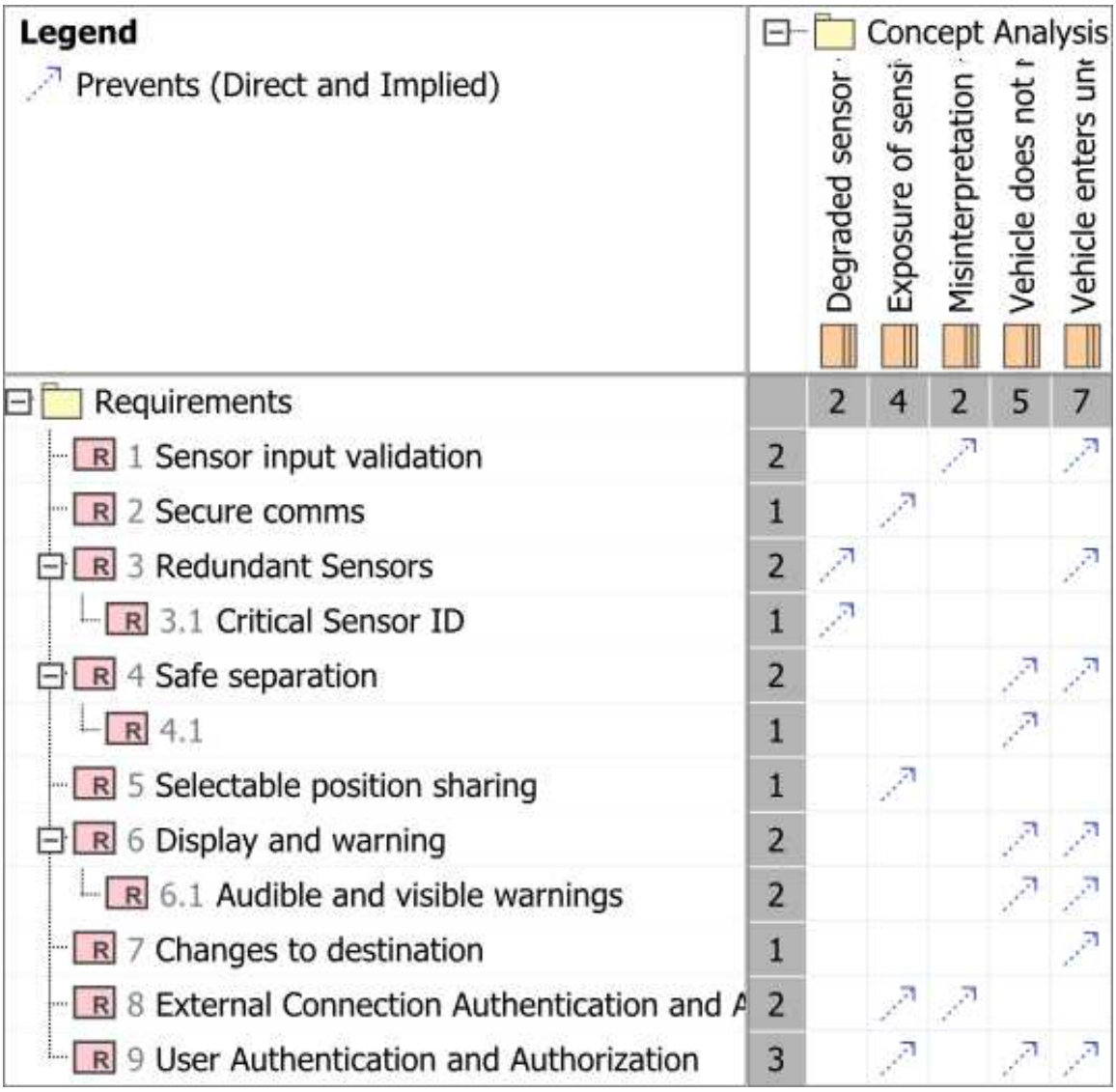

Figure 4.25: Off-Road Navigation System Traceability: Goals to Hazards

Traceability empowers decision-makers with clear information on how a security goal is required to ensure required system functionality in adverse operating conditions. This presents strong evidence to combat the mindset of security as an inconvenience or an unnecessary cost. This becomes even more effective as future specific security requirements are elicited. They can then be mapped back to the system security goal which is mapped to hazards and losses, thus empowering the design engineers with clear rationale for answering the question of *why* the security feature or control is required.

4.3 Discussion of Results

EGRESS is a new methodology leveraging best practices across MBSE, STPA, and security engineering disciplines. Current SysML elements and available cybersecurity extensions of the language do not adequately support RAAML. This section discusses takeaways for effectively utilizing EGRESS and modeling constructs that could be improved for further integration into existing modeling languages, packages, and techniques. Even in light of these challenges, the value the model provides in traceability to the system design team for security goals as the foundation of future security requirements is of high value and presents a strong argument for implementing EGRESS early in complex system design efforts.

4.3.1 EGRESS and RAAML Modeling Stereotypes

To support the incorporation of the EGRESS method into a graphical modeling language, a thorough exploration of suitable model stereotypes was conducted in an effort to align with existing standards. The primary implementation of EGRESS was integrated into the standard SysML language using the RAAML extension. In addition to SysML, the following extended stereotypes were used in the modeling effort:

- Core RAAML
  - Situation - describes a set of situation occurrences of some type. Hazards are a type of situation.
  - RelevantTo - the relationship is used to link situations to system model elements to provide context and relevance for the Situation
- STPA
  - Loss Scenario - a sequence of situations which lead to losses
- GSN
  - Goal - presents a claim forming part of the argument

- Strategy - describes the nature of the inference that exists between a goal and its supporting goal(s)

Integrating RAAML elements into the model provides a more complete and cohesive understanding of the SoI, enhancing the collaboration and transparency of security-related goals. Rather than extend the SysML profile with customized stereotypes, the first version of EGRESS used for the STMS example leverages profiles that have been accepted by the general SE community. Published profiles clearly define stereotypes, while customization is subject to interpretation. For example, the Unified Modeling Language (UML) *Focus* stereotype was used in lieu of the desired term *Mission* found in STPA. *Focus* identifies a class that defines the core logic or control flow of supporting entities.

Unfortunately, this meant certain stereotypes were a forced fit and did not exactly describe the intended system element in EGRESS. For instance, in Step 1, the purpose and goal of the system is not a best fit with *Focus, and Strategy* as available stereotypes. For the second iteration of EGRESS applied to the Off Road Navigation System, custom stereotypes were implemented for improved clarity. While this limits integration in traditional system modeling profiles/efforts, custom stereotypes were deemed necessary for the clarity and utility of the EGRESS method.

4.3.2 EGRESS Custom Stereotypes

In order to support the clearest implementation of the EGRESS method into a systems modeling language, custom stereotypes were deemed necessary. While attempts to minimize the amount of custom stereotypes were made, Figure 4.26 and Figure 4.27 summarizes the stereotypes that were implemented into five of the diagrams for the system model. The implementation of these stereotypes facilitates the clearest integration of the EGRESS method into the standard SysML modeling language. They are crucial for capturing the security-specific nuances and concepts of EGRESS, which ensures that the resulting model is highly representative of the methods' objective; eliciting early system security goals. Furthermore, the use of custom stereotypes enables a more streamlined and coherent modeling process, enhancing the effectiveness of brainstorming and

stakeholder engagement, and ultimately contributing to the method's effectiveness in addressing the unique characteristics of the targeted system of interest.

EGRESS Diagram	Custom Stereotype	SysML Element
System Purpose and Goal Diagram	- Goal - Purpose - Method - *By means of* - *In order to*	- Block
Loses and Hazards Diagram	- Hazard - Loss	- Block - *Allocate*
Requirement table (SysML Standard)	- Hazard - *Prevent*	- Requirement
Misuse Case Diagram	- Confidentiality Attack - Integrity Attack - Availibility Attack - Attacker - *Exploits*	- Use Case - Association - Include - Extend - Generalization
Activity Diagrams (SysML Standard)	- Security Critical Function	- Initial Node - Activity Final - Action - Control Flow - Decision and Merge - Fork and Join - Swimlanes

Figure 4.26: Stereotypes & Elements Used in Proposed EGRESS Diagrams

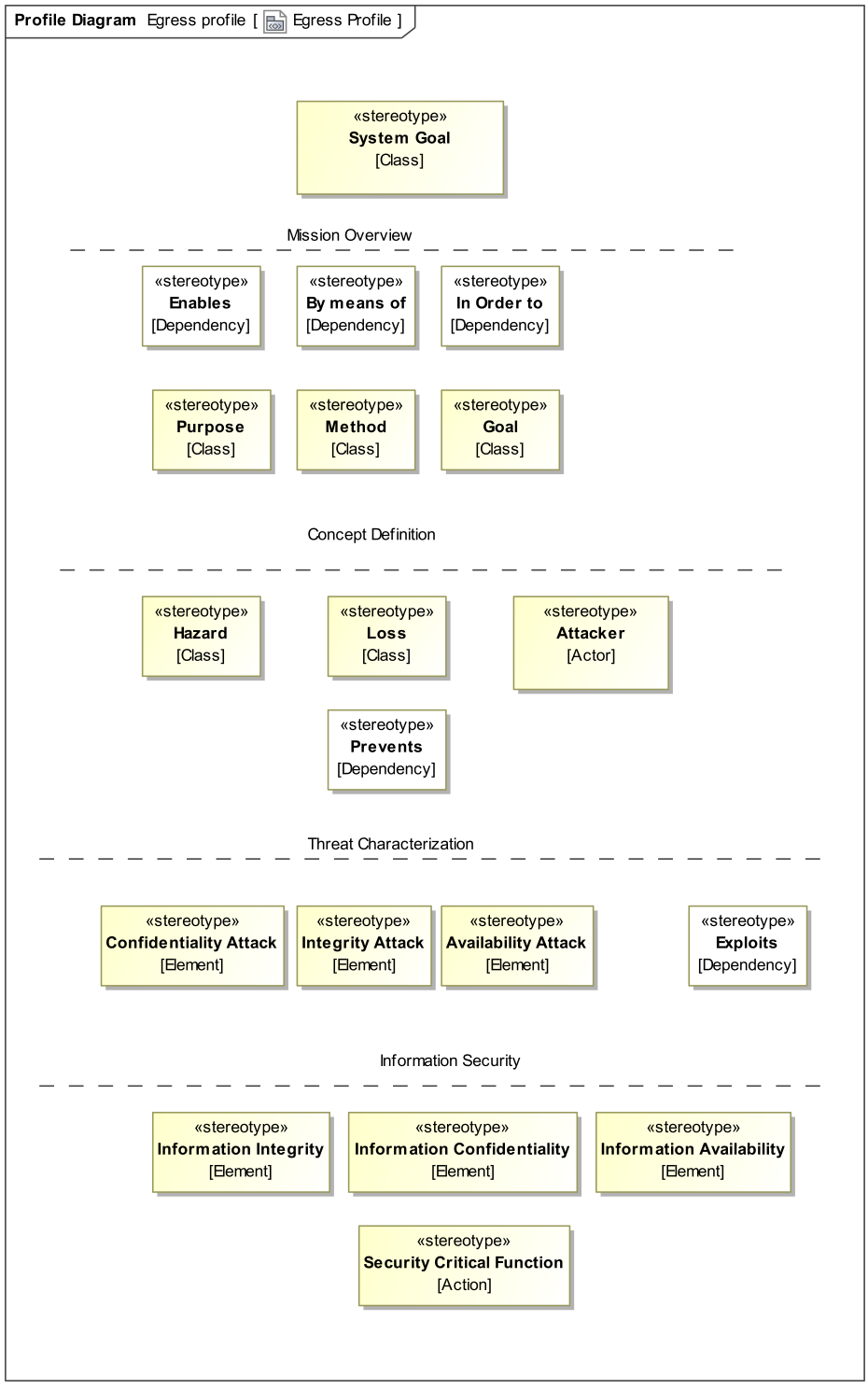

Figure 4.27: EGRESS Profile in Cameo

These custom stereotypes highlight areas for improvement in the existing implementation of the RAAML standard. The only allowable relationship according to the RAAML specification between losses and hazard is *relevant to*. *Relevant to* is a very nondescript relationship, much better represented with *causes* or *leads to*. Additionally, classifying losses and hazards as *situations* is another nondescript model element that could be further clarified for increasing understanding. Future iterations of RAAML will hopefully incorporate updates to the available stereotypes to improve the clarity of elements and their relationships.

4.3.3 Popper Falsification Approach Discussion:

Popper falsification [18], introduced in Section 1.4, is particularly useful for this research hypothesis as data is not available to induce increased system security over an operational lifecycle, as it shifts the focus from verification to potential refutation. This approach aligns with the scientific method's fundamental aspect of testing the null hypothesis [19].

The key advantages of applying Popper's falsification to this research hypothesis are:

1. Logical strength: Falsification uses deductive reasoning.

2. Overcoming the problem of induction: While it's impossible to test every possible condition, design study, or application, finding a single experiment that does not support the hypothesis can falsify the hypothesis. This addresses the limitation of not having comprehensive data to prove this hypothesis positively.

3. Provisional knowledge: Falsificationism acknowledges that scientific knowledge is subject to revision based on new evidence. The two detailed examples of applying EGRESS acknowledge the need for iterative and continuous testing, and the method itself was iterated and revised as applied to the different systems of interest.

4. Critical approach: Popper's method emphasizes the importance of critical discussion, continuous assessment, and peer review in evaluating this hypothesis.

By applying Popper's falsification principle, this work tests the hypothesis of "Secure system design can be improved through early system security goal elicitation." It advances the systems engineering body of knowledge, even where the challenge of lifecycle data limitations prevent positive proof. It relies on a more critical and robust scientific process, focusing on potential refutation rather than statistical confirmation. This is further supported by the peer-reviewed publication process in which experts in the field of system engineering and system security did not refute the claims of utility to the secure system design challenge. Additionally, the EGRESS method was presented to the leading minds in System Security Engineering (SSE) through the INCOSE SSE working group. They did not refute its utility, and in fact, sought co-authorship on similar work based on similar tenets and contributions [87].

The ability to inductively demonstrate that system design is more secure would be beyond the scope of this work as it would require a system to be designed using the primary methodology introduced in this work, EGRESS (Eliciting Goals for Requirement Engineering of Secure Systems), with metrics collected over the system's operational lifecycle proving it is more secure than similar systems designed without the elicitation of early security goals. The objective data to prove the reduced security vulnerability of the system would require many years of data collection against the progressively evolving security threats to prove objectively the value of EGRESS. While the data over a system lifecycle would be extremely valuable to prove this hypothesis, it is beyond the scope of this dissertation effort. Additionally, security threats are always evolving and the most vexing and important of those are unknown until implemented, like Zero-Day attacks. Instead, this work presents evidence that the hypothesis of EGRESS improving secure system design cannot be falsified.

The EGRESS method does emphasize iteration and human subject matter experts and engineers are in that iteration loop. Therefore, it is possible for those humans to make poor decisions or potentially subvert the process and make systems less secure. However, this is not an indictment of the EGRESS method, as the same threat actors can influence a design regardless of the method.

Research question two presents the hypothesis that secure system design can be improved through a method to elicit system-level goals for security before a preliminary architecture is available. The new EGRESS methodology introduced here provides a way to improve secure system design in early system design where a system is ill-defined, and an architecture is not yet available. Its utility is demonstrated as applied to Space Traffic Management and an Off-road Navigation system concept. The popper falsification approach is used in providing evidence of the inability to falsify the hypothesis that EGRESS's utilization to elicit system security goals improves secure system design.

Chapter 5

Conclusions

5.1 Summary

This work makes several novel contributions to the body of knowledge for systems engineering, primarily in the area of security by design for Cyber-Physical Systems (CPS). More specifically, contributions utilize Systems Thinking and Model-Based Systems Engineering (MBSE) to address organizational and technical challenges for secure system design in the early stages of CPS development. In answering research question one, a systematic analysis using iceberg models, causal loop diagrams, and SysML block definition diagrams identifies and proposes considerations to address the root causes of weak security design within CPS design teams. Extending this contribution in answering research question two, it introduces EGRESS, a new methodology for security goal elicitation in the conceptual design phase of a CPS, filling a critical gap in current systems engineering practices by providing a structured, traceable approach to defining security goals before a system architecture is established. This research also explores the application of the RAAML standard in security-focused CPS design, identifying areas where it can be refined for cybersecurity applications. Furthermore, this work highlights the importance of a holistic, systems-based approach to CPS security, offering recommendations on team composition, essential skills, and lifecycle considerations to enhance security integration from the outset. These contributions collectively support the development of more resilient and secure CPS architectures, while aligning with the broader goal of making security as fundamental to system design as functionality and safety.

5.1.1 Research Contribution Summary

This dissertation provides novel contributions centered around two primary research questions:

1. What recommendations can improve Cyber-Physical System (CPS) Design Teams with respect to Security?

2. Hypothesis: Secure system design can be improved through early system security goal elicitation

The investigation into research question one (RQ1) concluded that while training individuals in Systems Thinking and security practices is beneficial, it is not a universal solution for improving CPS design with respect to security. Systems Thinking tools, such as iceberg models, causal loop diagrams, and SysML block definition diagrams, provide insights into the underlying challenges faced by CPS design teams, including organizational structures and mental models that contribute to weak security practices. The analysis reveals that beyond training, a more fundamental shift in how security is integrated into CPS design processes is required. This finding led to Research Question Two (RQ2), which answers the need for a structured method to capture CPS security goals early in system design, providing a more systematic and universally implementable solution to increasing security by design.

RQ2 is based on the hypothesis that secure system design can be significantly improved through a formalized method for eliciting security goals at the system level before a preliminary architecture is available. This research identifies a critical gap in existing systems engineering methodologies, where security considerations are often addressed too late in the design process, leading to less secure designs and increased cost and schedule impacts. To address this, the EGRESS method was developed, providing a structured approach for eliciting security goals with traceability to stakeholder needs and system requirements. By integrating best practices from Systems Thinking principles, loss-driven engineering analysis, and leveraging MBSE, EGRESS ensures that security is considered as a foundational design element rather than an afterthought. This research also highlights the balance between utilizing common versus custom SysML profiles and languages, as well as the distinction between traditional system requirements and security-focused system goals, ensuring a more effective approach to secure CPS development from the earliest stages.

Contributions

This dissertation provides novel contributions to enhance the field of systems engineering. Specifically, it contributes to the efforts introduced above in enabling 'Security by Design' for complex CPS's.

1. (RQ1) Analysis utilizing systems thinking tools on the current state of Cyber-Physical System Design Teams - Chapter 2.

2. (RQ1) Recommendations, supported by systems thinking principles and tools, to improve CPS Design and Design Teams with respect to Security - Chapter 2.

3. (RQ2) Definition of a security goal in the context of stakeholder needs, system goals, and system requirements - Section 1.5.1.

4. (RQ2) Upon conducting an extensive literature review of research on requirements engineering, there is a notable scarcity of resources addressing security requirements from a systems engineering perspective. A consolidated reference for systems engineers integrating insights from both software centric security engineering and traditional systems engineering is a contribution to the SE Body of Knowledge, [17].

5. (RQ2) Consolidated and clear method to generate system goals with a focus on security for Cyber-Physical Systems in conceptual design - Chapter 3.

6. (RQ2) Recommendations of how to balance utilizing common versus custom SysML diagrams and languages, and traditional system requirements versus system goals within the security goal elicitation method - Section 4.3.2

7. (RQ2) Evidences of the security goal elicitation method, EGRESS, applied to complex CPS design - Chapter 4.

Popper Falsification Approach

This work utilizes Popper's falsification principle to evaluate the hypothesis that secure system design can be improved through early security goal elicitation. Given the lack of long-term operational data for a system utilizing EGRESS in its design to provide evidence of increased security over a system's lifecycle, falsification provides a rigorous alternative by focusing on potential refutation rather than statistical verification. This approach ensures logical strength through deductive reasoning, addresses the problem of induction by recognizing that a single contradictory instance can disprove a hypothesis, and acknowledges the iterative nature of scientific knowledge. The research demonstrates that the EGRESS methodology is not yet falsified, supporting its validity in improving secure system design. Evidence to support the hypothesis is provided by peer-review evaluations and expert discussions with SE and SSE experts, where its utility was endorsed. While deductive evidence of enhanced security would require extensive lifecycle data collection against evolving threats, EGRESS is demonstrated as a viable and valuable approach through its application to Space Traffic Management and Off-road Navigation system concepts. This advances the systems engineering body of knowledge by providing a systematic method for integrating security into early-stage system design, using a falsification-based scientific framework to substantiate its contributions.

5.2 Future Work

Future work should focus on refining and validating the proposed methodologies through practical implementation, real-world testing, and iterative improvement. Developing an executable Causal Loop Diagram (CLD) and applying and validating it within a CPS design team would provide empirical data on its effectiveness, challenges, and areas for enhancement. Documenting such an implementation would offer valuable case studies that could further justify investment in Systems Thinking and Systems Security Engineering training for CPS design teams. Further comparative exploration of security requirements elicitation methods would provide value to the body of knowledge, with an emphasis on assessing their relative effectiveness in different CPS contexts.

Another important avenue of future research involves formalizing the representation of "System Goals" within Model-Based Systems Engineering (MBSE). Currently, SysML does not differentiate between system goals and requirements, using the broad stereotype "requirement" for both. Introducing a distinct classification for system goals could improve clarity and traceability in system design, not just for security applications but for broader engineering use cases. Additionally, the emerging RAAML (Risk Analysis and Assessment Modeling Language) standard, while demonstrating potential for security applications, is still in its early stages. Further investigation into its practical applicability for cybersecurity is necessary, including potential revisions or extensions to better accommodate security-focused system modeling.

Beyond these technical advancements, future research should also explore the long-term impact of the EGRESS methodology on secure system design. While this work demonstrates the utility of early security goal elicitation, longitudinal studies are needed to assess its effectiveness in reducing security vulnerabilities over a system's operational lifecycle. Given the evolving nature of security threats, including unknown vulnerabilities and zero-day attacks, ongoing validation of the EGRESS approach through industry adoption and evaluations will be essential. Furthermore, collaboration with industry practitioners and standards organizations could help refine and standardize the approach into MBSE languages and tools, facilitating wider adoption across CPS domains.

Bibliography

[1] "ISO/SAE 21434: Road Vehicles - Cybersecurity Engineering - SAE International." [Online]. Available: https://www.sae.org/standards/content/iso/sae21434/

[2] J. Aviation, "JOBY Aviation," 2022. [Online]. Available: https://www.jobyaviation.com/

[3] H. Sohier, S. Guermazi, M. Yagoubi, P. Lamothe, A. Maddaloni, P. Menegazzi, and Y. Huang, "A tooled methodology for the system architect's needs in simulation with autonomous driving application," 2019, pp. 1–8.

[4] D. Wagner, "Building More Resilient Cybersecurity Solutions for Infrastructure Systems," in *Systems Engineering in the Fourth Industrial Revolution*. John Wiley & Sons, Ltd, 2019, pp. 415–443. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119513957.ch16

[5] T. M. Chen and S. Abu-Nimeh, "Lessons from Stuxnet," *Computer*, vol. 44, no. 4, pp. 91–93, 2011.

[6] Claroty Team 82, "Claroty Biannual ICS Risk and Vulnerability Report 2021," https://claroty.com/resources/reports/2h-2021, 2021.

[7] "Weapon Systems Cybersecurity: Guidance Would Help DOD Programs Better Communicate Requirements to Contractors," Tech. Rep. GAO-21-179, Mar. 2021. [Online]. Available: https://www.gao.gov/assets/gao-21-179.pdf

[8] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," p. 6, 2011.

[9] K. Baldwin, P. R. Popick, J. F. Miller, and J. Goodnight, "The United States Department of Defense revitalization of system security engineering through program protection," in *2012 IEEE International Systems Conference SysCon 2012*, Mar. 2012, pp. 1–7.

[10] Department of Defense, "DoD Instruction 5000.83 Technology and Program Protection to Maintain a Technological Advantage," 2020.

[11] "Defense Standardization Program (DSP): Standardization Directory," 2024. [Online]. Available: https://www.dsp.dla.mil/

[12] D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and T. M. Shortell, *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, 4th ed. John Wiley & Sons, Inc, Jul. 2015.

[13] R. Ross, M. Winstead, and M. McEvilley, "Engineering Trustworthy Secure Systems," Tech. Rep. NIST Special Publication (SP) 800-160 Vol. 1 Rev. 1, Nov. 2022. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/160/v1/r1/final

[14] "Vehicle Cybersecurity: Control the Code, Control the Road," 2020. [Online]. Available: https://www.vehicledynamicsinternational.com/features/vehicle-cybersecurity-control-the-code-control-the-road.html

[15] "Automotive ISAC." [Online]. Available: https://automotiveisac.com

[16] J. Daily and M. Span, "A model for cybersecurity education through challenge events," in *INCOSE International Symposium*, vol. 34, no. 1. Wiley Online Library, 2024, pp. 944–957.

[17] M. T. Span, G. Salinger, M. Rayno, and J. Daily, "Security requirements engineering: A survey for the systems engineer," in *2024 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2024, pp. 1–8.

[18] K. Popper, *The Logic of Scientific Discovery*, 2002nd ed. London: Routledge, 1959.

[19] S. Mitra, "An analysis of the falsification criterion of karl popper: A critical review," 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:214706762

[20] "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering," *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, 2018.

[21] "INCOSE Guide to Writing Requirements." [Online]. Available: https://www.incose.org/inet/working-groups/requirements

[22] H. Dezfuli, A. Benjamin, C. Everett, M. Feather, P. Rutledge, D. Sen, and R. Youngblood, "NASA System Safety Handbook. Volume 2: System Safety Concepts, Guidelines, and Implementation Examples," May 2015. [Online]. Available: https://ntrs.nasa.gov/citations/20150015500

[23] J. P. Monat and T. F. Gannon, "What is systems thinking? a review of selected literature plus recommendations," *American Journal of Systems Science*, vol. 4, no. 1, pp. 11–26, 2015.

[24] A. Pyster, D. Olwell, N. Hutchison, S. Enck, J. Anthony, D. Henry, and A. Squires, *Guide to the Systems Engineering Body of Knowledge (SEBoK) version 1.0*. Hoboken, NJ: The Trustees of the Stevens Institute of Technology, 2012.

[25] Senge, Peter, *The Fifth Discipline: The Art and Practice of the Learning Organization*. USA: Random House Books, 2006.

[26] D. H. Meadows, *Thinking in systems: A primer*. London, United Kingdom: Chelsea Green Publishing, 2008.

[27] D. Gurdur and M. Törngren, "Design Thinking and Systems Thinking for Cyber-Physical Systems," in *DS 91: Proceedings of NordDesign 2018, Linköping, Sweden, 14th - 17th August 2018*, 2018.

[28] INCOSE, "MBSE Initiative," 2022. [Online]. Available: https://www.incose.org/incose-member-resources/working-groups/transformational/mbse-initiative

[29] K. Shahroudi, *Systems Thinking for Real World Applications*, 2022.

[30] Shahroudi, Kamran et. all, *Turbocharge Your Future in a Complex World with Systems Thinking*. Springer Nature, 2025.

[31] F. Elakramine, R. Jaradat, N. Ullah Ibne Hossain, M. Banghart, C. Kerr, and S. El Amrani, "Applying systems modeling language in an aviation maintenance system," *IEEE Transactions on Engineering Management*, pp. 1–13, 2021.

[32] M. Hause, J. Hummell, and F. Grelier, "Mbse driven iot for smarter cities," in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, 2018, pp. 365–371.

[33] M. Nagahi, N. U. I. Hossain, R. Jaradat, and S. Grogan, "Moderation effect of managerial experience on the level of systems-thinking skills," 2019, pp. 1–5.

[34] R. K. Jonkers and K. Eftekhari Shahroudi, "A design change, knowledge, and project management flight simulator for product and project success," *IEEE Systems Journal*, vol. 15, no. 1, pp. 1130–1139, 2021.

[35] "Computer Security Technology Planning Study," Tech. Rep., 1972. [Online]. Available: https://apps.dtic.mil/sti/citations/AD0758206

[36] "Systems Engineering Vision 2035." [Online]. Available: https://www.incose.org/home-1/systems-engineering-vision-2035

[37] M. Reed and M. Winstead, "Secure cyber resilient engineering (scre) practice," 2023.

[38] R. Chatterjee, S. Mukherjee, and J. Daily, "Exploiting Transport Protocol Vulnerabilities in SAE J1939 Networks," in *Proceedings Inaugural International Symposium on Vehicle Security & Privacy*. San Diego, CA, USA: Internet Society, 2023. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2023/02/vehiclesec2023-23053-paper.pdf

[39] W. Young and N. Leveson, "An integrated approach to safety and security based on systems theory," *Communications of the ACM*, vol. 57, no. 2, pp. 31–35, 2014.

[40] Salinger, Gabe and Span, Trae and Chatterjee, Rik and Jepson, Jake and Daily, Jeremy, "A demonstration of mbsesec applied to securing cyber-physical system communications," in *2024 IEEE Aerospace Conference*. IEEE, 2024, pp. 1–13.

[41] "ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes," May 2015. [Online]. Available: https://www.iso.org/standard/63711.html

[42] B. Fabian, S. Gürses, M. Heisel, T. Santen, and H. Schmidt, "A comparison of security requirements engineering methods," *Requirements Engineering*, vol. 15, no. 1, pp. 7–40, Mar. 2010. [Online]. Available: http://link.springer.com/10.1007/s00766-009-0092-x

[43] "EVITA." [Online]. Available: https://www.evita-project.org/index.html

[44] A. Lautenbach, M. Almgren, and T. Olovsson, "Proposing HEAVENS 2.0 – an automotive risk assessment model," in *Computer Science in Cars Symposium*. Ingolstadt Germany: ACM, Nov. 2021, pp. 1–12. [Online]. Available: https://dl.acm.org/doi/10.1145/3488904.3493378

[45] T. UcedaVelez and M. M. Morana, *Intro to Pasta*, 2015, pp. 317–342.

[46] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "Stride-based threat modeling for cyber-physical systems," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. Torino: IEEE, Sep. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8260283/

[47] M. Deshmukh, "Security requirements engineering process," in *Seminar in Information System, Security Engineering*. Citeseer, 2009.

[48] S. u. Rehman, C. Allgaier, and V. Gruhn, "Security Requirements Engineering: A Framework for Cyber-Physical Systems," in *2018 International Conference on Frontiers of Information Technology (FIT)*, Dec. 2018, pp. 315–320, iSSN: 2334-3141.

[49] Mead, Nancy R and Hough, Eric D and Ii, Theodore R Stehney, "Security Quality Requirements Engineering (SQUARE) Methodology."

[50] D. Firesmith, "Engineering Security Requirements." *The Journal of Object Technology*, vol. 2, no. 1, p. 53, 2003. [Online]. Available: http://www.jot.fm/contents/issue_2003_01/column6.html

[51] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security Requirements Engineering: A Framework for Representation and Analysis," *IEEE Transactions on Software Engineering*, vol. 34, no. 1, pp. 133–153, Jan. 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4359475/

[52] A. Shostack, *Threat Modeling: Designing for Security*. Hoboken, New Jersey: John Wiley & Sons, 2014.

[53] Span, Martin Trae and Mailloux, Logan and Daily, Jeremy, "Considerations for cyber-physical design teams tasked with engineering safe and secure systems for a notional electrified aircraft concept," in *2023 IEEE International Systems Conference (SysCon)*. IEEE, 2023, pp. 1–8.

[54] N. Leveson, *Engineering a Safer World*. The MIT Press, Cambridge, Mass, Jan. 2012.

[55] N. Leveson and J. Thomas, *STPA Handbook*, Mar. 2018. [Online]. Available: http://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

[56] M. T. Span, L. O. Mailloux, M. R. Grimaila, and W. B. Young, "A systems security approach for requirements analysis of complex cyber-physical systems," in *2018 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE, 2018, pp. 1–8.

[57] Span, Martin "Trae" and Mailloux, Logan O and Grimaila, Michael R, "Cybersecurity architectural analysis for complex cyber-physical systems," *The Cyber Defense Review*, vol. 3, no. 2, pp. 115–134, 2018.

[58] M. Span, L. O. Mailloux, R. F. Mills, and W. Young, "Conceptual systems security requirements analysis: Aerial refueling case study," *IEEE Access*, vol. 6, pp. 46 668–46 682, 2018.

[59] Mailloux, Logan O and Span, Martin and Mills, Robert F and Young, William, "A top down approach for eliciting systems security requirements for a notional autonomous space system," in *2019 IEEE International Systems Conference (SysCon)*. IEEE, 2019, pp. 1–7.

[60] Sayers, Joshua M. and Feighery, Brynn E. and Span, Martin Trae, "A stpa-sec case study: Eliciting early security requirements for a small unmanned aerial system," in *2020 IEEE Systems Security Symposium (SSS)*, 2020, pp. 1–8.

[61] R. T. Reule, B. Feighery, M. W. Winstead, D. R. Hild, W. Barnum, and M. Span, "STPA-Sec analysis for DevSecOps reference design," in *INCOSE International Symposium*, vol. 31, no. 1. Wiley Online Library, 2021, pp. 296–309.

[62] K. X. Campo, T. Teper, C. E. Eaton, A. M. Shipman, G. Bhatia, and B. Mesmer, "Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature," *Systems Engineering*, vol. 26, no. 1, pp. 104–129, 2023, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21644. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sys.21644

[63] L. Delligatti, *SysML Distilled: a brief guide to the systems modeling language*. Addison-Wesley, 2014.

[64] P. De Saqui-Sannes, R. A. Vingerhoeds, C. Garion, and X. Thirioux, "A Taxonomy of MBSE Approaches by Languages, Tools and Methods," *IEEE Access*, vol. 10, pp. 120 936–120 950, 2022, conference Name: IEEE Access.

[65] M. J. Vinarcik and B. Pepper, "Modeling safety and cybersecurity controls in sysml," National Defense Industrial Association, 2016.

[66] "Cameo safety and reliability analyzer," 2024. [Online]. Available: https://docs.nomagic.com/display/CSRA2024x/Systems+Cybersecurity+Designer

[67] A. Reichwein and C. Paredis, "Overview of Architecture Frameworks and Modeling Languages for Model-Based Systems Engineering," *Proceedings of the ASME Design Engineering Technical Conference*, vol. 2, Jan. 2011.

[68] D. Mažeika and R. Butleris, "MBSEsec: Model-Based Systems Engineering Method for Creating Secure Systems," *Applied Sciences*, vol. 10, no. 7, p. 2574, Apr. 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/7/2574

[69] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in *UML 2002 — The Unified Modeling Language*, G. Goos, J. Hartmanis, J. van Leeuwen, J.-M. Jézéquel, H. Hussmann, and S. Cook, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, vol. 2460, pp. 412–425, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/3-540-45800-X_32

[70] T. Lodderstedt, D. Basin, and J. Doser, "Secureuml: A uml-based modeling language for model-driven security," in *International Conference on the Unified Modeling Language*. Springer, 2002, pp. 426–441.

[71] Y. Roudier and L. Apvrille, "SysML-Sec: A model driven approach for designing safe and secure systems," in *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, Feb 2015, pp. 655–664.

[72] C. Schmittner, Z. Ma, E. Schoitsch, and T. Gruber, "A Case Study of FMVEA and CHASSIS as Safety and Security Co-Analysis Method for Automotive Cyber-physical Systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. Singapore Republic of Singapore: ACM, Apr. 2015, pp. 69–80. [Online]. Available: https://dl.acm.org/doi/10.1145/2732198.2732204

[73] M. T. Span, S. Gowdanakatte, J. Daily, I. Ray, and K. E. Shahroudi, "Systems thinking and model based systems engineering's utility to solve complex organizational problems-cyber-

physical system design teams," in *2022 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE, 2022, pp. 1–8.

[74] Security Boulevard, "Cyber attacks on the power grid," *https://securityboulevard.com/2022/05/cyber-attacks-on-the-power-grid/*, 2022.

[75] C. Beaman, A. Barkworth, T. D. Akande, S. Hakak, and M. K. Khan, "Ransomware: Recent advances, analysis, challenges and future research directions," *Computers & Security*, vol. 111, p. 102490, Dec. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016740482100314X

[76] B. S. Bloom *et al.*, *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain*. New York: David McKay Company, 1956.

[77] S. Khou, L. O. Mailloux, J. Pecarina, and M. Mcevilley, "A customizable framework for prioritizing systems security engineering processes, activities, and tasks," *IEEE Access*, vol. 5, pp. 12 878–12 894, 05 2017.

[78] R. Dove, K. Willett, T. McDermott, H. Dunlap, D. P. MacNamara, and C. Ocker, "Security in the Future of Systems Engineering (FuSE), a Roadmap of Foundation Concepts," *INCOSE International Symposium*, vol. 31, no. 1, pp. 175–194, 2021. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.2021.00832.x

[79] R. Dove, M. Winstead, H. Dunlap, M. Hause, A. Scalco, K. Willett, A. D. Williams, and B. Wilson, "Democratizing Systems Security," *INCOSE International Symposium*, vol. 33, no. 1, pp. 86–97, 2023. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/iis2.13010

[80] S. Friedenthal and C. Oster, *Architecting spacecraft with SysML: A Model-based Systems Engineering Approach*. CreateSpace Independent Publishing Platform, 2017.

[81] "Risk Analysis and Assessment Modeling Language (RAAML) Libraries and Profiles, v1.0." [Online]. Available: https://www.omg.org/spec/RAAML/1.0/About-RAAML

[82] T. Kelly and R. Weaver, "The goal structuring notation – a safety argument notation," in *Dependable systems and networks workshop on assurance cases*, 2004, p. 6.

[83] W. Young and N. G. Leveson, "An integrated approach to safety and security based on systems theory," *Communications of the ACM*, vol. 57, no. 2, pp. 31–35, 2014.

[84] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*, 2nd ed. Waltham, MA: Morgan Kaufmann, 2012, oCLC: ocn754518532.

[85] G. Hurlburt, "'Good Enough' Security: The Best We Will Ever Have," *Computer*, vol. 49, no. 07, pp. 98–101, jul 2016.

[86] D. Mažeika and R. Butleris, "Integrating Security Requirements Engineering into MBSE: Profile and Guidelines," *Security and Communication Networks*, vol. 2020, pp. 1–12, Mar. 2020. [Online]. Available: https://www.hindawi.com/journals/scn/2020/5137625/

[87] B. Papke, R. Kratzke, M. T. Span, and N. N. Shevchenko, "Enabling fuse security objectives by leveraging cyber survivability methods," *INCOSE International Symposium*, vol. 34, no. 1, pp. 72–89, 2024. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/iis2.13133