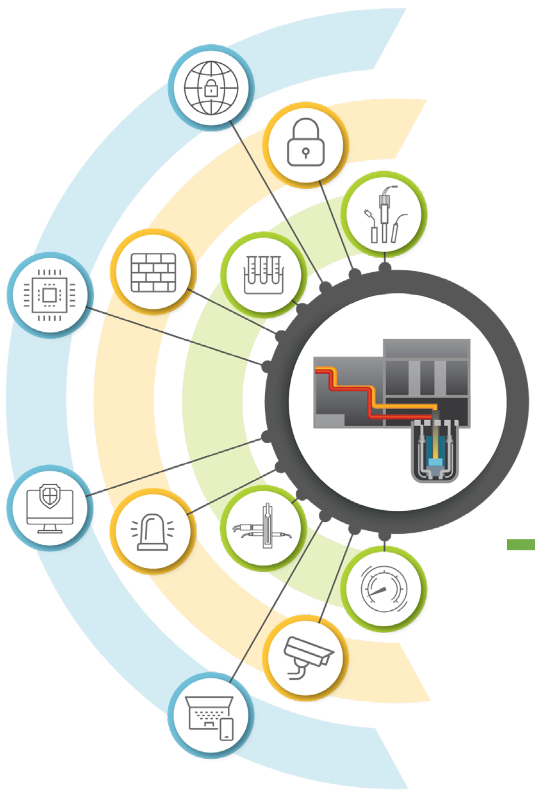


Advanced Reactor Safeguards and Security

FY25 Security Inclusive Model-Based Systems Engineering Project

Prepared for
U.S. Department of Energy

Shannon L. Eggers, Sonali Sinha Roy,
Kevin M. O'Rear, Ross D. Hays

Idaho National Laboratory

September 2025
INL/RPT-25-87232, Rev. 0

INL/RPT-25-87232, Rev. 0

Advanced Reactor Safeguards and Security

FY25 Security Inclusive Model-Based Systems Engineering Project

Shannon L. Eggers, Sonali Sinha Roy, Kevin M. O'Rear, Ross D. Hays

Idaho National Laboratory

September 2025

Idaho National Laboratory
Advanced Reactor Safeguards and Security
Idaho Falls, ID 83415

https://energy.sandia.gov/arss

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Page intentionally left blank

EXECUTIVE SUMMARY

The Security-Inclusive Model-Based Systems Engineering (MBSE) project explores the integration of Cyber-Informed Engineering (CIE) and System-Theoretic Process Analysis (STPA) within an MBSE ecosystem for the design of the Microreactor Applications Research Validation and Evaluation (MARVEL) microreactor. MARVEL is a small-scale nuclear tested using Stirling engines to convert thermal energy to electricity in the Power Generation System (PGS). The engines can be operated manually or via the human-machine interface in the control room via signals sent through a network to the engine control units (ECUs).

STPA is a structured approach to hazard analysis that identifies unsafe control actions (UCAs) and safety requirements early in the systems engineering lifecycle, addressing both intentional and unintentional digital threats. In this project, STPA was used to evaluate the control structure of the ECU and it's controlling action "Engine STOP." Six different UCAs were identified that could potentially result in the following impacts: engine damage, reactor trip, or an overcooling event leading to thermal shock of reactor components. Focusing on three UCAs, potential loss scenarios leading to these UCAs were also developed. The scenarios were organized into ECU behavior, feedback, control pathway, and controlled process categories in alignment with the STPA method. CIE emphasizes the use of both engineering design and traditional information assurance activities to eliminate or reduce digital risk in operational technology. Once the loss scenarios were identified, risk treatments were proposed, both from an engineering and information assurance perspective.

The Innsolate MBSE tool was evaluated for its capability to integrate with MathWorks Simulink to provide a platform for integrating requirements, risk management, and modeling and simulation. It was found that the necessary capabilities for this integration were unavailable or unsupported. As a result, the focus of the research shifted to using the MathWorks platform as a holistic MBSE ecosystem. A PGS model was developed in System Composer, including network architecture and information flow pathways. The physical attributes of the PGS and ECU were developed in Simulink and were included in the System Composer architecture model.

Future work will evaluate the identified risk treatments using an engineering design analysis approach within the MathWorks MBSE ecosystem. Future work will also focus on expanding this ecosystem to include additional components of the MARVEL reactor, facilitating comprehensive system evaluation and collaboration with related projects, such as the DOE-ARSS remote monitoring and ARCADE projects. The goal is to develop a cohesive digital engineering environment that supports ongoing risk management and system optimization. The project aims to systematically integrate CIE, STPA, and MBSE to mitigate digital risks associated with MARVEL's digital systems by identifying UCAs and implementing engineering design changes and cybersecurity measures. This integration is proposed to improve overall system safety, performance, and resilience.

Page intentionally left blank

CONTENTS

EXECUTIVE SUMMARY v
ACRONYMS x
1 INTRODUCTION 12
2 BACKGROUND 14
2.1 MARVEL Microreactor 14
2.1.1 Overview 14
2.1.2 Power Generation System (PGS) 15
2.2 Innoslate MBSE Tool 17
2.3 MathWorks MBSE Ecosystem 18
3 METHODOLOGY 18
3.1 Innoslate Evaluation 18
3.1.1 STPA and CIE integration 18
3.1.2 Risk management integration 19
3.1.3 Modeling and simulation integration 19
3.2 MathWorks Evaluation 19
3.2.1 Requirements integration 19
3.2.2 Simulink and System Composer modeling 19
4 RESULTS 20
4.1 Innoslate Evaluation 20
4.1.1 STPA and CIE integration 20
4.1.2 Risk management integration 28
4.1.3 Modeling and simulation integration 29
4.2 MathWorks Evaluation 30
4.2.1 Requirements integration 30
4.2.2 System architecture in System Composer 30
4.2.3 Physical system modeling in Simulink 30
5 DISCUSSION AND FUTURE WORK 33
6 CONCLUSION 34
7 REFERENCES 34

FIGURES

Figure 1. Visual representation of how both engineering design and cybersecurity measures work together to lower digital risk [2]. 12
Figure 2. Examples of different control structures [3]. 13
Figure 3. MARVEL systems and interfaces. 14

Figure 4. (a) Stirling engine. (b) MARVEL primary coolant system and secondary coolant system flow [6]. (c) Installed power converter in MARVEL [6]. 15

Figure 5. Simplified piping and instrumentation diagram for the engine control unit and cooling loops. 16

Figure 6. ECU local user interface panel. 17

Figure 7. Simplified network architecture diagram for the ECUs. 17

Figure 8. STPA decision flow chart for UCA, hazards, and losses for the control action “ECU Engine Stop.” 21

Figure 9. Entity-relationship class diagram for modeling digital risk 28

Figure 10. Innoslate scripting feature for MATLAB integration. 29

Figure 12. Simulink model for the PGS showing the subsystems. 31

Figure 13. ECU control logic. 32

Figure 14. PGS and ECU HMI setup. 32

TABLES

Table 1. UCA’s for the controlling action “Engine STOP” for the ECU. 20

Table 2. Potential scenarios for UCA1-N-1, engine runs without adequate coolant flow leading to engine damage. 21

Table 3. Potential scenarios for UCA1-N-2, engine continues to run after reactor trip leading to a PCB overcooling event 24

Table 4. Potential scenarios for UCA1-P-1, engine stopped when reactor is running leading to loss of reactor cooling and reactor trip on high temperature. 26

Page intentionally left blank

ACRONYMS

A2D	analog to digital
API	application programming interface
ARSS	Advanced Reactor Safeguards and Security
CA	controlling action
CAN	control area network
CIA	central insurance absorber
CIE	Cyber-Informed Engineering
CS	control system
DOE	U.S. Department of Energy
DOORS	Dynamic Object-Oriented Requirements System
ECU	engine control unit
HMI	human-machine interface
ICS	MARVEL instrument and control system
ICT	information and communications technology
INL	Idaho National Laboratory
LUI	local user interface
MARVEL	Microreactor Applications Research Validation and Evaluation
MBSE	model-based systems engineering
OT	operational technology
PGS	power generation system
PLC	programmable logic controller
R&D	research and development
RCS	reactivity control system
RPS	reactor protection system
SSC	structure, system, component
STPA	System-Theoretic Process Analysis
SysML	systems modeling language
TREAT	Transient Reactor Test facility
UCA	unsafe control action
V&V	verification and validation

Page intentionally left blank

FY25 Security Inclusive Model-Based Systems Engineering Project

1 INTRODUCTION

Cyber-Informed Engineering (CIE) is an approach for integrating cybersecurity into engineering practices to ensure that critical infrastructure is designed with intrinsic risk treatments to enhance the security and reliability of digital operational systems (OT) and industrial control systems against cyber threats [1]. While cybersecurity-by-design approaches highlight the need to consider cybersecurity early in the systems engineering lifecycle, a primary focus of CIE is to use good engineering design practices in addition to traditional cybersecurity practices to eliminate or reduce digital risk in OT. As illustrated in Figure 1, combining the ability to lower the consequence threshold, or severity of physical impact, through traditional engineering design (e.g., functional assurance) with the ability to lower the exposure threshold, or likelihood of digital compromise, through traditional information and communications technology (ICT) cybersecurity measures (e.g., information assurance) shifts potential loss scenarios from digital threats into a more acceptable region, [2].

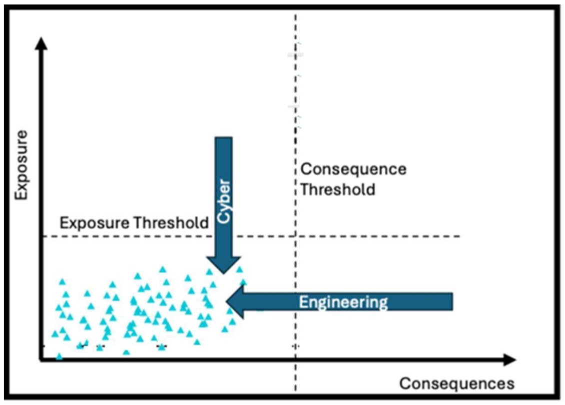


Figure 1. Visual representation of how both engineering design and cybersecurity measures work together to lower digital risk [2].

System-Theoretic Process Analysis (STPA) is a hazard analysis technique based on an extended model of causation that assumes accidents can be caused by both component failures and unsafe interactions of system components [3]. STPA is a proactive analysis method that can be used early in the systems engineering lifecycle to assist in identifying safety requirements and constraints enabling elimination or control of hazards during design. The stepwise process with STPA includes defining the purpose of the analysis, modeling the control structure, identifying unsafe control actions (UCA), and identifying loss scenarios. As the examples demonstrate in Figure 2, a control structure captures the functional relationships and interactions by modeling the system as a set of feedback control loops in order to identify how and why unsafe control actions may occur [3]. While many hazard analysis approaches exist, STPA is a very structured process that is focused on identifying a comprehensive set of unsafe control actions that may lead to hazards regardless of the cause (e.g., failure, degradation, cyber attack).

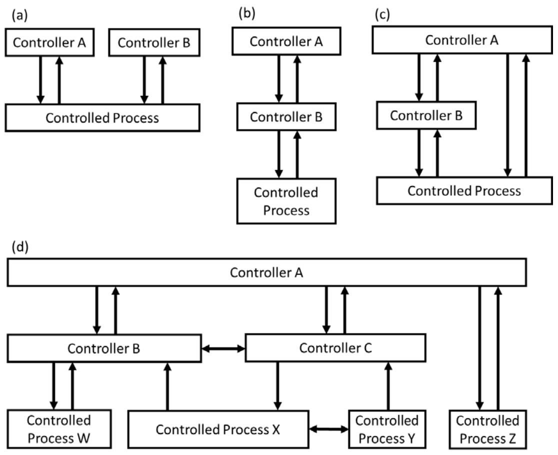


Figure 2. Examples of different control structures [3].

Additionally, many cyber risk analysis methods are focused on defining the likelihood of attack scenarios in which a threat exploits a vulnerability leading to an adverse consequence. These methods often emphasize the need to define adversary attributes and *attack* scenarios, including the techniques, tactics, and procedures an adversary might use to compromise a system. However, both STPA and CIE focus more holistically on digital risk (which includes both adversarial and non-adversarial threats) and mitigating *hazard* scenarios. The goal in using STPA and CIE is to control what an intentional or unintentional action can or cannot do, as opposed to attempting to enumerate the innumerable pathways an adversary could use to achieve their mission.

Model-based systems engineering (MBSE) is an integrated approach that uses digital models to support the specification, design, analysis, verification, and validation of engineered systems throughout their systems engineering lifecycle. Providing a visual representation of the boundaries, context, and behavior of interconnected systems enables better communication, consistency, and management of complex system engineering activities. The intent is to ensure there is a single source of truth, or one central repository of accurate, verified, and up-to-date information, that all stakeholders can reliably use throughout the lifecycle [4]. MBSE is traditionally used to develop, manage, and trace requirements (e.g., safety, performance, reliability, regulatory) from conceptual design through final verification and validation (V&V) to ensure acceptable integration and compliance with requirements.

STPA and CIE are complementary approaches that, when integrated within an MBSE ecosystem, can improve the safety, performance, security, reliability, and resilience of complex digital system designs. STPA can identify UCAs and hazards leading to mission losses, CIE can identify opportunities to eliminate or mitigate them, and the MBSE ecosystem can be used to evaluate and test design decisions. Thus, the purpose of this research project is to demonstrate a process for integrating all three of these concepts (e.g., CIE, STPA, and MBSE) in the design of a microreactor. Specifically, the research question is: Can an MBSE ecosystem be used to integrate the STPA process of identifying UCAs and hazards in digital systems along with the decision analysis and V&V associated in the use of CIE and good engineering practices to eliminate or mitigate this digital risk.

2 BACKGROUND

2.1 MARVEL Microreactor

2.1.1 Overview

The Microreactor Applications Research Validation and Evaluation (MARVEL) microreactor is a small nuclear applications testbed designed for research and development purposes at Idaho National Laboratory (INL). MARVEL is intended to be an advanced reactor consisting of a small uranium zirconium hydride fueled microreactor cooled by natural convection of sodium-potassium alloy [5]. Reactivity will be controlled by control drums and a central control rod, with thermal energy converted to electricity using four Stirling engines. For this report, the 90% MARVEL design is used [6]

MARVEL systems and interfaces are illustrated in Figure 3. The primary purpose of the fuel and core system is to create and sustain nuclear fission chain reaction, transfer heat to fuel cladding, and support decay heat removal. The primary purpose of the MARVEL reactor structure is to remove heat from the core, transfer heat to end user systems, and transfer decay heat to the ultimate heat sink [6]. The MARVEL power generation system (PGS) contains two subsystems, the engine cooling subsystem and the electrical production subsystem. The electrical production subsystem contains Stirling engines that absorb high-grade heat from the secondary coolant subsystem in the MARVEL reactor structure which itself absorbs heat from the primary coolant subsystem [6].

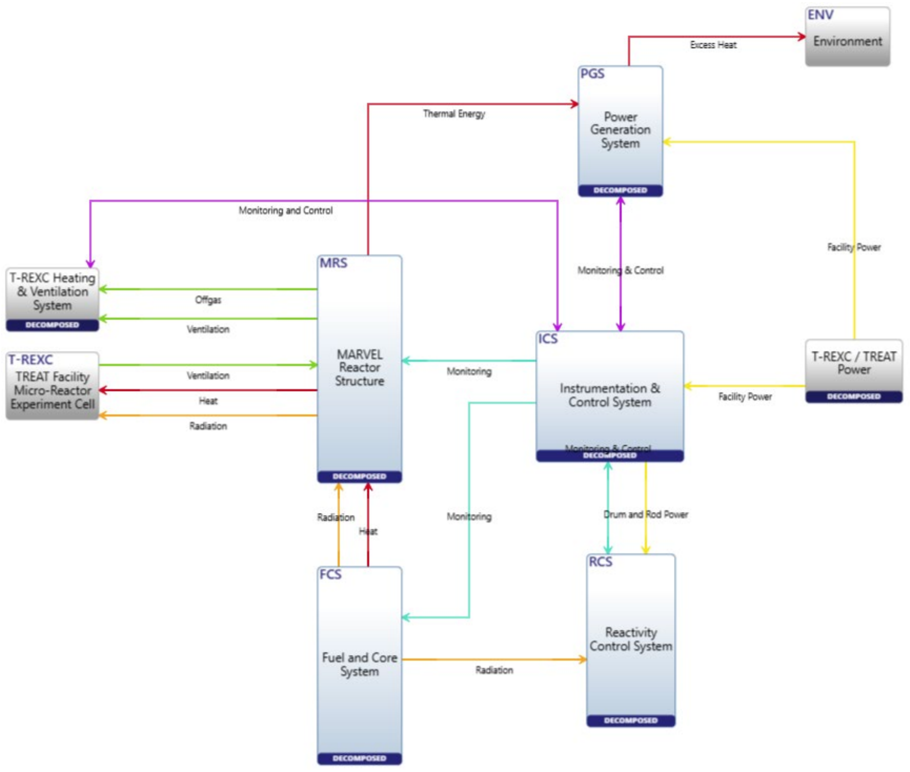


Figure 3. MARVEL systems and interfaces.

The reactivity control system (RCS) provides reactivity control during normal operation and controlled shutdown as well as reactor trip upon signal from the reactor protection subsystem (RPS) in response to abnormal or postulated design basis accidents [6]. The RCS consists of four rotatable control drums that are evenly distributed about the core periphery and one translatable central insurance absorber (CIA) rod in the center of the core [6]. The drum forcing subsystem receives a position request from the instrument and control system (ICS) control subsystem (CS) to move a control drum and/or the CIA rod.

Additionally, upon a trip signal from the RPS, the control drum clutch and CIA electromagnet are deenergized to rotate the control drums to their shutdown position and drop the CIA rod to its shutdown position [6].

The MARVEL ICS contains the CS, reactor instrumentation subsystem, RPS, and human machine interface (HMI). The reactor instrumentation subsystem measures critical operating parameters which are then sent to the CS. The RPS initiates a reactor shutdown upon signals from seismic sensors, manual trip button, loss of power, or shutdown from the RCS [6]. The CS also receives information from multiple subsystems for monitoring and control functions. CS data is sent to the HMI for operator monitoring. Operators can also interface with the HMI to send control requests to the CS [6]. The interlocks ensure that only one control drum or the CIA rod is moved at a time.

2.1.2 Power Generation System (PGS)

To convert reactor heat to electricity, the secondary coolant system removes fission heat from the primary coolant system by transferring heat through an intermediate heat exchanger to the Stirling engines (Figure 4a) through natural convection (Figure 4b). The Stirling engine's lower tubes are suspended partially in the secondary coolant (Figure 4c). The engines are free piston engines which transfer heat from the secondary coolant system to inert working gas (e.g., helium) in the engine which then expands and contracts in a frictionless, linear manner that can be directly converted into electricity or exhausted as thermal energy.

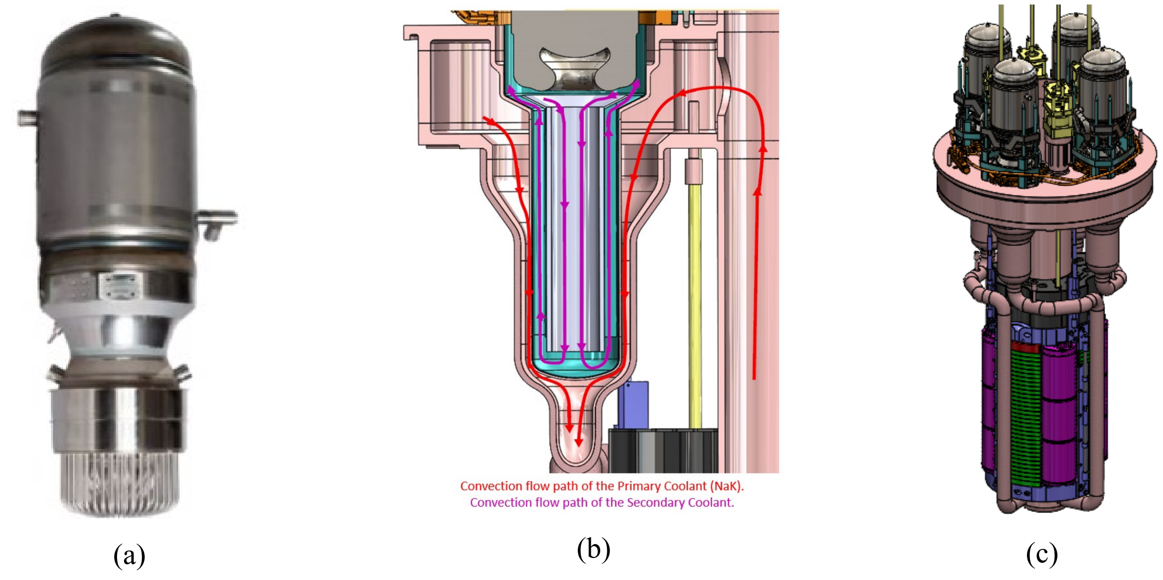

Convection flow path of the Primary Coolant (NaK).
Convection flow path of the Secondary Coolant.
(a)
(b)
(c)

Figure 4. (a) Stirling engine[a]. (b) MARVEL primary coolant system and secondary coolant system flow [6]. (c) Installed power converter in MARVEL [6].

Figure 5 illustrates the heat rejection system for the PGS that consists of two loops – (i) a primary chilled water inner loop that directly cools the engine and (ii) an air-cooled secondary outdoor outer loop which uses a glycol solution as coolant for freeze protection. The engine control unit (ECU) connected to the Stirling engine enables remote monitoring and control by the ECU through an internal control area network (CAN) bus interface. A maintenance laptop can be connected to the ECU via a CAN bus cable for configuration. The ECU also integrates an onboard MODBUS TCP/IP interface and discrete signals that are connected to a local area network which enables control of the ECU by Operators via an HMI in the control room. The ECU controls the coolant flows in both loops and the air flow for the secondary

[a] Image from Qnergy.

loop by regulating the pump and fan motor speeds though pulse wave modulation. Operators can also adjust power output by changing user load settings. The ECU uses an integrated analog vortex flow and temperature sensor for run-safe protection. The flow temperature sensor is connected to the ECU via a four-pin connector with pin 1 as the temperature signal and pin 2 as the flow signal. The voltage range is 0 to 3.5 V relative to the ground pin 3. The ECU performs an analog to digital (A2D) conversion of the flow signal to liters per minute and temperature signal to degrees Celsius. These digital values are available on the ECU CAN message list.

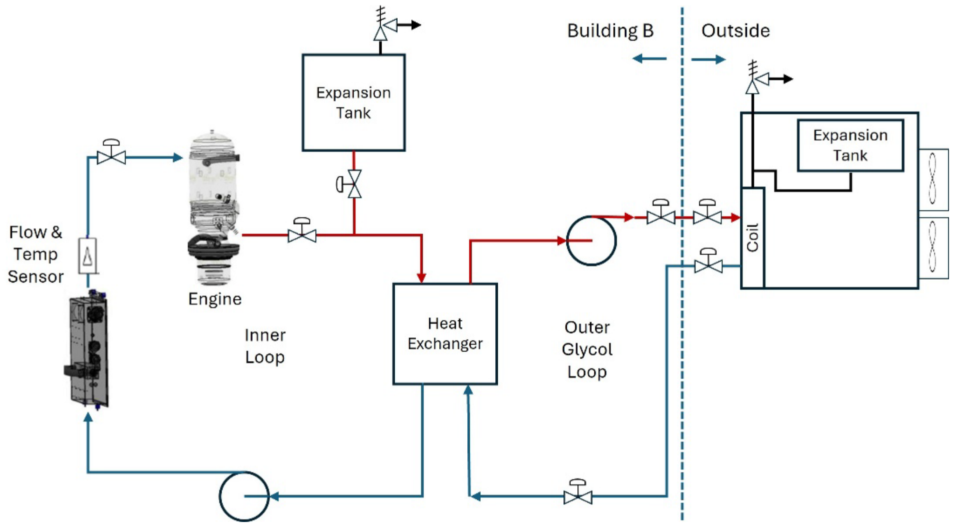


Figure 5. Simplified piping and instrumentation diagram for the engine control unit and cooling loops.

Procedures for the operation of the MARVEL reactor have not yet been formalized; however certain aspects can be approximated using available documentation. Initial reactor heat-up to hot standby conditions may be accomplished either via electric cartridge heaters in the primary coolant or through heat from nuclear fission. During normal reactor startup, operators raise reactor reactivity via the HMI until it is critical at which time the reactor is heated up to hot standby conditions (approximately 200°C). Coolant flow through the Stirling engines will be started and maintained as necessary to ensure that the coolant water remains below boiling and that the maximum external engine temperature remains below 70ºC; this also ensures the necessary temperature differential exists across the Stirling engine to permit operation.

Once the reactor is at hot standby, or at the desired temperature, Operators will ‘bump’ the engine by selecting engine start on the HMI which sends a signal to the ECU to initiate the motion of the engine piston. Once started, motion continues until either the temperature differential drops too low or until sufficient braking action is provided by the ECU (on receipt of an “Engine STOP” signal). Assuming all constraints are met (e.g., minimum 20 L/min cooling flow, emergency stop inactive), the engine will start generating electricity once the minimum threshold temperature is met at the engine’s heat exchanger head. As Operators raise reactor power, the secondary coolant system temperature increases, which increases the engine helium temperature, causing the engine piston displacement to increase, thereby increasing electrical generation. The engines will produce electricity up to the Operator setpoint and divert the excess electricity to heat via a load bank in the ECU which rejects heat to the inner and outer heat loops (or an alternate user thermal load if connected). Upon reactor shutdown, the Operator will send an “Engine STOP” signal to the ECU, and the engine will stop once it reaches the thermal threshold and then return to standby mode.

In addition to the HMI interface to the ECU in the control room, the ECU also has a local user interface (LUI) and controls, as shown in Figure 6. A simplified network architecture for the ECU communication pathway to the control room HMI is shown in Figure 7. All four ECUs connect to the same programmable logic controller (PLC) which is then routed from Building B to the control room HMI in Building A.

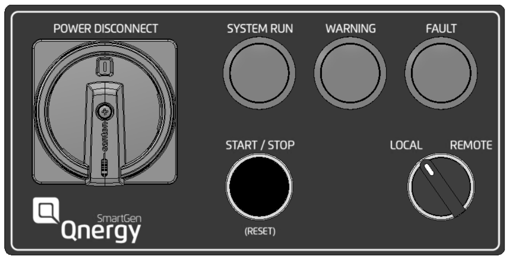


Figure 6. ECU local user interface panel.

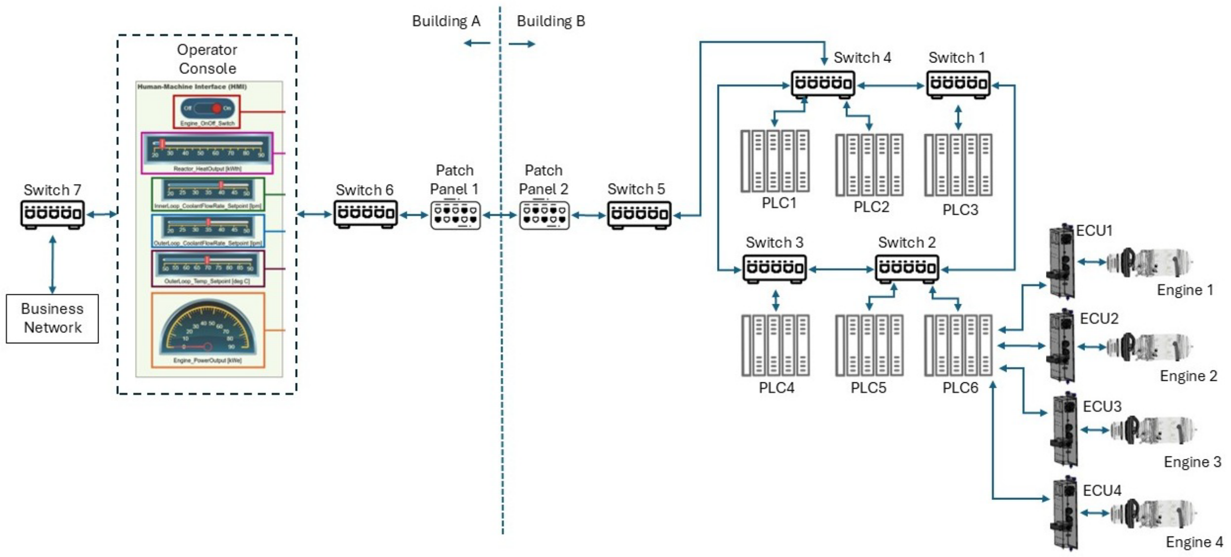


Figure 7. Simplified network architecture diagram for the ECUs.

2.2 Innoslate MBSE Tool

Innoslate is an MBSE tool developed by SPEC Innovations[b]. INL uses version 4.11.0.1 of Innoslate deployed on an INL server and accessed through a web browser. Innoslate is designed to streamline the systems engineering process by incorporating requirements management, traceability, and gap analysis using diagrams, documents, and reports. Innoslate is compliant with the system modeling language (SysML) and natively supports lifecycle modeling language. In addition to the ability to import and export SysML models in Extensible Markup Language (XML) format, Innsolate integrates with IBM DOORS, a requirements management tool, and supports import/export of comma-separated value files. Prior work updated a MARVEL Innsolate MBSE project to add a more complete set of asset diagrams and requirements, including digital requirements, with a focus on establishing an STPA process for the Stirling engine control within the framework.

[b] https://specinnovations.com/

2.3 MathWorks MBSE Ecosystem

Within the larger digital engineering ecosystem, modeling and simulation, including the use of digital twins, can be used to develop, evaluate, and optimize designs as well as verify and validate system requirements and performance to improve efficiency and reduce risk in complex engineering projects.

MathWorks' MATLAB is a programming language specifically designed for engineering mathematics. Simulink is a graphical tool that provides a block diagram environment for modeling and simulating system behavior over time and evaluating performance under different conditions. The MathWorks MBSE ecosystem includes several specialized libraries and packages, such as System Composer. System Composer provides tools for designing, analyzing, and optimizing system architecture and for connecting architecture models to Simulink-based behavioral models. It supports most of the functionalities of SysML within a Simulink environment and allows import of SysML models through the SysML Connector Product Support Package. The Requirements Toolbox can link with external requirements management applications (e.g., IBM DOORS) and can provide traceability throughout architecture development with System Composer and design with Simulink. An important advantage of the MathWorks suite of tools is the integration of MBSE functionalities with physical system model, which allows for a deeper, more integrated, and multi-faceted assessment.

3 METHODOLOGY

3.1 Innoslate Evaluation

3.1.1 STPA and CIE integration

Researchers have evaluated different approaches for integrating STPA analysis with SysML. Albrecht and Durak proposed an integrative approach that enables execution of STPA within SysML, including automation and validation through formalization [7]. Albrecht and Bertram also demonstrate how SysML-STPA can be used for safety trade-off analysis, including automatic generation of STPA hazards and UCAs to derive STPA requirements and aid decision analysis for architecture designs [8]. Additionally, de Souza, et al., proposes a method that combines STPA with SysML modeling activities, including simulation and formal verification of system models [9].

To accomplish this STPA analysis, the STPA evaluation from [10] was updated and expanded to include additional UCAs and scenarios. Example threats, both intentional and unintentional, were also developed for each scenario. Based on the UCAs and scenarios, potential modifications from engineering design and information assurance perspectives were proposed.

It was also hypothesized that STPA could be modeled within Innoslate to provide an automated methodology for performing STPA on the ECU and for providing traceability during design changes to identify if (and how) digital risk was eliminated or reduced. The following steps were established to test this hypothesis:

1. Define and model the ECU control structure as a block definition diagram.
2. Define and model the hazards, losses, and context variables as block definition diagrams using the standard STPA process.
3. Use the analysis block entries to identify the UCAs.
4. Identify relevant scenarios for each UCA and generate design requirements.
5. Identify options for reducing digital risk and mitigating the UCA using engineering design and IT solutions.
6. Compare design modifications and perform a decision analysis.

3.1.2 Risk management integration

In conjunction with the STPA analysis, it was hypothesized that a class diagram approach could be modeled within Innoslate to establish a traceable, programmatic method for defining and treating risk. The following steps were established to test this hypothesis in Innoslate:

1. Define the attributes and operations for each entity in the digital risk management entity-relationship diagram which was presented in [10].
2. Model each entity, or asset, as a class in the Innoslate class diagram.
3. Establish the proper relationship between each class.
4. Establish a method for automating the digital risk analysis of the ECU.

3.1.3 Modeling and simulation integration

It was hypothesized that MARVEL PGS and network architecture asset and action diagrams developed in Innoslate could be integrated with Simulink to provide comparisons on how engineering changes and security controls affect the safety, performance, and reliability of the ECU and its control of the Stirling engines. The following steps were established to test this hypothesis:

1. Consult with Spec Innovations to establish the capability to integrate Innoslate with Simulink.
2. Develop a simple example for proof of concept for communicating information between Innoslate and Simulink.
3. Expand the example to include the PGS system and network architecture.
4. Develop a Simulink model in accordance with the asset and action diagrams.
5. Evaluate how modifications in the diagrams are reflected in the Simulink model.

3.2 MathWorks Evaluation

3.2.1 Requirements integration

MathWorks provides a connector for the desktop version of IBM's Dynamic Object-Oriented Requirements System (DOORS) Next. This connector makes it possible to link Simulink models to the requirements held in DOORS. However, INL utilizes the browser-based version of DOORS, which would require creating a bespoke adapter utilizing the application programming interface (API) of both software. For this test case, requirements were simply provided to Simulink via an excel spreadsheet. In a full production environment, the API link would be desirable to maintain the traceability between the requirements management tool and the Simulink model.

3.2.2 Simulink and System Composer modeling

It was hypothesized that by modeling a system's hierarchic architecture (including its physical components, interfaces, control logic, signals, etc.) and simulating its operations, the input/output dependencies of digital signals can be studied. This can help in identifying the immediate as well as cascading impacts of UCAs, thereby revealing the system's vulnerability to digital threats. These vulnerabilities can then be mitigated through redesigning the system and/or adding safety or security measures and redundancies to make the system more safe and secure by design.

System Composer supports MBSE workflows within a MATLAB Simulink environment to support system architecture design and modular functional decomposition. It was hypothesized that with the seamless integration of Simulink and MATLAB elements with architecture models created in System Composer, greater flexibility and complexity could be achieved in the design and analysis tasks. Additionally, it offers greater interoperability with other sources of data compared to traditional MBSE tools. Steps taken using the 90% design for the MARVEL microreactor as the reference test case [6] include development of the hierarchical functional architecture of the MARVEL PGS and control components, followed by modeling the physical systems within PGS.

4 RESULTS

4.1 Innoslate Evaluation

4.1.1 STPA and CIE integration

Prior research identified several UCAs associated with the ECU [10]. Table 1 expands upon the previously identified UCAs for the controlling action (CA) “Engine STOP” (note that this control action is different than the “Emergency STOP” control action which is not modelled in this report). Damage to the Stirling engines could occur if an engine is run without a minimum of 20 L/min of coolant flow and a reactor trip could occur (or be required) if an engine is stopped while the reactor is at power. Additionally, a high consequence event is the potential for an overcooling event if the engines continue to run after a reactor trip. This event could impact primary coolant system boundary (PCB) structures, systems, and components (SSCs) by causing unacceptable thermal shock, violating ASME code. Thermal shock can lead to loss of integrity to SSCs, such as cracks, deformation, low-cycle fatigue, and structural failure. Depending on the damage from thermal shock, SSCs could fail, leading to fission product release and/or long-term reactor shutdown to replace damaged SSCs.

Table 1. UCA’s for the controlling action “Engine STOP” for the ECU.

CA ID	Controlling Action (CA)	Pathway	Required but not provided (N)	Provided but not required (P)	Required but wrong timing (T)	
					Too Early	Too Late
1	Engine STOP	HMI / ECU to engine	UCA1-N-1: STOP command not given when coolant flow lowers to less than 20 L/min Hazard: Engine runs without proper coolant flow. Impact: Engine damage. UCA1-N-2: STOP command not given after reactor is shutdown. Hazard: Overcooling of PCB SSCs. Impact: Thermal shock and loss of integrity of PCB. Potential for fission product release.	UCA1-P-1: STOP command given when reactor at power. Hazard: Thermal energy not removed from reactor. Impact: Reactor temperature increase leading to manual or automatic reactor trip.	UCA1-T-1: STOP command given before reactor lowers to specified power level. Hazard: Thermal energy not removed from reactor. Impact: Reactor temperature increase leading to manual or automatic reactor trip.	UCA-T-2: STOP command given too late after coolant flow lowers below 20 L/min. Hazard: Engine runs without proper coolant flow. Impact: Engine damage. UCA-T-3: STOP command given too late after the reactor is shutdown. Hazard: Overcooling of PCB SSCs. Impact: Thermal shock and loss of integrity of PCB. Potential for fission product release.

A visual representation of the UCAs, hazard types, and loss types identified for the control action "Engine STOP" is shown in Figure 8.

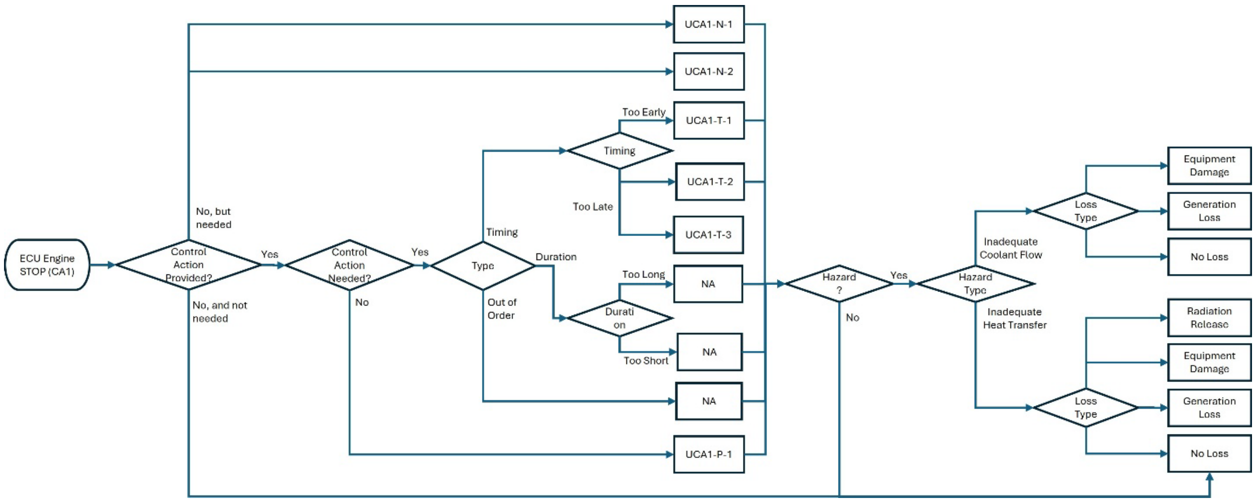


Figure 8. STPA decision flow chart for UCA, hazards, and losses for the control action "ECU Engine Stop."

4.1.1.1 *UCA1-N-1 scenarios and risk treatments*

Table 2 provides a listing of potential scenarios for UCA1-N-1 that could lead to an engine running without proper coolant flow. Table 3 identifies potential scenarios that could cause UCA1-N-2, an overcooling event. Table 4 provides a listing of potential scenarios for UCA1-P-1 that could cause a reactor trip on high temperature. Unintentional and intentional threat examples are included for each scenario, as applicable.

Table 2. Potential scenarios for UCA1-N-1, engine runs without adequate coolant flow leading to engine damage.

Category	Scenario	Threat Type Example	
		Unintentional[c]	Intentional
ECU Behavior	1. Design flaw in ECU	Inadequate design or programming of ECU	Supply chain attack, malware
	2. Misconfiguration of ECU	User incorrectly configures ECU	Intentional misconfiguration
	3. Degradation/failure of ECU	Component wear, poor environmental conditions, ECU hardware/firmware issue	
Feedback	4. Incorrect flow voltage sent to the ECU	Degradation/failure of the flow/temperature sensor, air in loop	
	5. Incorrect flow voltage received by the ECU	Wiring problem, noise, pin connection issue	

[c] Additional faults and possible causes are listed in the Qnergy PowerGen Installation and Operation Manual.

Category	Scenario	Threat Type Example	
		Unintentional[c]	Intentional
	6. Incorrect flow value derived by ECU	User incorrectly calibrates the sensor A2D in the ECU	Intentional sensor A2D miscalibration
	7. Communication between the flow sensor and ECU is unavailable (e.g., value not received by ECU)	Wiring problem, pin connection issue	
	8. Communication between the flow sensor and ECU is delayed	Wiring problem, noise, pin connection issue	
	9. Engine status is not sent to the ECU	Engine hardware/firmware issue	
	10. Engine status is not received by the ECU	Wiring problem, noise, user inaccurately sets the LUI to “Local”	Availability attack (e.g., denial of service), LUI intentionally set to “Local”
	11. Inaccurate engine status is sent to the ECU (e.g., engine is running, but the status sent is OFF)	Corruption on internal CAN bus, inadequate ECU programming	Integrity attack (e.g., spoofing, malware)
Control pathway	12. Engine STOP command sent but not received by engine	Wiring problem, corruption on internal CAN Bus	Availability attack (e.g., denial of service)
	13. Engine STOP command sent but received late	Wiring problem, corruption on internal CAN Bus	Availability attack (e.g., denial of service)
	14. Correct command sent but incorrect command received	Wiring problem, corruption on internal CAN Bus, race condition	Integrity attack (e.g., spoofing)
Controlled process	15. Conflicting engine commands	User incorrectly sets LUI to “Local” and Start/Stop button depressed	LUI intentionally set to “Local” and Start/Stop button depressed
	16. Engine does not stop after STOP command received	Engine hardware/firmware issue.	

CIE risk treatments include traditional engineering design changes that can eliminate or reduce the digital risk, including adding diverse and redundant SSCs and simplifying design by removing unnecessary features and shrinking the digital footprint (e.g., attack surface). From a traditional ICT perspective, risk treatments include implementing a defensive cybersecurity architecture, designing in security controls for defense in depth, and adding active defense features. The goal is to design SSCs that include safety, performance, resiliency, and security against all types of unintentional and intentional threats.

For UCA1-N-1, when the engine could run without adequate coolant flow leading to engine damage, the supplier intentionally added an analog vortex flow switch to the engine system for engine protection. The voltage signal is sent from the sensor to the ECU which then converts the analog signal to a digital flow value in liters per minute. This digital value is used in the ECU to automatically shut down the engine (or prevent startup) if flow is less than 20 L/min.

While this flow and temperature sensor is analog, the signal is converted into a digital signal for use in the ECU. There is a possibility that connection from the sensor to the ECU could have a wiring or pin connection issue, including noise on the line, that interferes with proper signal transmission. Additionally, the ECU could be improperly designed or misconfigured. And, either the sensor or ECU could degrade or fail, leading to incorrect process behavior.

With regards to communication between the ECU and engine, there could be wiring issues, noise, corruption on the internal CAN bus, a spoofing attack, a denial of service attack, or race conditions that could interfere with the engine receiving the STOP command. The engine or ECU could be degraded or have hardware or firmware issues leading to maloperation. And there is a possibility that the LUI is improperly operated, resulting in no commands sent or conflicting commands sent.

Referring to the scenarios listed in Table 2, engineering design and information assurance modifications could reduce the digital risk associated with the engine damage from operation without adequate cooling flow. Potential engineering design modifications and operational activities include:

- Improved design traceability to V&V requirements, including V&V throughout the systems engineering lifecycle (especially during implementation, and updates) to test the ability of the ECU to automatically stop the engine when coolant flow is less than 20 L/min. (Relevant for all scenarios)
- Add a diverse and redundant analog flow switch with an adjustable flow setpoint that is directly wired to the engine via a relay (e.g., bypassing the ECU). This provides additional engine protection in the event the ECU fails to stop the engine. While this does provide additional reliability and resilience, it also adds an additional failure mode that must be analyzed, and the flow setpoint must be set correctly and tested. It may be desired to set the setpoint for this switch lower than the ECU setpoint to avoid conflict scenarios while maintaining engine protection. (Relevant for all scenarios)
- Establish operator rounds (dependent on equipment accessibility) and a condition monitoring program to monitor the sensor, ECU, and engine operability for anomaly detection, diagnostics, and prognostics. (Relevant for scenarios 3, 4, 16)

Potential traditional information security controls include:

- Physical and technical security controls to limit access to the ECU. This would include requiring that the ECU cabinet be in a locked room with a locked cabinet that can only be accessed via the maintenance laptop through a controlled user account with password. The username and password should be kept in a controlled manner and only provided to authorized users. (Relevant for scenarios 2, 6, 11)
- The maintenance laptop should be kept in a secured and monitored location, such as required for maintenance and test equipment, with user check in and check out strictly controlled. The maintenance laptop should be scanned for viruses and malware prior to use. (Relevant for scenarios 2, 6, 11)
- The LUI panel should be physically secured, similar to the ECU. An additional physical security control is the addition of a key on the panel to limit local control to those who have access to the key. This key should be controlled under a key control program and only provided to users with access/use authority. (Relevant for scenarios 10, 15)
- Add a defensive security architecture that ensures the isolation of the CAN bus communication between the engine and the ECU. Extend this architecture to the entire process network to ensure that it is appropriately segmented into levels and zones with appropriate boundary devices to prevent inadvertent network flooding (e.g., broadcast storm) or intentional, malicious data availability attack (e.g., denial of service attack). Include continuous network and endpoint monitoring via a security event and information monitoring system or similar. (Relevant for scenarios 10, 11, 12, 13, 14)

4.1.1.2 UCA1-N-2 scenarios and risk treatments

Table 3. Potential scenarios for UCA1-N-2, engine continues to run after reactor trip leading to a PCB overcooling event

Category	Scenario	Threat Type Example	
		Unintentional	Intentional
ECU Behavior	1. Design flaw in ECU	Inadequate design or programming of ECU	Supply chain attack, malware
	2. Misconfiguration of ECU	User incorrectly configures ECU	Intentional misconfiguration
	3. Degradation/failure of ECU	Component wear, poor environmental conditions, ECU hardware/firmware issue	
Feedback	4. Reactor trip/engine shutdown signal is not sent to the ECU	Inadequate programming of reactor trip interface to ECU	Malware
	5. Reactor trip/engine shutdown signal is not received by the ECU	Wiring, network problem	Availability attack (e.g., denial of service)
	6. Inaccurate reactor trip/engine shutdown status sent to the ECU	Corruption along communication pathway	Integrity attack (e.g., spoofing)
	7. Engine status is not sent to the ECU	Engine hardware/firmware issue	
	8. Engine status is not received by the ECU	Wiring problem, noise, user inaccurately sets the LUI to “Local”	Availability attack (e.g., denial of service), LUI intentionally set to “Local”
	9. Inaccurate engine status is sent to the ECU (e.g., engine is running, but the status sent is OFF)	Corruption on internal CAN bus, inadequate ECU programming	Integrity attack (e.g., spoofing, malware)
Control pathway	10. Engine STOP command sent but not received by engine	Wiring problem, corruption on internal CAN Bus	Availability attack (e.g., denial of service)
	11. Engine STOP command sent but received late	Wiring problem, corruption on internal CAN Bus	Availability attack (e.g., denial of service)
	12. Correct command sent but incorrect command received	Wiring problem, corruption on internal CAN Bus, race condition	Integrity attack (e.g., spoofing)
Controlled process	13. Conflicting engine commands	User incorrectly sets LUI to “Local” and Start/Stop button depressed	LUI intentionally set to “Local” and Start/Stop button depressed
	14. Engine does not stop after STOP command received	Engine hardware/firmware issue.	

With UCA1-N-2 the engine could continue to run after reactor trip leading to a PCB overcooling event. Normal operation upon a reactor trip is for an engine stop signal from the MARVEL CS to be sent to the engine by the ECU. In an Operator-initiated shutdown, the Operator will use the HMI to stop the engine, and this signal will be sent through the MARVEL CS to the ECU and engine. Without a stop signal, the engine will continue to run until the lower heater head threshold temperature is reached.

Similar to UCA1-N-1, an engine stop command may not be sent to the engine due to an improperly designed or misconfigured ECU, wiring issues in the connection from the ECU to the engine, noise or corruption on the internal CAN bus, race conditions, or an integrity or availability attack. Additionally, the engine or ECU could be degraded or have hardware or firmware issues leading to maloperation. And there is a possibility that the LUI is improperly operated, resulting in no commands sent or conflicting commands sent.

There could also be corruption along part of the communication pathway in the network architecture from the HMI to the CS to the ECU. Further, there could be maloperation due to noise or an integrity or availability attack. The programming or configuration related to the reactor trip/shutdown signal may also be inadequate or compromised.

Referring to the scenarios listed in Table 3, engineering design and information assurance modifications could reduce the digital risk associated with this overcooling event. Potential engineering design modifications and operational activities include:

- Improved design traceability to V&V requirements, including V&V throughout the systems engineering lifecycle (especially during implementation, and updates) to test the ability of the reactor trip/shutdown signal to stop the engine. (Relevant for all scenarios)
- A diverse and redundant engine stop circuit could be added to enable manual engine shutdown from the control room or other location. This could be an analog circuit similar to the flow switch for UCA1-N-1 that is directly wired to the engine via a relay or contact. This redundant circuit relies on procedure adherence, so human performance error risk remains. (Relevant for all scenarios)
- Addition of an interim cooling system that can isolate the secondary coolant system from the engines. (Relevant for all scenarios)[d]
- Establish operator rounds (dependent on equipment accessibility) and a condition monitoring program to monitor the ECU, and engine operability for anomaly detection, diagnostics, and prognostics. (Relevant for scenarios 3, 7, 14)

Potential traditional information security controls include:

- Physical and technical security controls to limit access to the ECU. This would include requiring that the ECU cabinet be in a locked room with a locked cabinet that can only be accessed via the maintenance laptop through a controlled user account with password. The username and password should be kept in a controlled manner and only provided to authorized users. (Relevant for scenarios 2, 9)
- The maintenance laptop should be kept in a secured and monitored location, such as required for maintenance and test equipment, with user check in and check out strictly controlled. The maintenance laptop should be scanned for viruses and malware prior to use. (Relevant for scenarios 2, 9)
- The LUI panel should be physically secured, similar to the ECU. An additional physical security control is the addition of a key on the panel to limit local control to those who have access to the key. This key should be controlled under a key control program and only provided to users with access/use authority. (Relevant for scenarios 8, 13)
- Add a defensive security architecture that ensures the isolation of the CAN bus communication between the engine and the ECU. Extend this architecture to the entire process network to ensure that it is appropriately segmented into levels and zones with appropriate boundary devices to prevent

[d] Note that in 2025, the MARVEL project team designed a new heat extraction system that will act as an intermediate cooling loop, with the engines moved out of the reactor pit. One reason for this change was the postulated overcooling event; another reason was related to engine reliability issues when mounted on top of the reactor.

inadvertent network flooding (e.g., broadcast storm) or intentional, malicious data availability attack (e.g., denial of service attack). Include continuous network and endpoint monitoring via a security event and information monitoring system or similar. (Relevant for scenarios 5, 6, 8, 9, 10, 11, 12)

4.1.1.3 UCA1-P-1 scenarios and risk treatments

Table 4. Potential scenarios for UCA1-P-1, engine stopped when reactor is running leading to loss of reactor cooling and reactor trip on high temperature.

Category	Scenario	Threat Type Example	
		Unintentional	Intentional
ECU Behavior	1. Design flaw in ECU	Inadequate design or programming of ECU	Supply chain attack, malware
	2. Misconfiguration of ECU	User incorrectly configures ECU	Intentional misconfiguration
	3. Degradation/failure of ECU	Component wear, poor environmental conditions, ECU hardware/firmware issue	
Feedback	4. Incorrect flow voltage sent to the ECU (engine STOP command on low flow)	Degradation/failure of the flow/temp sensor, air in loop	
	5. Incorrect temperature voltage sent to ECU (engine STOP command on high temp)	Degradation/failure of the flow/temp sensor, coolant leak	
	6. Incorrect flow/temp voltage received by the ECU	Wiring problem, noise, pin connection issue	
	7. Incorrect flow/temp value derived by the ECU	User incorrectly calibrates the sensor A2D in the ECU	Intentional sensor A2D miscalibration
	8. Communication between the flow/temp sensor and ECU is unavailable (e.g., value not received by ECU)	Wiring problem, pin connection issue	
	9. Communication between the flow/temp sensor and ECU is delayed.	Wiring problem, noise, pin connection issue	
	10. Inaccurate reactor status/engine STOP signal sent to ECU from network	Corruption along communication pathway	Integrity attack (e.g., spoofing)
	11. Engine status is not sent to the ECU	Engine hardware/firmware issue	
	12. Engine status is not received by the ECU	Wiring problem, noise, user inaccurately sets the LUI to “Local”	Availability attack (e.g., denial of service), LUI intentionally set to “Local”
	13. Inaccurate engine status is sent to the ECU (e.g., engine is stopped, but the status sent is ON)	Corruption on internal CAN bus, inadequate ECU programming	Integrity attack (e.g., spoofing, malware)
Controlled process	14. Conflicting engine operation action	User incorrectly sets LUI to “Local” and Start/Stop button depressed	LUI intentionally set to “Local” and Start/Stop button depressed
	15. Engine unexpectedly stops	The voltage on the emergency stop CAN interface drops below 2 V, engine degradation or failure	

With UCA1-P-1, the engine could stop when the reactor is running leading to a loss of reactor cooling and reactor trip on high primary coolant system temperature. Of note is that this is the appropriate response from the RPS; the RPS should trip the reactor on high primary coolant system temperature. However, a facility owner or reactor vendor may consider this a high consequence event based upon a business requirement for continued electrical generation (also calculated as capacity factor in the nuclear industry).

Similar to the scenarios for the previous UCAs, the engine may stop when not required (or desired) due to an improperly designed or misconfigured ECU, wiring issues on the connection from the ECU to the engine, noise or corruption on the internal CAN bus, race conditions, or an integrity or availability attack. Additionally, the engine or ECU could be degraded or have hardware or firmware issues leading to maloperation. If the signal from the analog flow and temperature switch is either sent to, received by, or derived as too low (flow) or too high (temperature) the ECU will send a STOP command to the engine.

And there is also a possibility of improper operation of the LUI, resulting in no commands sent or conflicting commands sent to the engine. There could also be corruption, an integrity attack, or an availability attack along any part of the communication pathway in the network architecture, from the HMI to the CS to the ECU.

Referring to the scenarios listed in Table 4, engineering design and information assurance modifications could reduce the digital risk associated with this reactor trip event. Potential engineering design modifications and operational activities include:

- Improved design traceability to V&V requirements, including V&V throughout the systems engineering lifecycle (especially during implementation, and updates) to ensure the engine remains running as expected through a wide range of operational scenarios. (Relevant for all scenarios)
- Establish operator rounds (dependent on equipment accessibility) and a condition monitoring program to monitor the ECU, and engine operability for anomaly detection, diagnostics, and prognostics. (Relevant for scenarios 3, 4, 5, 11, 15)

Potential traditional information security controls include:

- Physical and technical security controls to limit access to the ECU. This would include requiring that the ECU cabinet be in a locked room with a locked cabinet that can only be accessed via the maintenance laptop through a controlled user account with password. The username and password should be kept in a controlled manner and only provided to authorized users. (Relevant for scenarios 2, 7, 13)
- The maintenance laptop should be kept in a secured and monitored location, such as required for maintenance and test equipment, with user check in and check out strictly controlled. The maintenance laptop should be scanned for viruses and malware prior to use. (Relevant for scenarios 2, 7)
- The LUI panel should be physically secured, similar to the ECU. An additional physical security control is the addition of a key on the panel to limit local control to those who have access to the key. This key should be controlled under a key control program and only provided to users with access/use authority. (Relevant for scenarios 12, 14)
- Add a defensive security architecture that ensures the isolation of the CAN bus communication between the engine and the ECU. Extend this architecture to the entire process network to ensure that it is appropriately segmented into levels and zones with appropriate boundary devices to prevent inadvertent network flooding (e.g., broadcast storm) or intentional, malicious data availability attack (e.g., denial of service attack). Include continuous network and endpoint monitoring via a security event and information monitoring system or similar. (Relevant for scenarios 10, 13)

The initial plan was to use Innoslate with Simulink to evaluate how digital risk changes before and after changing the design and/or adding cybersecurity controls. However, as described in the following sections, this evaluation was not possible with Innoslate and this part of the research was halted. Therefore, a future step in this project is to perform decision analysis on the mitigation or elimination of UCAs using the MathWorks MBSE environment comprised of Requirements Toolbox, Simulink, and System Composer.

4.1.2 Risk management integration

Prior work developed an entity-relationship diagram mapping a digital risk management process where both intentional and unintentional threats exploit vulnerabilities in digital assets leading to loss of critical functions and adverse consequences [10]. This diagram also included risk treatments, including digital requirements and the elimination or reduction of digital risk using engineering design, protective measures, and detection and response capabilities. To demonstrate the incorporation of this risk management process into an MBSE tool, a class diagram was developed in Innoslate where each entity was established as a class with related attributes and operations (Figure 9). Purple represents the digital requirements; red represents the typical digital risk parameters of threats, vulnerabilities, and consequence; green represents risk reduction measures, and blue indicates assets and functions.

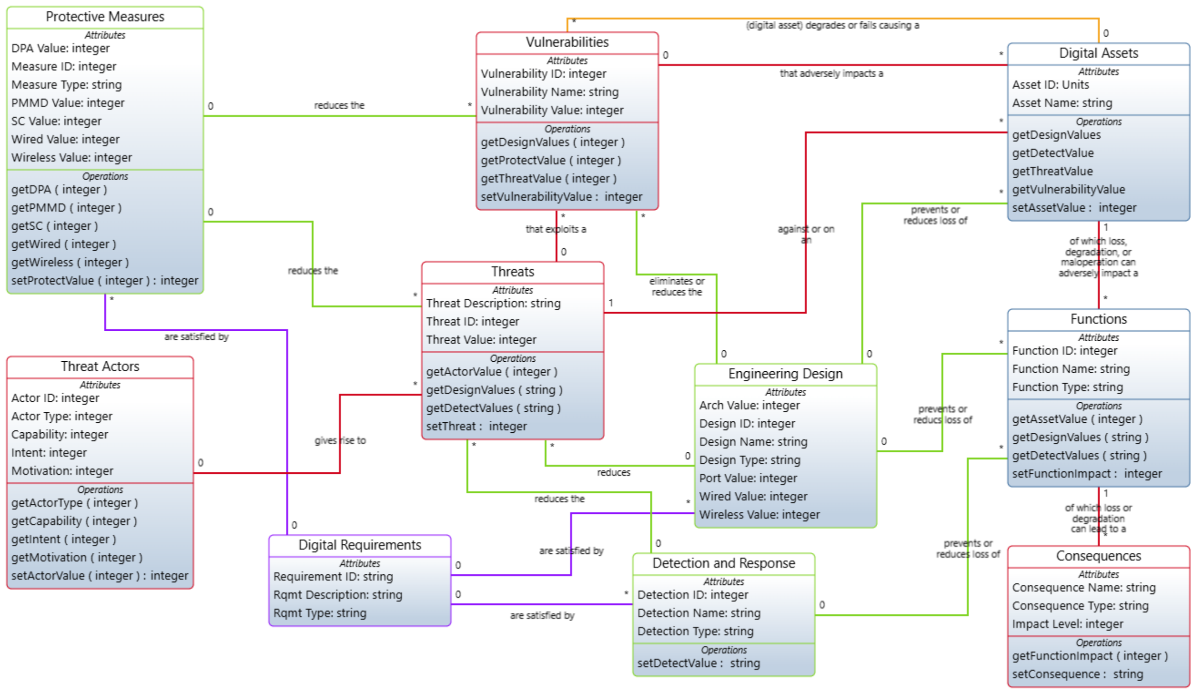
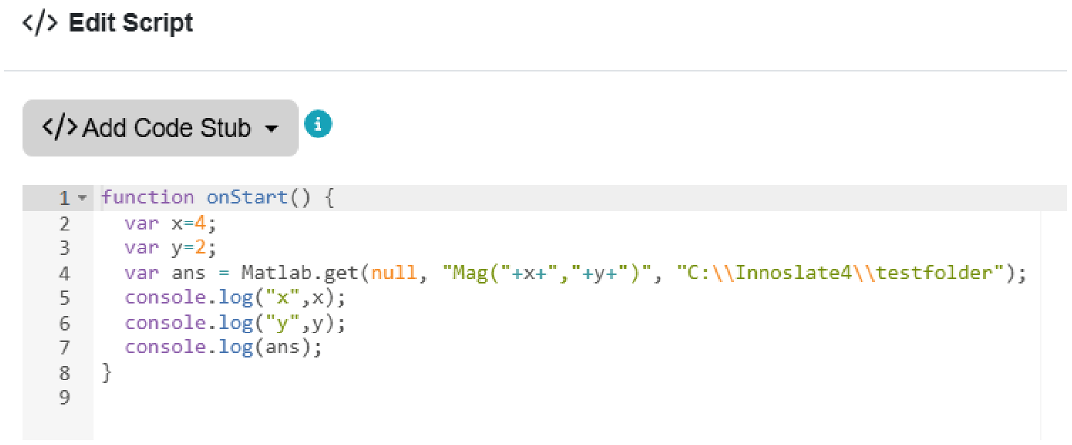


Figure 9. Entity-relationship class diagram for modeling digital risk.

The intent of this activity was to establish a traceable, programmatic method for defining and treating risk. However, while this was modeled in Innoslate, there was limited capability for using it within the Innoslate environment. This class diagram approach, using the established attributes and operations, will be evaluated for use in the MathWorks environment in the future. Ideally, this approach can be integrated with STPA to help automate and formalize the process of identifying UCAs, scenarios, and risk treatments.

4.1.3 Modeling and simulation integration

Innoslate integration with Simulink was attempted via the scripting interface of Innoslate action diagrams and through the export of Innoslate database information. However, this approach resulted in several issues that ultimately led to a change in methodology. First, the Innoslate scripting limitations only allowed for returning results from MATLAB functions for each action diagram element with individual scripts. Innoslate scripting allowed a user to define specific code associated with each action block in the diagram. The MATLAB.get function within Innoslate was utilized to call a simple MATLAB code for magnitude calculation as a proof of concept (Figure 10). This could be used for returning numerical results in the action diagram simulator but was not able to integrate with a Simulink diagram. The script feature could only be utilized with a specified MATLAB function to return the results to the web browser console.



```
function onStart() {
  var x=4;
  var y=2;
  var ans = Matlab.get(null, "Mag("+x+","+y+")", "C:\\Innoslate4\\testfolder");
  console.log("x",x);
  console.log("y",y);
  console.log(ans);
}

```

Figure 10. Innoslate scripting feature for MATLAB integration.

Another issue with utilizing MATLAB functions in Innoslate was the hosting and licensing of MATLAB. The MATLAB.get function is limited to passing in variables to a MATLAB file hosted on the same server as Innoslate. MATLAB must be installed on the Innoslate server, and all MATLAB files must be hosted there. Due to the enterprise architecture used at INL, access to specific server locations requires elevated permissions that most users do not have. Therefore, an administrative user must upload and manage server-side MATLAB files before they may be called in Innoslate. This may be circumvented if a user has administrator access to all ICT assets but is impractical for larger organizations with more complicated architecture. Additionally, while Innoslate can utilize MATLAB scripts, the link between the programs cannot pass model architecture to Simulink through MATLAB.

Since Innoslate could not be used to automatically recreate a Simulink model through the scripting interface, an attempt was made to export the Innoslate database and import it into Simulink via the SysML Composer. The issue with this, however, is that Innoslate is an lifecycle modeling language tool with some SysML capabilities. The regeneration of the original Innoslate system models was not possible due to the mismatched syntax of Innoslate exports and SysML files. Because of this, each model would need to be recreated in Simulink instead of imported.

Innoslate does not have the ability to provide simulation of system functionality. The simulation features within Innoslate are limited to action diagrams. Since these action diagrams could not be sufficiently linked to MATLAB/Simulink for modeling it was decided to forgo further efforts within Innoslate in favor of using Simulink directly within the larger MathWorks MBSE environment as described in the following section.

4.2 MathWorks Evaluation

4.2.1 Requirements integration

For simplicity and expediency, an initial sample set of requirements was manually created in the Simulink model. If these prove to be useful, then methods utilizing imports of requirements in comma-separated value or requirements interchange format may be pursued.

4.2.2 System architecture in System Composer

System Composer was used to develop the system architecture shown in Figure 11. This design models the network architecture in Figure 7. The functional subsystems (e.g., PGS, communication, CS) are divided between two buildings. Using System Composer, both structural (physical location) and functional decomposition were modeled. Additionally, each block in the architecture model was connected to Simulink models that represent its physical composition or behavior, as described in the following section. Data transmission between the systems occurs through designated in and out ports with custom stereotypes applied to ports to constrain the type and structure of the data transmitted.

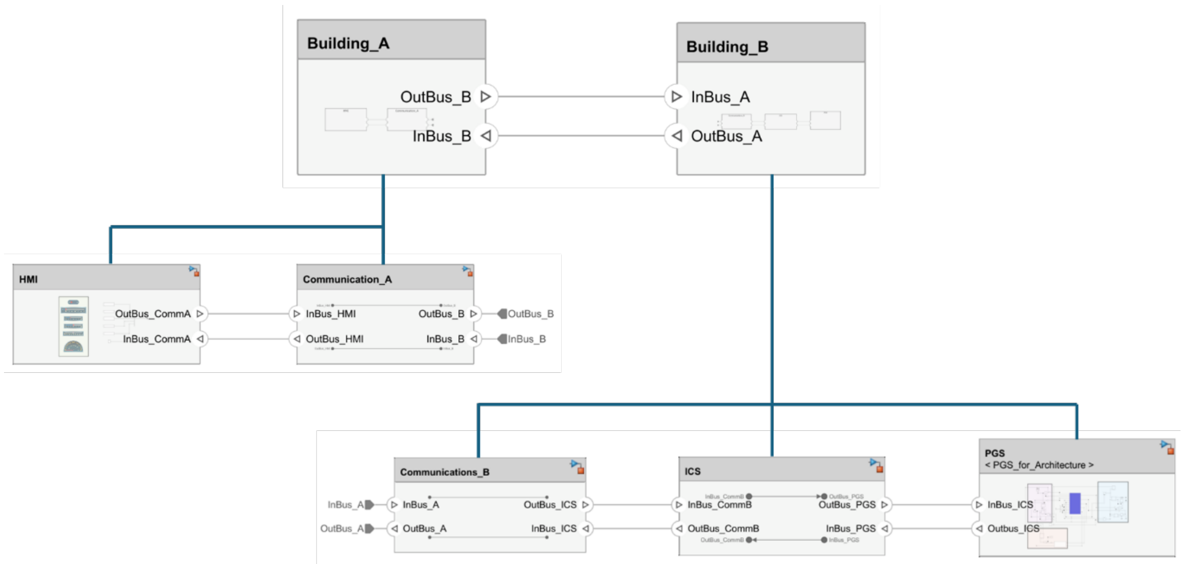


Figure 11. System architecture showing structural (buildings) and functional (subsystems) decomposition.

4.2.3 Physical system modeling in Simulink

Simulink was used to model the physical components of the subsystems. The current version of the model focuses on the PGS, although the modular nature of the architecture allows for easy expansion to other systems in the future. The Simscape library was used to model physical components (coolant network, pumps, fan, motors, sensors, heat exchangers, etc.). The following assumptions were used in developing the PGS model:

1. The ECU is capable of controlling the engine cooling inner loop pump speed, outer loop pump speed, and heat rejection unit fan motor speed.
2. The thermal-to-electric power conversion efficiency of the engine is assumed to be 17%.
3. Both inner and outer-loop coolant pumps can provide flow rates between 20 L/min and 50 L/min. The desired flow rate is set by the operator through the HMI.
4. The outer-loop return-side coolant temperature ranges between 50°C and 90°C. The desired temperature setpoint is specified by the operator through the HMI.

5. The control system generates a warning if the coolant temperature differential across the engine exceeds 10°C.
6. The control system generates an alarm if the rate at which the engine coolant temperature decreases exceeds -15°C/min or if the rate at which the engine coolant temperature increases exceeds 7°C/min.
7. The control system generates an alarm if the coolant for the engine is above 70ºC.

The PGS model is shown in Figure 12. Each subsystem (inner and outer coolant loops, engine control unit, sterling engine) is shown in a different colored box.

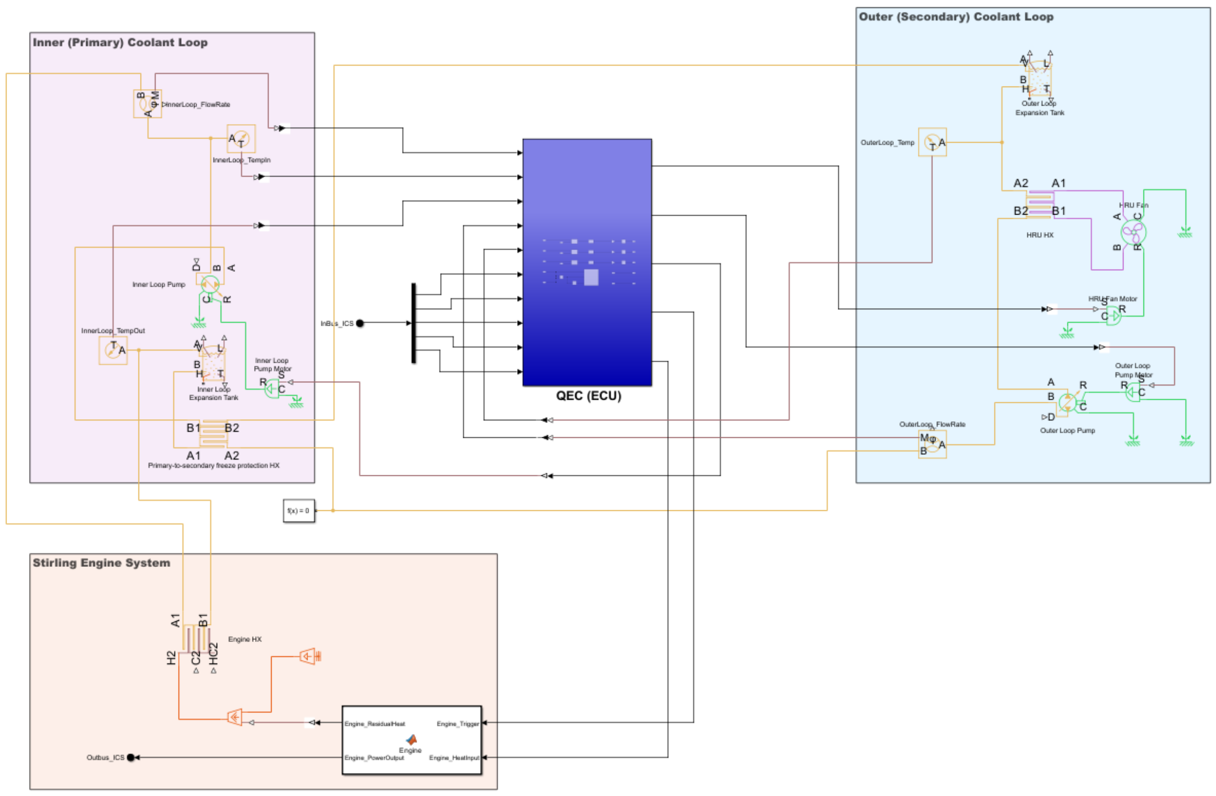


Figure 12. Simulink model for the PGS showing the subsystems.

For this model, the reactor was simplified into a heat source. The Stirling engine was modeled using a MATLAB function to calculate the electric power output residual heat. In the model, the residual heat is transferred to the primary coolant through a heat exchanger and each of the two coolant loops have a pump that controls their respective flow rate. Additionally, the outer loop is air-cooled using a fan. The ECU controls the motor speed for both pumps and the fan.

Figure 13 illustrates the ECU control logic. The ECU controls the motor speed for the inner and outer loop pumps as well as the outdoor fan. For both pumps, it is assumed that a proportional-integral-derivative controller maintains the coolant flow rates at the operator-provided setpoints. The proportional-integral-derivative output is then converted into a pulse wave modulated signal. A similar control scheme was implemented for the fan, but the feedback variable is the coolant temperature. The control signals need to be scaled to the maximum motor speed for each of these components; since these values have not yet been derived, the current placeholder value is 1.

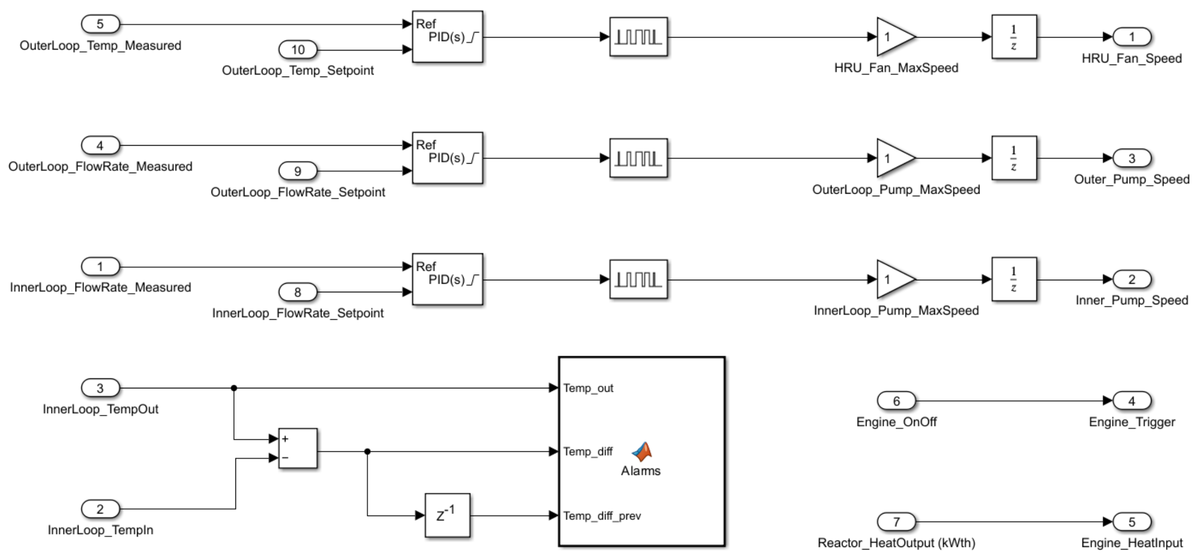
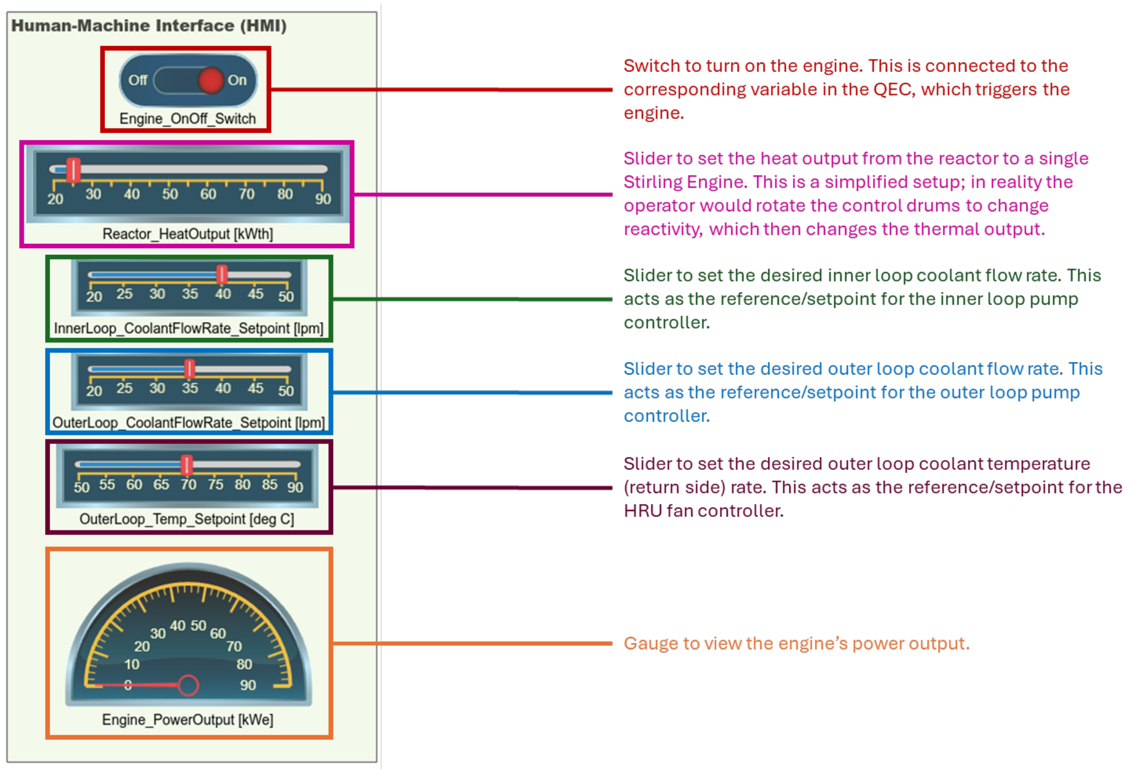


Figure 13. ECU control logic.

Operators interact with the ECU through an HMI panel in the control room. For MARVEL, this control panel is located in a separate building. The setpoints for the ECU were set in the HMI and the performance was observed through gauges and indicators. Figure 14 shows the setup and describes the functions.



Human-Machine Interface (HMI)
Off
On
Engine_OnOff_Switch
Reactor_HeatOutput [kWth]
InnerLoop_CoolantFlowRate_Setpoint [lpm]
OuterLoop_CoolantFlowRate_Setpoint [lpm]
OuterLoop_Temp_Setpoint [deg C]
Engine_PowerOutput [kWe]
Switch to turn on the engine. This is connected to the corresponding variable in the QEC, which triggers the engine.
Slider to set the heat output from the reactor to a single Stirling Engine. This is a simplified setup; in reality the operator would rotate the control drums to change reactivity, which then changes the thermal output.
Slider to set the desired inner loop coolant flow rate. This acts as the reference/setpoint for the inner loop pump controller.
Slider to set the desired outer loop coolant flow rate. This acts as the reference/setpoint for the outer loop pump controller.
Slider to set the desired outer loop coolant temperature (return side) rate. This acts as the reference/setpoint for the HRU fan controller.
Gauge to view the engine's power output.

Figure 14. PGS and ECU HMI setup.

5 DISCUSSION AND FUTURE WORK

CIE is intended to be used with existing hazard analysis techniques. As noted in [11], the baseline STPA method incorporates seven of the twelve CIE principles. Using both CIE and STPA together provides an integrated process that effectively identifies adverse impacts from the use of digital technology and reduces overall digital risk through use of both traditional engineering design and traditional information assurance risk treatments. This project expanded on the ECU controlling action "Engine STOP" and identified six different UCAs associated with this action. Considering three of these UCAs, multiple scenarios were developed, and corresponding risk treatments were identified. Additionally, examples of both intentional and unintentional threats were provided. Moving forward, the changes in engineering design and/or addition of cybersecurity mitigations will be evaluated in the MBSE ecosystem currently under development within the MathWorks environment.

The currently envisioned method for performing these analyses relies upon the creation or existence of a sufficiently complete functional model of the system under examination. Ideally, such a model would be an existing product of a well-functioning systematic design approach and would be available for import or use via an open data standard (such as SysML). In practice, this is not always the case, and the model must be manually created from other existing project documentation. The complexity of this task may vary depending upon the scope and quality of available data.

Innoslate and MathWorks are each capable tools for MBSE, where Innoslate is primarily a systems engineering tool with limited simulation capabilities, and MathWorks is primarily a simulation tool with added systems engineering capabilities. Based on the outcome of this research, it was determined that the MathWorks ecosystem will provide a more straightforward pathway to a flexible and successful demonstration for integrating STPA. Both tools offer the ability to interact with external sources of data, however the documentation available for the MathWorks tools is considerably more accessible and complete, lending further weight to its selection for continued evaluation. Additionally, modeling risk management and integrating simulation capabilities in Innoslate was unsuccessful within the current capabilities of the resources available.

The MathWorks environment was expanded to include System Composer which enables the ability to define a system architecture and network architecture to model information flows. Future work will continue to build on this MathWorks MBSE ecosystem to evaluate the capabilities for automating and formalizing the STPA risk analysis and investigating how decision analysis can be used for evaluation of engineering and ICT/OT solutions in elimination or reduction of UCAs and reduction of digital risk. Given a suitable model, it appears the computational aspects of the analysis can be highly automated, but continued research is necessary to evaluate how the results are interpreted and the identified UCAs or issues are mitigated.

Future work will also expand the evaluation to include the MARVEL RCS and I&C system to enable a more thorough system of systems evaluation. Requirements will be linked to test cases to show how and where requirements are or are not met through selected operational scenarios. It is anticipated that this expanded model will be used for integrated collaboration with both the DOE-ARSS remote monitoring project and ARCADE project. Not only will the recommended changes to the network architecture from this research be analyzed from a digital risk perspective, but proposed changes for secure remote operation or monitoring RCS will also be analyzed.

The benefits of using integrated tools in an MBSE ecosystem enable not only the development of output reports, but also the capability of maintaining the analysis results in an integrated product lifecycle management application to ensure identified issues and solutions are maintained throughout the systems engineering lifecycle. Using an integrated MBSE ecosystem maintains the digital thread, thereby reducing the need to input the same data multiple times to accomplish different purposes. For example, system functions and identified requirements could automatically be transferred to the digital risk management process, including consequence and vulnerability assessments.

6 CONCLUSION

As recognized in the INCOSE Systems Engineering Vision 2035, systems solutions will be increasingly cyber-physical systems which need to incorporate security, privacy, and explainability as foundational perspectives along with the traditional perspectives of system performance and safety [1]. Using an MBSE ecosystem which integrates STPA and CIE into the engineering decision analysis process will provide tools an organization can use to link project management, stakeholder analysis, requirements management, product lifecycle management, design, and simulation to enable development of digital systems that include safety, performance, reliability, and security as baseline attributes.

This project expanded on the list of UCAs identified for the MARVEL ECU system, described scenarios in which the UCAs could occur (including intentional and unintentional digital threat examples), and listed potential engineering design and information assurance solutions that could eliminate or mitigate the UCAs. It was attempted to integrate digital risk management, network architecture, and ECU simulation with Innoslate, but there were several limitations that prevented the successful use of the tool. Thus, efforts were refocused on using the MathWorks ecosystem which includes Simulink, System Composer, and the Requirements Toolbox connector.

System Composer was successfully used to develop the MARVEL PGS and network architecture from the control room HMI to the ECU. The physical systems associated with the PGS were also successfully modeled in Simulink. Future work will continue development of the models to include the MARVEL RCS and CS to enable an evaluation of the improvement suggestions from the STPA process. Integration with the DOE-ARSS remote monitoring and ARCADE projects are also planned for future work.

7 REFERENCES

[1] DOE, "National Cyber-Informed Engineering Strategy from the U.S. Department of Energy," U.S. Department of Energy Office of Cybersecurity, Energy Security, and Emergency Response, 2022, Available: https://www.energy.gov/sites/default/files/2022-06/FINAL%20DOE%20National%20CIE%20Strategy%20-%20June%202022_0.pdf.

[2] Lampe, B., "INL/RPT-25-87184, Hazard understanding for security and hardening (HUSH) Chart," Idaho National Laboratory, 2025.

[3] Leveson, N.G. and J.P. Thomas, "STPA Handbook," Massachusetts Institute of Technology, 2018, Available: http://psas.scripts.mit.edu/home/get_file.php?name=STPA_Handbook.pdf.

[4] INCOSE, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, version 5.0*. Hoboken, NJ, USA: John Wiley and Sons, Inc., 2023.

[5] Johnson, J., C. Parisi, B. Baker, R. Clark, S. Jensen, and A. Abou-Jaoude, "INL/RPT-25-84647, Lessons learned from MARVEL initial fabrication and testing," Idaho National Laboratory, 2025, Available: https://doi.org/10.2172/2569192.

[6] Gerstner, D. and Y. Arafat, "INL/RPT-23-74280, MARVEL 90% final design report," Idaho National Laboratory, 2023, Available: https://doi.org/10.2172/2406267.

[7] Ahlbrecht, A. and U. Durak, "Integrating safety into MBSE processes with formal methods," in *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, pp. 1-9: IEEE.

[8] Ahlbrecht, A. and O. Bertram, "Evaluating system architecture safety in early phases of development with MBSE and STPA," in *2021 IEEE International Symposium on Systems Engineering (ISSE)*, 2021, pp. 1-8: IEEE.

[9] de Souza, F.G.R., J. de Melo Bezerra, C.M. Hirata, P. de Saqui-Sannes, and L. Apvrille, "Combining STPA with SysML modeling," in *2020 IEEE International Systems Conference (SysCon)*, 2020, pp. 1-8: IEEE.

[10] Eggers, S.L., K. O'Rear, R. Hays, and P. Suyderhoud, "INL/RPT-24-80756, Security inclusive model-based systems engineering: Demonstration using the MARVEL reactor," Idaho National Laboratory, 2024.

[11] Wright, V.L. *et al.*, "Adapting traditional hazrds analysis methods to addres cyber risks: A Cyber-Informed Engineering (CIE) approach," Idaho National Laboratory, 2025.