



New Project Incubator

Technical Report SERC-2018-TR-105

April 30, 2018

Principal Investigator:

Jon Wade, Stevens Institute of Technology

Research Team:

David Coe, University of Alabama Huntsville

Paul Grogan, Stevens Institute of Technology

Azad Madni, University of Southern California

Karen Marais, Purdue University

Tom McDermott, Georgia Institute of Technology

Gary Witus, Wayne State University

Lu Xiao, Stevens Institute of Technology

Sponsor: DASD(SE)



Castle Point on Hudson, Hoboken, NJ 07030

Copyright © 2018 Stevens Institute of Technology, Systems Engineering Research Center

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract HQ0034-13-D-0004 (TO#0286).

Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense nor ASD(R&E).

No Warranty.

This Stevens Institute of Technology and Systems Engineering Research Center Material is furnished on an “as-is” basis. Stevens Institute of Technology makes no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material. Stevens Institute of Technology does not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

This material has been approved for public release and unlimited distribution.

Table of Contents

1	Introduction	5
2	The Feasibility of Enhancing Security of Complex Systems Through Disaggregation of Security Components to Asymmetric Multi-Processors – David Coe, University of Alabama-Huntsville	8
2.1	Motivation	8
2.2	Proposed Approach and Its Advantages	8
2.3	Security Coprocessors: Current Practice and Limitations	10
2.4	Proposed Work and Its Impact	11
2.5	Risk and Mitigations	12
2.6	Project Management	12
2.7	References	14
3	Game-theoretic Risk Assessment for Distributed Systems (GRADS) - Paul Grogan, Stevens Institute of Technology	16
3.1	Introduction and Motivation	16
3.2	Value-Centric Design for Distributed Systems	16
3.3	Game Theory and Equilibrium Selection	18
3.4	Multi-actor Value Functions and Strategic Design Games	19
3.5	Example Application Case	20
3.6	Extensions and Opportunities	25
3.7	References	26
4	Program Protection System Security and Trust Extending Flexible Contracts for Mission Assurance – Azad Madni, University of Southern California	28
4.1	Introduction	29
4.2	Current MA Approach	29
4.3	The Flexible Contract Construct	31
4.4	Closed Loop MA Approach	35
4.5	POMDP Concept of Operations for Exemplar Problem	36
4.6	An Illustrative Example	37
4.7	Concluding Comments	40
4.8	References	40
5	Data Science Approaches to Prevent Failures in Systems Engineering - Karen Marais, Purdue University	41
5.1	Our Vision and Roadmap	42
5.2	Relational Deep Learning with few data points	44
5.3	Identifying Potential Systems Engineering Failure Signals	46
5.4	Case study: Student Teams	52
5.5	Conclusion	54
5.6	References	54
6	Systemic Security and the Role of Heterarchical Design in Cyber-Physical Systems – Tom McDermott, Co-PI Val Sitterle, Georgia Tech	57
6.1	Objectives of the Research	57
6.2	Current Practice and its Limitations	57
6.3	Research Approach	59
6.4	Relevance of the Research	66

6.5	Risks, Mitigations, and Payoffs	66
6.6	Budget.....	66
6.7	Timeline	67
6.8	Project Evaluation and Measurement.....	67
6.9	References.....	67
7	Trusted Autonomy – Methods for Test and Evaluation – Gary Witus, Wayne State	69
7.1	Problem and Need	69
7.2	Technical Solution.....	71
7.3	Program Plan.....	76
7.5	Conclusions and Recommendations	79
7.6	References	79
8	Identifying and Measuring Modularity Violations in Cyber-Physical Systems – Lu Xiao, Co-PI	
	Michael Pennock, Stevens Institute of Technology	81
8.1	Problem Statement, Motivations, and Objectives	81
8.2	Current Practice and Its Limitations	83
8.3	Research Approach	84
8.4	Proposed Work in the Next Phase	88
8.5	Relevance of the Research	90
8.6	Expected Contribution	91
8.7	Risks and Payoffs	92
8.8	Budget.....	92
8.9	Timeline	92
8.10	Project Evaluation and Measurement	93
8.11	References	93

1 INTRODUCTION

As described in SERC Technical Plan, the SERC performs research on 20-25 active tasks on well-defined topics that are aligned with the SERC's research strategy. While it is believed that the aforementioned research programs have a great potential to have a transformative impact on the DoD and IC, there is a need to support new ideas in their infancy that may become the critical research programs for emerging challenges. This incubation capability will be supported by a biennial open call to the SERC research collaborating universities to propose early stage research that can be nurtured through relatively small levels of seed funding.

The initial open call took place in December 19, 2016 with the objective of identifying and developing several short white papers outlining research programs with a significant potential to improve the practice of engineering systems. A total of 33 responses were received as shown in Table 1 below:

Table 1: Responses to 2016 SERC Incubation Grant Solicitation

University	PI	Team	Title
AFIT	Badiru, Deji		Toward the Postulation of Theory of Systems Engineering
UAH	Bailey, Carmine		Augmented Reality Technologies
NPS	Beery, Paul	Paulo, Eugene	MBSE for Methodology for the Employment of Architectures in Systems Analysis (MEASA)
Stevens	Chandramouli, R.		Co-operative Artificial Intelligence for Cognitive Mobile Systems
Stevens	Chandramouli, R.		Modeling and Simulation of Synthetic Bio-Nano Systems
UAH	Coe, David	Kulick, Jeffrey	The Feasibility of Enhancing Security of Complex Systems through Disaggregation of Security Components to Asymmetric Multi-Processors
Purdue, Wayne State, Gtech, Stevens	DeLaurentis, Dan	Davendalingam, Navindran Witus, Gary Paredis, Chris Xiao, Lu	Approaches to Achieve Modularity Benefits in Defense Acquisitions
Purdue	Mendoza-Garcia, John	Cardella, Monica Kenley, Charles	A Learning-Theory-Based Assessment Instrument to Measure the Ability to Address Complex Socio-Technical Systems
NPS, Missouri S&T	Giammarco, Kristin	Dagli, Cihan	Methods and Analytics for Purging System Designs of Unwanted Behaviors
Stevens	Grogan, Paul		Game-theoretic Risk Assessment for Distributed Systems (GRADS)
Stevens	Hoffenson, Steven	-	Agent-based Modeling of Semi-autonomous Military Vehicle Systems
Stevens, USMA	Klappholz, David	Engling, Michael Condly, Steven	Building a Computer Science/Software Engineering Education Research Group

USC	Madni, Azad		Flexible and Adaptable Systems and Processes Negotiation and Sociability in Resilient SoS Networks
USC	Madni, Azad		SE Transformation for Next Gen. DoD Systems Adaptive Cyber-Physical Human Systems
USC	Madni, Azad		Program Proection System Security and Trust Extending Flexible Contracts for Mission Assurance
Purdue	Marais, Karen		Data Science Approaches to Prevent Failures in Systems Engineering
Gtech, UAH, SIT	Paredis, Chris	McDermott, Tom Collopy, Paul Blackburn, Mark Pennock, Mike	ESoS Model for Digital Thread Enabled Acquisition
Gtech	McDermott, Tom	Sitterle, Val	Systemic Security and the Role of Heterarchical Design in Cyber-Physical Systems
USC	Medvidovic, Nenad		Systematic Exploration of Decision Spaces in CPH Systems
Stevens	Pennock, Michael		Identifying Leading Indicators of System of System Interoperability Issues
Auburn	Smith, Alice E.	Sakinc, Eren	Cost Prediction in the Presence of Categorical and Numeric Cost Drivers
Stevens	Subbalakshmi, K.P.		Resilient, Fast Mobile Data to Decisions Systems
UAH	Thomas, Dale		Robust Methods in Model Based Systems Engineering
USC	Wang, Chao		Automated Techniques for Testing and Diagnosing PLC Software
USC	Wang, Chao		Incremental Analysis of Concurrent Software Systems
Stevens	Wang, Wendy		Research and Development for Resilient Cyber- Physical-Human Systems
Wayne State	Witus, Gary		Trusted Autonomy -- Methods for Test and Evaluation
Stevens	Xiao, Lu	Pennock, Michael	Identifying and Measuring Modularity Violations in Cyber-Physical Systems
Stevens	Yang, Ye		A Hybrid Approach in Testing of Autonomous and Learning CPH Systems: Combining Human Computing and Machine Learning
Penn State	Yukish, Michael		Methodology for Manufacturability Design Assist Tools

Each of the received proposals were reviewed by the SERC Research Council except where they noted that they had a potential conflict. Each member provided a final score based on an equal weighing in each of the following four criteria as well as a set of short comments:

- Intellectual Merit
- Clarity of Vision
- Past Performance
- Potential Strategic Impact

Each of the Research Council member's ratings were normalized based on their average numerical score and an average score was calculated for each proposal. To aid in the visualization of the results, the rankings for each Research Council member were color coded to be Green – top 1/3, Yellow – middle 1/3 and Red – bottom 1/3. The proposals were ranked based on these results.

The sponsor independently ranked the top ten projects based on these criteria. Preference was given to proposals that contend with issues not currently being addressed by SERC research, or use novel approaches, but which support the current SERC UARC research focus areas and have a strong potential for additional funding outside of SERC core funds. In this case, the selected six proposals were in the top ten proposals as determined by the Research Council.

Of these, the following seven proposals were selected:

1. David Coe, University of Alabama-Huntsville, *The Feasibility of Enhancing Security of Complex Systems through Disaggregation of Security Components to Asymmetric Multi-Processors*
2. Paul Grogan, Stevens Institute of Technology, *Game-theoretic Risk Assessment for Distributed Systems (GRADS)*
3. Azad Madni, University of Southern California, *Program Protection System Security and Trust Extending Flexible Contracts for Mission Assurance*
4. Karen Marais, Purdue University, *Data Science Approaches to Prevent Failures in Systems Engineering*
5. Val Sitterle, Georgia Tech, *Systemic Security and the Role of Heterarchical Design in Cyber-Physical Systems*
6. Gary Witus, Wayne State, *Trusted Autonomy -- Methods for Test and Evaluation*
7. Lu Xiao, Stevens Institute of Technology, *Identifying and Measuring Modularity Violations in Cyber-Physical Systems*

This report contains the white papers that was supported by each of these funded proposals.

2 THE FEASIBILITY OF ENHANCING SECURITY OF COMPLEX SYSTEMS THROUGH DISAGGREGATION OF SECURITY COMPONENTS TO ASYMMETRIC MULTI-PROCESSORS – DAVID COE, UNIVERSITY OF ALABAMA-HUNTSVILLE

2.1 MOTIVATION

Given the increasing frequency of cyber breaches and near daily announcements of newly discovered system vulnerabilities, it is clear that the computer systems that control critical cyber-physical systems or conduct critical business practices are under increasing threat of attack [1][2][3][4]. Cyberattacks have become so rampant and successful that the Federal Bureau of Investigation predicts an exponential increase in the frequency of cyberattacks for the foreseeable future [5]. To date the common industry practice of attempting to add security to the system after the fact through the addition of a firewall or malware scanner has been an inadequate response to the threat environment.

2.2 PROPOSED APPROACH AND ITS ADVANTAGES

2.2.1 WHAT ARE WE TRYING TO DO?

Security, as an emergent property of the system as a whole and the way the system is deployed, used, and maintained, must be a consideration at the outset. Our goal is to provide a solid computer architectural foundation that will make cyber-physical computing systems inherently more *securable*.

To this end, the proposed research project shall investigate the feasibility of disaggregating and exporting critical security functions such as code and data integrity checkers, to an Asymmetric Multiprocessor System-on-a-Chip (ASMP SoC) not accessible to the applications processor to enhance the overall security of cyber-physical systems. Several levels of realizations are proposed from passive snooping on application processor signatures through memory scanning for virus signatures through control flow graph verification through hardware architecture additions to soft core processors. Each of these levels of realization require different levels of cooperation between the exposed protected system and ASMP protector.

What is *new* in our approach is that the proposed security monitoring tasks are not vulnerable to software processes executed by the application processor, which is inherently untrustworthy because it is exposed and under attack, but rather these security monitoring tasks are carried out orthogonal to the application processor by the asymmetric multi-processor, which is sequestered behinds layers of software and hardware isolation.

2.2.2 ADVANTAGES OF OUR APPROACH

Our approach combining ASMP SoC with disaggregation of security components has a number of advantages that are enumerated below.

2.2.2.1 Enhanced Security through Use of SoC Technology

The semiconductor industry has already recognized that the use of SoC technology provides an inherently more securable platform by integration of components into a single chip package, making probing of communications between components far more challenging requiring observation of interconnects within a chip. Moreover, the use of ever decreasing feature sizes in the fabrication of the semiconductor components and their interconnects shall make it increasingly difficult for those without extensive knowledge and access to specialized hardware to successfully decapsulate and probe the SoC integrated circuit.

2.2.2.2 Enhanced Security through Disaggregation

Current security practice typically relies upon security services such as firewalls, anti-malware scanners, and event logging services being able to operate correctly despite the fact that they are being executed by the system that is under attack. It is our assertion that the system under attack is inherently unreliable, and it is thus unreasonable to assume that it can either fully protect itself or reliably gather the forensic data required for incident analysis, response, and recovery.

An excellent example of this is a desktop computer running typical antivirus software. Commonly available exploit tool suites such as Metasploit [6] include easy to execute scripts designed specifically to disable antivirus software once an attacker gains a foothold in the compromised system. Moreover, the Metasploit tool suite also includes scripts that can readily destroy the event and activity log information and thereby hamper the use of forensics techniques to discover how the attacker gained access to the system and what the attacker did once they gained access.

The principle of disaggregation is already commonly used to enhance the security of operating systems and hypervisors. In this context, disaggregation refers to the structuring of software components in a way that provides isolation to protect the system from untrusted services or components such as third-party device drivers. Operating systems make use of virtual address spaces to isolate processes. A hypervisor may encapsulate untrusted third-party device drivers entirely within a stub-domain whose only function is to provide secure wrapping of third party drivers. Garfinkel and Rosenblum proposed using a hypervisor to disaggregate the Intrusion Detection System (IDS) from the system being secured, placing both the protected system and the IDS into separate virtual machines running on the same physical host [7].

In our previous work funded by the National Security Agency, we introduced Dielectric Xen, a security-enhanced version of the Xen hypervisor in which we employed disaggregation to relocate an IDS from the guest and host virtual machines into the Xen hypervisor itself where it is less visible and less exposed to a remote attacker yet still able to perform the intrusion detection task [8][9]. Attackers that gain access to either the guest or host virtual machines are unable to view the IDS process since it is located outside of either virtual machine.

As part of the proposed project, we intend to disaggregate critical security services from application processors into (a) Field Programmable Gate Arrays (FPGAs) or (b) hard processor cores solely dedicated to security tasks that are hidden (disaggregated) from the target platform. These resources will serve as a trusted hardware component within the ASMP SoC. By selecting an SoC and board architecture in which the FPGA may only be reconfigured by physical access to the circuit board as through a JTAG port or boot rom, the FPGA will be inherently more resistant to remote attacks, making the FPGA-hosted security and event logging algorithms more reliable.

2.2.2.3 Enhanced Security through Asymmetry

We seek to establish and exploit *asymmetry in the level of exposure* of the protected systems and the ASMP hardware tasked with its protection. The designer of the computing system has control over which communication paths exist and whether those paths are read-only, write-only, or readable and writable. Careful design and selection of these communication paths shall provide additional protections to the security components that have been disaggregated from the protected system and placed in the ASMP SoC by eliminating unnecessary paths for interactions between the potentially compromised protected system and its ASMP protector.

We also seek to exploit *asymmetry in the execution engines* employed in the protected system and the protector to enhance security. The use of an FPGA allows for the deployment of a hardware-realization of a security algorithm, such as a malware signature scanner or an IDS. Again, proper ASMP SoC and board architecture selection shall make it more difficult for a remote attacker to alter the FPGA reconfiguration without physical access and thus provide for enhanced security for the disaggregated security services.

2.2.2.3 Enhanced Security through Encapsulation and Information Hiding

By sequestering the most critical security algorithms in the ASMP SoC FPGA, an attacker that compromises the protected system may be unaware of the existence of the security algorithms or the nature of those algorithms in use. An examination of running processes in the compromised protected system will reveal no information regarding the hidden algorithms executed in the FPGA.

2.3 SECURITY COPROCESSORS: CURRENT PRACTICE AND LIMITATIONS

A variety of coprocessors have historically been employed to improve overall system performance by off-loading computationally intensive tasks from the main processor [10]. Early examples of the use of coprocessors include the use of input/output coprocessors and floating-point coprocessors (FPUs). Dedicated Graphical Processing Units (GPUs) that include multiple cores to speed rendering of scene imagery have become very popular for gaming and machine learning applications. Cryptographic coprocessors were originally developed to accelerate encryption and decryption operations.

More recently, coprocessors have been employed specifically in an attempt to enhance system security. The Trusted Platform Module (TPM) is a secure coprocessor standard that attempts to create trusted regions of hardware for storage or generation of encryption keys [11]. The IBM 4758 is a FIPS 140 level 4 cryptographic coprocessor that provides additional support in the form of anti-tamper protection, but the processor is not a mainstream processor that can be used for critical infrastructure applications such as banking but not in cyber-physical systems such as missiles [12].

A number of companies including Altera and Microsemi SoC Corporation have investigated the use of a Field Programmable Gate Array (FPGA) as a part of a mechanism to secure the boot process [13] [14]. An FPGA may be used as a secure code storage location or to verify the signature of codes prior to booting the system. An advantage of this approach is that the FPGA may be integrated onto the same die as the processor (SoC) making it more challenging to access those stored codes or signatures since the chip package must be breached.

DARPA SSITH [15] is an ambitious 3+year, \$50M program that seeks to mitigate the following seven categories of hardware vulnerabilities: (1) buffer errors, (2) permissions/privileges/access control, (3) resource management, (4) code injection, (5) information leakage/exposure, (6) cryptographic errors, and (7) numeric errors. Participants are required to utilize a RISC-V soft-core processor in an FPGA board. Security additions to the systems are limited with respect to the impact on chip area (< 50% => < 30%) and performance (< 20% => <10%). Limitations of the SSITH approach includes maturity of the RISC-V development tool chains and the extensive base of existing software that would require rehosting for the RISC-V environment.

The modern processor cores can be partitioned and configured to run independent software stacks. For example, the ARM Cortex A family of processors supports the TrustZone technology [16]. TrustZone partitions a multi-core system between a secure world for critical system resources and a non-secure world for everything else. For example, one core can run a secure real-time operating system (RTOS) and

another core can run a main operating system (e.g., Linux). Thus, a trusted execution environment that includes parts of the system that can only be accessed by the secure world is separated from the potentially vulnerable operating system. With TrustZone processor cores and memory are partitioned between the secure and non-secure worlds thus preventing any interference. Two cores communicate over a shared physical memory channel. Peripherals are divided as well with no possibility of sharing and access rights are enforced using a security bit and security status of each resource. The non-secure software stack cannot change any configuration parameters that may affect the secure software stack. The secure software stack can typically access all system resources. In hybrid systems that combine hard ARM processor cores and FPGA fabric (Xilinx Zynq and Altera Cyclone V), TrustZone can be extended to custom IP cores residing in the FPGA fabric, making them an attractive option for deploying hardware-oriented security solutions.

It is clear after a review of the work summarized above and similar works that attempts to add security to a system after the fact are unsuccessful. ***One must design the system from the outset to be securable.***

2.4 PROPOSED WORK AND ITS IMPACT

Below is an overview of our 4-phase program for implementing ever more secure realizations of the ASMP paradigm. The four phases are briefly outlined below.

Phase 1 - An ASMP soft core processor will passively observe the memory of the application processor having been provided a layout of system and application codes in a real-time embedded system environment. Deviations from the layout and content will be announced through an independent communication channel to the operator in this phase, the protected system is assumed to be untrustworthy. The primary threat being remediated is code and data modification of application programs.

Phase 2 - In this phase, previously developed work by one of the applicants to passively extract a control flow graph information in real time from executing programs on a soft-core processor from the processor's instruction execution pipeline will be compared against a previously obtained allowable control flow graphs. The previous work demonstrated a very low bit bandwidth requirement of approximately 1 bit per instruction retired to capture executing control structures [17] [18]. The primary threat being remediated is return-oriented programming (ROP) attacks where the attacker, being thwarted in modifying existing code, uses code fragments from the legitimate application to assemble an attack. Although difficult and time consuming, this attack vector continues to be a significant threat.

Phase 3 - In Phase 3 we will begin to use the Trust Zone capabilities of the ARM processor to allow the protected host to provide information to the protector without requiring hardware modifications to the protected processor, as required by Phase 2. In this case the protected processor will provide real time control flow information to the protection processor using data collected from PMU registers currently present in ARM processors. In this case the control flow, data and code integrity provided by Phases 1 and 2 will be available without hardware modification. This phase will also explore generating live forensics data of attack mechanisms as they operate in the open. Threats remediated will move up the complexity chain and include threats to operating system architectural elements such as scheduling queues and memory management tables.

Phase 4 - In this phase all the remediations available in Phases 1-3 will be exported to an environment utilizing hypervisors. Recent unpublished work by the applicants has shown that real time cyber-physical

control systems can be deployed with hypervisor frameworks with extremely low latency penalties - on the order of 10s of microseconds on low cost processors such as the ODROID C2. This phase will investigate the extension of mitigations and remediations up to operating system and hypervisor architectural components such as scheduling tables and network interfaces.

A table containing additional propriety details on each phase of work is provided as an appendix.

2.4.1 PROPOSED WORK IMPACT

As discussed earlier, many attempts today to harden cyber physical systems consist of closing the door in hope the horse cannot be seen through the slats of the barn. The better way to develop secure systems is from the ground up, starting with the processors, operating systems, architectures, and deployment methodologies. The work proposed herein will provide system and software engineers a common platform for constructing cyber physical systems without jury rigging and wrapping systems that are inherently insecure. This should significantly **increase the quality of products** developed while reducing the procurement and maintenance costs of disparate systems.

2.4.2 AUDIENCE FOR PROPOSED WORK

Acquisition agencies, program management authorities, system developers, and people responsible for ensuring the safe and secure operation of cyber physical systems should welcome a potential **common secure deployment platform architecture** for future systems.

2.5 RISK AND MITIGATIONS

Risk #1 - Performance Impact on Real-Time Embedded Applications

Mitigation - Use an asymmetric SoC which does not use resources required by the embedded application.

Risk #2 - Insufficient Isolation Between ASMPs

Mitigation - Use threat analysis to identify vulnerabilities and reduce attack surface.

Risk #3 - Obtaining Buy In and Adoption of New Secure Architectures by Developers

Mitigation - Hard evidence of enhanced security and education of next generation developers. Acquisition must drive participation of hardware vendors to adopt new architectures and participation of software vendors who must provide required memory layouts, signatures, or control-flow graphs as needed.

Risk #4 - Evolving Technological Advances in Computing, Security, and Threats

Mitigation - Continuous analysis of evolving technologies and threats.

2.6 PROJECT MANAGEMENT

2.6.1 RESEARCH TEAM

Our research team includes three faculty from the UAH Department of Electrical and Computer Engineering, engineering support staff, and graduate student researchers.

- Dr. David J. Coe (PI) shall perform threat assessment and project management/reporting tasks.
- Dr. Jeffrey H. Kulick (Co-I) is the system architect.
- Dr. Aleksandar Milenkovic (Co-I) is our expert in execution tracing and performance measurement.

The members of our team have established research records in the areas of software safety and security, anti-tamper, embedded systems, FPGA coprocessors, and computing system performance analysis. Selected publications include [8-9] [17-28].

2.6.2 COST AND SCHEDULE ESTIMATES

An outline of the four phases of the research plan appears in Section 2.4 above. The technical challenges being addressed at each phase increase in complexity and effort required. Each phase includes an **investigations** subtask which performs preliminary research for the subsequent phase. The results of these investigations shall permit our team to refine the cost and schedule estimates as well as the technical goals for the upcoming phases.

Presented below in Figure 2.1 is a projected Plan of Work for Phase 1 and Phase 2. Our estimated budget for Phase 1 is \$300K and the estimated budget for Phase 2 \$400K.

	Year 1				Year 2				Year 3	
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2
Phase 1										
Threat Modeling										
Prototyping										
Assessment										
Investigations										
Phase 2										
Threat Modeling										
Prototyping										
Assessment										
Investigations										

Figure 2.1: Proposed plan of work for Phase 1 and Phase 2

Rough order of magnitude estimates for Phase 3 and Phase 4 are 18-24 months and \$600K per phase due to the increased technical challenges. The estimates for Phase 3 and Phase 4 shall be refined based upon what we learn in the first two phases.

2.6.3 RESEARCH MILESTONES AND METRICS

Each phase produces a proof-of-concept prototype that may be deployed in absence of subsequent phases. Project milestones for each phase include quarterly telecons with project sponsor and biannual status reports.

2.6.3.1 Midterm Exam for Each Phase

*The **Midterm Exam** for each phase shall be a demonstration of the assessment infrastructure which includes the threat to be mitigated and its delivery mechanism-- this establishes that the assessment criteria and mechanism are in place.*

2.6.3.2 Final Exam for Each Phase

*The **Final Exam** for each phase shall be a final demonstration of the mitigated threat with documentation and appropriate deliverables.*

2.7 REFERENCES

- [1] Kim Zetter, "A Cyberattack Has Caused Confirmed Physical Damage for the iSecond Time Ever," Wired, 01/08/15, <http://www.wired.com/2015/01/german-steel-mill-hack-destruction/>
- [2] <http://www.heritage.org/research/reports/2014/10/cyber-attacks-on-us-companies-in-2014>
- [3] David Z. Morris, "Hackers stole restricted F-35 data from Australian contractor," Fortune, October 14, 2017. <http://fortune.com/2017/10/14/hacked-f-35-data>
- [4] Dan Klinedinst, Joel Land, and Kyle O'Meara, *2017 Emerging Technology Domains Risk Survey*. CMU/SEI-2017-TR-008. Software Engineering Institute, Carnegie Mellon University. 2017. <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=505311>
- [5] Paul Lagarde, "FBI Expects Number of Cyber Attacks 'To Grow Exponentially' in Coming Years," CNSnews.com, May 23, 2014, 2:10pm, <http://www.cnsnews.com/news/article/paul-lagarde/fbi-expects-number-cyber-attacks-grow-exponentially-coming-years>
- [6] Metasploit, <https://www.metasploit.com/>
- [7] Tal Garfinkel and Mendel Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection", Network and Distributed System Security Symposium • February 2003.
- [8] Reece Johnston, Sun-il Kim, David J. Coe, Letha Etkorn, Jeffrey H. Kulick, and Aleksandar Milenkovic, "Xen Network Flow Analysis for Intrusion Detection," 11th Cyber and Information Security Research Conference, Oak Ridge, Tennessee, April 5-7, 2016.
- [9] David J. Coe, Jeffrey H. Kulick, Aleksandar Milenkovic, Sun-il Kim, and Letha Etkorn, "An Approach to Securing Cloud and Internet of Things Applications, 2016 National Cyber Summit, June 7-9, 2016.
- [10] Wikipedia, <https://en.wikipedia.org/wiki/Coprocessor>
- [11] <https://trustedcomputinggroup.org/tpm-main-specification/>
- [12] "Extracting a 3DES key from an IBM 4758," <http://www.cl.cam.ac.uk/~rnc1/descrack/ibm4758.html>
- [13] US Patent US 9600291 B1, "Secure boot using a field programmable gate array (FPGA), Altera Corporation, published March 21, 2017.
- [14] US Patent US 20150012737 A1, "Secure boot for unsecure processors," January 8, 2015.
- [15] Linton Salmon, "System Security Integrated Through Hardware and Firmware (SSITH)," DARPA Proposers Day Overview, April 21, 2017.

- [16] SoC and CPU System-Wide Approach to Security, <https://www.arm.com/products/security-on-arm/trustzone>, retrieved October 2017.
- [17] Amrish K. Tewar, Albert R. Myers, Aleksandar Milenković, "mcfTRaptor: Toward unobtrusive on-the-fly control-flow tracing in multicores," *Journal of Systems Architecture*, Vol. 61, No. 10, November 2015, pp. 601-614, doi: 10.1016/j.sysarc.2015.07.005.
- [18] Vladimir Uzelac, Aleksandar Milenković, Milena Milenković, Martin Burtscher, "Using Branch Predictors and Variable Encoding for On-the-fly Program Tracing," *IEEE Transactions on Computers*, Vol. 63, No. 4, April 2014, pp. 1008-1020, doi: 10.1109/TC.2012.267.
- [19] Ryan Cowart, David Coe, Jeffrey Kulick, and Aleksandar Milenkovic, "An Implementation and Experimental Evaluation of Hardware Accelerated Ciphers in All-Programmable SoCs" in ACM SE '17: SouthEast Conference.
- [20] Austin Rogers, Aleksandar Milenković, "Security extensions for integrity and confidentiality in embedded processors," *Microprocessors and Microsystems*, Vol. 33, Issues 5-6, pp. 398-414 (August 2009).
- [21] J.M. English, D.J. Coe, D. Hyde, R.K. Gaede, and J. Kulick, "MEMS-Assisted Cryptography for CPI Protection," *IEEE Security & Privacy*, vol. 5, issue 4, July-August 2007, pp. 14-21.
- [22] Jennifer English, David Coe, Rhonda Gaede, Jeffrey Kulick, "Protection of Cryptographic Systems Using Reconfigurable MEMS-Based Tamper Sensors," *Proceedings of the Reconfigurable Systems, Microsystems, and Nanotechnology Conference*, Redstone Arsenal, May 8, 2007.
- [23] J.H. Kulick, D.J. Coe, J.S. Hogue, and R.K. Gaede, "Cyber-physical Systems Design: The Software Safety Engineering & Model-Based Design Divide," *2011 Army/NASA System and Software Engineering Forum*, July 26-27, 2011, Huntsville, AL.
- [24] Michael C. Lindsey, David J. Coe, Jeffrey Kulick, Letha Etzkorn, and Yujian Fu, "Design of Safety Critical Survivable Systems Using Autonomic and Semantic Web Methodologies," *2012 International Conference on Software Engineering Research and Practice (SERP'12)*, WORLDCOMP 2012, July 16-19, 2012, Las Vegas, NV.
- [25] Travis Cleveland, David J. Coe, and Jeffrey H. Kulick "Video Processing for Motion Tracking of Safety Critical Systems," *2013 International Conference on Software Engineering Research and Practice (SERP'13)*, WORLDCOMP 2013, July 22-25, 2013, Las Vegas, NV.
- [26] David J. Coe and Jeffrey H. Kulick, "A Model-Based Agile Process for DO-178C Certification," *2013 International Conference on Software Engineering Research and Practice (SERP'13)*, WORLDCOMP 2013, July 22-25, 2013, Las Vegas, NV.
- [27] David J. Coe and Jeffrey H. Kulick, "Positive Train Control: Concepts, Implementations, and Challenges," *2014 International Conference on Software Engineering Research and Practice (SERP'14)*, WORLDCOMP 2014, July 21-24, 2014, Las Vegas, NV.
- [28] Jason D. Winningham, David J. Coe, and Jeffrey H. Kulick, "Agile Systems Integration Process," *2015 Frontiers in Education: Computer Science and Computer Engineering (FECS'15)*, WORLDCOMP 2015, July 27-30, 2015, Las Vegas, NV.

3.1 INTRODUCTION AND MOTIVATION

Sustained interest in distributed system architectures presents an important tradeoff in conceptual design. Distributed systems pursue superior performance compared to traditional monolithic systems through greater flexibility, robustness, and efficiency. For example, arguments for fractionated spacecraft systems emphasize elements of architectural flexibility in uncertain contexts as more information is gathered in operations, risk diversification across multiple systems, spatial distribution to mitigate failures or attacks, and lower cost for individual components (Brown and Eremenko, 2006). Similar arguments in the decentralization theorem in economics argues distributed systems can exhibit finer control to more efficiently match available resources with localized demands (Oates, 2008).

However, by definition, distributed architectures also introduce new interdependencies between constituent modules which can lead to overall system failure if not understood or anticipated. These effects are amplified by communication barriers if constituent systems are owned and operated by independent entities as a system-of-systems or federation-of-systems. In infrastructure, for example, cross-sector interdependencies can directly lead to cascading failures and loss of critical societal functions (e.g. Rinaldi, Peerenboom, and Kelly, 2001). This paradoxical relationship has been described as “robust yet fragile” in systems literature (Alderson and Doyle, 2010) and highlights a fundamental tradeoff between risk and reward which remains a critical area of research in systems engineering.

Engineers and decision-makers must understand the tradeoffs associated with alternative system architectures during early conceptual design activities to best inform concept selection and detailed design. However, the current approach of treating systems engineering as a centralized decision-making process is not appropriate for distributed system architectures due to the inherent lack of control. Applying existing value-centric and tradespace exploration methods to distributed systems only emphasizes the positive upsides of collective action and provides little analysis of the strategic incentives among interactive decision-makers. The objective of this project is to develop and evaluate a game-theoretic risk dominance metric and assessment method to compare monolithic and distributed system alternatives in the context of multi-architecture tradespace exploration.

Grogan et al. (2016) showed how federated systems—one type of distributed system—can be modeled as a Stag Hunt game where an independent (centralized) design is analogous to a payoff dominated equilibrium and a federated (distributed) design is analogous to a payoff dominant equilibrium. Results showed how Selten’s (1995) weighted average log measure (WALM) can assess strategic risk for two-player cases and demonstrated why the payoff-maximizing alternative may not be the most desirable choice. This project seeks to extend and demonstrate game-theoretic risk assessment approaches and, particularly, investigate how risk dominance measures similar to Selten’s WALM can evaluate alternative architectures in more general design cases. This report documents progress on initial tasks to formulate and illustrate the risk assessment method and identifies further extensions as future work.

3.2 VALUE-CENTRIC DESIGN FOR DISTRIBUTED SYSTEMS

Systems engineering increasingly relies on value-centric methods to guide conceptual design activities. These methods build on rational decision-making theory to guide design decisions to maximize system

value (Collopy and Hollingsworth, 2011) rather than minimizing cost to meet requirements. Traditional tradespace exploration methods enumerate a design space and evaluate one or more design attributes including value/utility to visualize a set of alternatives (Ross et al. 2004). The central feature of value-centric methods is a value function in Eq. (1) to map elements from a design space \mathcal{D} to a scalar quantity interpreted as utility, net present value, or some other measure of preference.

$$V: \mathcal{D} \rightarrow \mathbb{R}, \quad V = V(d) \quad (1)$$

Value functions for distributed architectures benefit from an alternative design space representation. A revised form in Eq. (2) maps a composite design space with up to n constituent systems, each with its own design space \mathcal{D}_i , to a scalar value. From this perspective, the distributed architecture may be large, but not fundamentally different from a centralized architecture.

$$V: \left(\prod_{i=1}^n \mathcal{D}_i \right) \rightarrow \mathbb{R}, \quad V = V(d_1, \dots, d_n) \quad (2)$$

However, two additional issues arise in valuing distributed architectures and particularly from distributed control or autonomy among constituent systems. First, well-established results such as Arrow's Impossibility Theorem clearly explain how and why group preferences cannot generally be transformed into consistent aggregate preferences (Hazelrigg, 1996). Rather, one must maintain N individual value functions by mapping the design space to a vector value in Eq. (3). Multi-actor value functions lead to strategic behaviors and generally limit the existence of a globally optimal solution.

$$V: \left(\prod_{i=1}^n \mathcal{D}_i \right) \rightarrow \mathbb{R}^N, \quad V_i = V_i(d_1, \dots, d_n) \quad (3)$$

Second, the focus of this project, distributed architectures introduce interactive effects for individual values V_i as a function of others' decisions $d_{j \neq i}$. Essential uncertainty of others' decisions contributes a fundamental source of risk in distributed systems: the system may not perform as anticipated due to the inherent lack of control across system boundaries. Even in cases where a distributed system is controlled by a single central actor, anomalous operations exhibit similar dynamics where the anticipated design does not perform as expected due to failures of individual components.

Traditional systems engineering methods treat risk as uncertainty or variation in value which can be analyzed with Monte Carlo sampling of a stochastic value function. However, this approach is computationally intense, particularly for distributed systems, due to combinatorial factors of the design space coupled with pairwise interactions between interdependent components. More importantly, however, this type of risk should not be considered as an explicit attribute to be traded during concept evaluation, but rather only as uncertainty on other attributes (Abbas and Cadenbach, 2016).

In contrast, this work seeks to codify strategic risk as a type of *design instability* instead of uncertainty on value. Even designs with high expected value may be unstable due to interactive effects. Risk analysis should be a fundamental part of conceptual design; however, it may presently be omitted in tradespace analysis due to computational tractability and methodological issues. This leads to overly optimistic and fragile results which focus only on the upside potential of distributed systems without sufficient recognition of the downside risk contributed by additional interdependencies. Program managers and systems engineers will benefit from the development of quantitative risk metrics and associated methodologies for their use in multi-architectural tradespace analysis.

3.3 GAME THEORY AND EQUILIBRIUM SELECTION

Game theory is a branch of economics which studies strategic decision-making among multiple interacting players. A game is defined by a set of available decisions (strategies) \mathcal{C}_i for each player and corresponding payoffs or resulting utilities under interactive effects. The value function in Eq. (4) maps strategy decisions for each player to utilities for each player.

$$V: \left(\prod_{i=1}^N \mathcal{C}_i \right) \rightarrow \mathbb{R}^N, \quad V_i = V_i(c_1, \dots, c_N) \quad (4)$$

Binary games only consider two strategies per player, i.e. $\mathcal{C}_i = \{\phi_i, \psi_i\}$. Table 2 shows the normal form notation of a binary game with $N = 2$ players and payoffs V_i resulting from various strategy pairs.

Table 2. Normal Form for a Binary Strategic Game and an Example Stag Hunt Game

	ϕ_2	ψ_2		<i>Hare</i>	<i>Stag</i>
ϕ_1	$V_1(\phi_1, \phi_2)$ $V_2(\phi_1, \phi_2)$	$V_1(\phi_1, \psi_2)$ $V_2(\phi_1, \psi_2)$	<i>Hare</i>	2 2	4 0
ψ_1	$V_1(\psi_1, \phi_2)$ $V_2(\psi_1, \phi_2)$	$V_1(\psi_1, \psi_2)$ $V_2(\psi_1, \psi_2)$	<i>Stag</i>	0 4	5 5

Nash equilibria are stable strategy sets under interactive effects and often considered solutions to games because of this stability feature. Bipolar games are a subclass of binary games with two Nash equilibria defined by strategies $\phi = (\phi_1, \dots, \phi_N)$ and $\psi = (\psi_1, \dots, \psi_N)$. A Stag Hunt is a classic example of a two-player bipolar game in Table 2. Strategy pairs where both players choose hare (ϕ_1, ϕ_2) and both choose stag (ψ_1, ψ_2) are Nash equilibria. In the above case, a common stag strategy yields 5 payoff for each player while a common hare strategy yields only 2 each. However, a player can pursue a hare alone and earn a total payoff of 4 while pursuing stag alone sends a player home hungry. Both strategies are equally stable from an equilibrium perspective but have different payoff and risk characteristics.

Harsanyi and Selten (1988) and Selten (1995) develop theory for equilibrium selection in bipolar games based on the concept of risk dominance. Similar to how some equilibria may exhibit payoff dominance, risk dominance objectively captures resistance to losses. Selten (1995) proposes a quantitative metric to measure risk dominance in bipolar games with linear incentives called the weighted average log measure (WALM). A positive WALM indicates the payoff dominated equilibrium is risk dominant while a negative WALM indicates the payoff dominant equilibrium is also risk dominant.

Schmidt et al. (2003) simplify WALM to the form in Eq. (5) for games with two symmetric players.

$$R(\phi, \psi) = \ln \left(\frac{V_i(\phi_i, \phi_j) - V_i(\psi_i, \phi_j)}{V_i(\psi_i, \psi_j) - V_i(\phi_i, \psi_j)} \right) \quad (5)$$

For the Stag Hunt game in Table 2, WALM is shown to be positive in Eq. (6) indicating ϕ risk dominates ψ . This result can be interpreted as the large losses associated with deviating from ϕ provides more stability than the smaller deviation loss associated with ψ . In the absence of subjective information, players may reasonably choose to hunt hare as the risk-dominant strategy.

$$R(\phi, \psi) = \ln \left(\frac{2-0}{5-4} \right) = \ln \left(\frac{2}{1} \right) \approx 0.69 \quad (6)$$

For greater generalizability to asymmetric games with $N \geq 2$ players, Selten's WALM of risk dominance is defined in terms of an abstraction called a *biform* uniquely determined by an influence matrix A capturing

interdependencies between players and a vector of normalized deviation losses u . Together, these factors assign losses to deviations from a baseline strategy. Equation (7) defines WALM risk dominance for generalized games (Selten, 1995) where w_i represent influence weights for each player.

$$R(\phi, \psi) = \sum_{i=1}^N w_i(A) \ln\left(\frac{u_i}{1-u_i}\right) \quad (7)$$

At its core, risk dominance deals with resistance to deviation losses L_i experienced from switching strategies in Eq. (8) where the notation $\phi = (\phi_1, \dots, \phi_N)$ indicates all players choose strategy ϕ and ϕ_{-i} indicates all except player i choose strategy ϕ . Formally, deviation losses may be subject to complex behavioral factors such as diminishing sensitivity and loss aversion (Tversky and Kahneman, 1992); however, for clarity this work expresses deviation losses as a simple difference between payoff values.

$$L_i(\phi) \propto V_i(\phi) - V_i(\phi_{-i}), \quad L_i(\psi) \propto V_i(\psi) - V_i(\psi_{-i}) \quad (8)$$

Normalized deviation losses u_i in Eq. (9) transform the quantity to a unit scale.

$$u_i = \frac{L_i(\phi)}{L_i(\phi) + L_i(\psi)}, \quad 1 - u_i = \frac{L_i(\psi)}{L_i(\phi) + L_i(\psi)} \quad (9)$$

Influence weights measure the importance of one player on others' stability. In the class of games studied by Selten, the weights $w_i(A)$ are a function of an influence matrix A which captures the degree of influence players have on each other for choosing between strategies. Weights can be interpreted as the eigenvector of A^T rescaled to unit norm corresponding to the eigenvalue of 1 which is equivalent to the stationary stochastic distribution for the Markov chain with state transition probabilities a_{ij} .

The only possible solution for two player games is $w_i(A) = w_j(A) = 0.5$ which leads to the Schmidt et al.'s simplification in Eq. (5) for symmetric players. However, for more general cases, influence matrix elements a_{ij} can be computed in Eq. (10) by measuring how deviations in player j 's strategy influence player i 's payoff relative to player i 's own effect via global deviation losses.

$$a_{ij} = \frac{l_{ij}(\phi) + l_{ij}(\psi)}{L_i(\phi) + L_i(\psi)} \quad (10)$$

Pairwise deviation losses l_{ij} are computed in Eq. (11) by considering combinations of players where $\mathcal{K} = \mathcal{P}(\{1, \dots, N\} \setminus \{i, j\})$ is the power set of all players except i and j with cardinality $|\mathcal{K}| = 2^{N-2}$.

$$l_{ij}(\phi) \propto \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} (V_i(\phi_{-k}) - V_i(\phi_{-kj})), \quad l_{ij}(\psi) \propto \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} (V_i(\psi_{-k}) - V_i(\psi_{-kj})) \quad (11)$$

3.4 MULTI-ACTOR VALUE FUNCTIONS AND STRATEGIC DESIGN GAMES

A strategic design game based on the multi-actor design framework in Grogan et al. (2016) distinguishes between two levels of decisions: strategy decisions $c_i \in \mathcal{C}$ govern collective behavior between actors and design decisions $d_i \in \mathcal{D}_i$ specify system configurations. A corresponding multi-actor value function for N design actors and n design decisions in Eq. (12) maps design and strategy decisions for each actor to a vector of real-number values interpreted as von Neumann-Morgenstern utilities.

$$V: \left(\prod_{i=1}^n \mathcal{D}_i \right) \times \mathcal{C}^N \rightarrow \mathbb{R}^N, \quad V_i = V_i^{c_1, \dots, c_N}(d_1, \dots, d_n) \forall i \quad (12)$$

While the design spaces \mathcal{D}_i may be large or unbounded and unique to each actor, the strategy space $\mathcal{C} = \{\phi, \psi\}$ is limited to a few options common to all actors, namely choosing between independent action (ϕ)

or collective action (ψ). In the context of distributed systems, independent means monolithic, disconnected, or defective while collective means distributed, connected, or anticipated.

A strategic design game benefits from a few special conditions. For simplicity, this section assumes actor i controls strategy decision c_i and design decision d_i such that $N = n$. If there are no interaction effects between independent players, described by Grogan et al. (2016) as a *perfectly limited* case, the multi-actor value function can be replaced by a single-actor value function in Eq. (13) when player i chooses an independent strategy $c_i = \phi$ or all other players choose an independent strategy $c_j = \phi \forall j \neq i$.

$$V_i^{c_1, \dots, c_N}(d_1, \dots, d_n) \approx V_i(d_i) \text{ if } c_i = \phi \text{ or } c_j = \phi \forall j \neq i \quad (13)$$

Perfectly limited cases allow local design optimization in Eq. (14) under an independent strategy.

$$\mathcal{V}_i^\phi = \max_{d \in \mathcal{D}_i} V_i(d_i), \quad d_i^\phi = \arg \max_{d \in \mathcal{D}_i} V_i(d_i) \quad (14)$$

A second notational simplification in Eq. (15) aggregates all players participating in a collective strategy.

$$V_i^{c_1, \dots, c_N}(d_1, \dots, d_n) \approx V_i^\psi(\{d_k: c_k = \psi\}) \quad (15)$$

Thus, a strategic design game can be represented using multi-actor value functions parameterized by candidate collective designs (d_1, \dots, d_n) using the simplified notation above for independent and collective alternatives. Table 3 shows a normal form game parameterized for $N = 3$ players.

Table 3. Normal Form Perfectly Limited Strategic Design Game for Three Players

	$c_2 = \phi$		$c_2 = \psi$			$c_2 = \phi$		$c_2 = \psi$	
$c_1 = \phi$	\mathcal{V}_1^ϕ	\mathcal{V}_2^ϕ	\mathcal{V}_3^ϕ	V_1^ϕ	$V_2(d_2)$	\mathcal{V}_3^ϕ	\mathcal{V}_1^ϕ	\mathcal{V}_2^ϕ	\mathcal{V}_3^ϕ
$c_1 = \psi$	$V_1(d_1)$	\mathcal{V}_2^ϕ	\mathcal{V}_3^ϕ	$V_1^\psi(d_1, d_2)$	$V_2^\psi(d_1, d_2)$	\mathcal{V}_3^ϕ	$V_1^\psi(d_1, d_3)$	\mathcal{V}_2^ψ	\mathcal{V}_3^ψ
	$c_3 = \phi$					$c_3 = \psi$			

Using the strategic design game notation, simplified deviation loss functions are redefined in Eq. (16).

$$L_i(\phi) \propto \mathcal{V}_i^\phi - V_i(d_i), \quad L_i(\psi) \propto V_i^\psi(d_1, \dots, d_n) - \mathcal{V}_i^\phi \quad (16)$$

For games with $N = 3$ players, as in Table 3, the pairwise deviation loss functions reduce to Eq. (17).


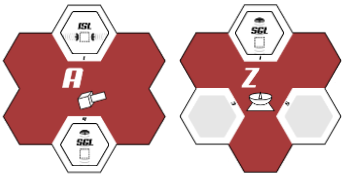
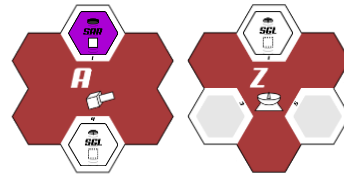
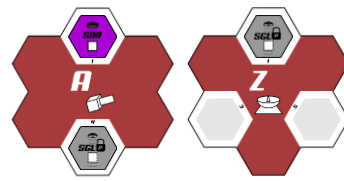
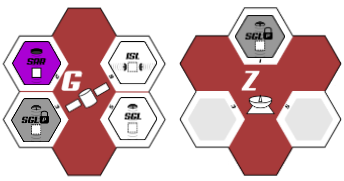
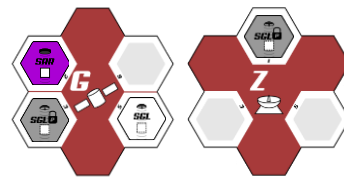
$$l_{ij}(\phi) \propto 0, \quad l_{ij}(\psi) \propto \frac{1}{2} \left[\left(V_i^\psi(d_i, d_j) - V_i(d_i) \right) + \left(V_i^\psi(d_i, d_j, d_k) - V_i^\psi(d_i, d_k) \right) \right] \quad (17)$$

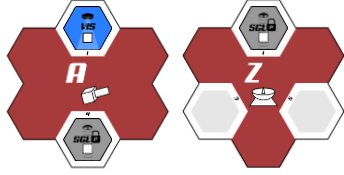
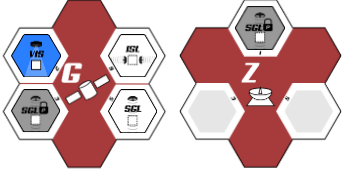
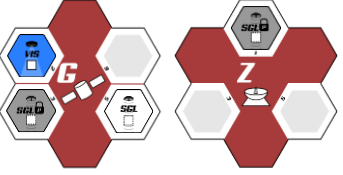
3.5 EXAMPLE APPLICATION CASE

To illustrate the proposed method to assess risk dominance in distributed systems, consider the following application case based on *Orbital Federates*, a stylized model of distributed Earth-observing space systems (Grogan and de Weck, 2015) paired with an existing simulation implemented in Python. The simulation model acts as a multi-actor value function by mapping design and strategy sets (inputs) to net present value earned by each player over a simulated lifetime (outputs). The model includes stochastic features to capture uncertainty in demands such that results must be sampled using Monte Carlo methods. While many model details are clearly fictional, *Orbital Federates* has been developed with similar features to space systems to help understand strategic player behavior.

The design scenario in Table 4 considers $N = 3$ players who design, build, and operate space systems to collect and downlink data to satisfy demands and earn revenue. Each player chooses between a monolithic strategy of independent operations and a distributed strategy allowing exchange of data services. The monolithic alternative excludes participation by player 1 and includes small standalone observing spacecraft for players 2 and 3 who specialize in synthetic aperture radar (SAR) and visual light (VIS) sensors, respectively. Distributed designs consider an opportunistic data exchange policy where inter-satellite link (ISL) and space-to-ground (SGL) services are priced at a fixed cost of 100. Distributed design A includes participation by player 1 with a data relay spacecraft and SGL receiver and ISL adoption among all three players. Distributed design B eliminates the ISL technology option and establishes an independent observing spacecraft for player 1 with the SGL receiver.

Table 4. Orbital Federates Design Space

Player	Monolithic Design	Distributed Design A	Distributed Design B
1	 <p>None.</p> $V_1^\phi = 0$	 <p>Small satellite with inter-satellite link (ISL) and open space-to-ground link (oSGL). Ground station with oSGL.</p> $V_1(d_{1A}) = -1100$ $V_1^\psi(d_{1A}, d_{2A}) = -572$ $V_1^\psi(d_{1A}, d_{3A}) = -653$ $V_1^\psi(d_{1A}, d_{2A}, d_{3A}) = 100$	 <p>Small satellite with SAR sensor and open space-to-ground link (oSGL). Ground station with oSGL.</p> $V_1(d_{1B}) = -104$ $V_1^\psi(d_{1B}, d_{2B}) = 83$ $V_1^\psi(d_{1B}, d_{3B}) = 33$ $V_1^\psi(d_{1B}, d_{2B}, d_{3B}) = 271$
2	 <p>Small satellite with SAR sensor and proprietary space-to-ground link (pSGL). Ground station with pSGL.</p> $V_2^\phi = -39$	 <p>Medium satellite with SAR sensor, inter-satellite link (ISL) and proprietary and open space-to-ground links (pSGL and oSGL). Ground station with pSGL.</p> $V_2(d_{2A}) = -389$ $V_2^\psi(d_{1A}, d_{2A}) = 440$ $V_2^\psi(d_{2A}, d_{3A}) = -255$ $V_2^\psi(d_{1A}, d_{2A}, d_{3A}) = 536$	 <p>Medium satellite with SAR sensor and proprietary and open space-to-ground links (pSGL and oSGL). Ground station with pSGL.</p> $V_2(d_{2B}) = -288$ $V_2^\psi(d_{1B}, d_{2B}) = 141$ $V_2^\psi(d_{2B}, d_{3B}) = -288$ $V_2^\psi(d_{1B}, d_{2B}, d_{3B}) = 150$

3	 <p>Small satellite with VIS sensor and proprietary space-to-ground link (pSGL). Ground station with pSGL. $V_3^\phi = 30$</p>	 <p>Medium satellite with VIS sensor, inter-satellite link (ISL) and proprietary and open space-to-ground links (pSGL and oSGL). Ground station with pSGL. $V_3(d_{3A}) = -320$ $V_3^\psi(d_{1A}, d_{3A}) = 386$ $V_3^\psi(d_{2A}, d_{3A}) = -158$ $V_3^\psi(d_{1A}, d_{2A}, d_{3A}) = 639$</p>	 <p>Medium satellite with VIS sensor and proprietary and open space-to-ground links (pSGL and oSGL). Ground station with pSGL. $V_3(d_{3B}) = -220$ $V_3^\psi(d_{1B}, d_{3B}) = 391$ $V_3^\psi(d_{2B}, d_{3B}) = -161$ $V_3^\psi(d_{1B}, d_{2B}, d_{3B}) = 378$</p>
---	--	---	--

Value function outputs shown in Table 4 evaluate expected net present value over a 24-turn game using a discount rate of 2% per turn sampled using 1000 seeded runs of the Orbital Federates Simulation – Python (OFSPY) executable. The monolithic design requires only one value function evaluation while each distributed design requires five function evaluations to consider all combinations of two or more cooperating players and the single case with zero or one cooperating players. Preprocessing configures OFSPY inputs to consider favorable spatial locations for player’s spacecraft for each function evaluation.

3.5.1 ANALYSIS OF DISTRIBUTED ALTERNATIVE A

This section analyses distributed design A compared to the monolithic alternative. Table 5 frames a strategic design game for the distributed design alternatives A. Note that this design scenario constitutes a bipolar game with Nash equilibria (ϕ, ϕ, ϕ) and (ψ, ψ, ψ) .

Table 5. Strategic Design Game for Distributed Designs A

	$c_2 = \phi$		$c_2 = \psi$			$c_2 = \phi$		$c_2 = \psi$	
$c_1 = \phi$	0	-39	0	-389	0	-39	0	-255	-158
		30		30		-320		-158	
$c_1 = \psi$	-1100	-39	-572	440	-652	-39	100	536	639
		30		30		386		639	
	$c_3 = \phi$					$c_3 = \psi$			

From the above data, the deviation losses can be calculated using Eq. (16) with results in Eq. (18).

$$[L_i(\phi)] = \begin{bmatrix} L_1(\phi) \\ L_2(\phi) \\ L_2(\phi) \end{bmatrix} = \begin{bmatrix} 1100 \\ 350 \\ 350 \end{bmatrix}, \quad [L_i(\psi)] = \begin{bmatrix} L_1(\psi) \\ L_2(\psi) \\ L_2(\psi) \end{bmatrix} = \begin{bmatrix} 100 \\ 575 \\ 609 \end{bmatrix} \quad (18)$$

Pairwise deviation losses can be calculated using Eq. (17) with results in Eq. (19).

$$l_{ij}(\phi) = 0 \forall i \neq j, \quad [l_{ij}(\psi)] = \begin{bmatrix} 0 & l_{12}(\psi) & l_{13}(\psi) \\ l_{21}(\psi) & 0 & l_{23}(\psi) \\ l_{31}(\psi) & l_{32}(\psi) & 0 \end{bmatrix} = \begin{bmatrix} 0 & 640 & 560 \\ 810 & 0 & 115 \\ 751.5 & 207.5 & 0 \end{bmatrix} \quad (19)$$

Next, the interaction matrix can be calculated using Eq. (10) with results in Eq. (20).

$$A = \begin{bmatrix} 0 & a_{12} & a_{13} \\ a_{21} & 0 & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.533 & 0.467 \\ 0.876 & 0 & 0.124 \\ 0.783 & 0.217 & 0 \end{bmatrix} \quad (20)$$

Eigenvector analysis of the transposed influence matrix A^T yields the eigenvector corresponding to the eigenvalue of 1 (rescaled to unit norm) in Eq. (21).

$$w(A) = \begin{bmatrix} w_1(A) \\ w_2(A) \\ w_3(A) \end{bmatrix} = \begin{bmatrix} 0.455 \\ 0.296 \\ 0.249 \end{bmatrix} \quad (21)$$

Bringing these results together in Eq. (22), the result shows the monolithic strategy ϕ is risk dominant.

$$\begin{aligned} R_\psi(d_{1A}, d_{2A}, d_{3A}) &= w_1(A) \ln\left(\frac{L_1(\phi)}{L_1(\psi)}\right) + w_2(A) \ln\left(\frac{L_2(\phi)}{L_2(\psi)}\right) + w_3(A) \ln\left(\frac{L_3(\phi)}{L_3(\psi)}\right) \\ &= 0.455 \ln\left(\frac{1100}{100}\right) + 0.296 \ln\left(\frac{350}{575}\right) + 0.249 \ln\left(\frac{350}{609}\right) = 0.80 \end{aligned} \quad (22)$$

Analysis of distributed design A shows the monolithic strategy is more stable for player 1 because of large downside losses incurred if other players chose independent action. This risk arises because player 1 has no independent source of revenue to recover the high cost of the distributed design. However, the distributed strategy is more stable for players 2 and 3 because large upside gains realized from shared data services outweigh the additional cost of larger spacecraft and additional hardware. Influence analysis identifies player 1 as the most influential which can be explained by their central role in both providing shared inter-satellite link (ISL) relay and space-to-ground (SGL) downlink services via the spacecraft and ground station, respectively. Players 2 and 3 have similar weights of 0.296 and 0.249 reflecting their symmetry in operational mission. Given player 1's aversion to the distributed strategy and strong influence, the monolithic strategy is risk dominant and the distributed strategy is prone to failure due to disengagement by player 1.

3.5.2 ANALYSIS OF DISTRIBUTED ALTERNATIVE B

Design B takes two actions to relieve strategic risk and attempt to make the distributed strategy risk dominant. First, it reduces the scope (and potential payoffs) of the distributed strategy by eliminating inter-satellite link (ISL) services to reduce the initial cost of spacecraft and hardware. Second, it provides player 1 an independent source of revenue by hosting an instrument to complete data contracts.

Table 6 frames a strategic design game for the distributed design alternatives B. Note that this design scenario constitutes a bipolar game with Nash equilibria $\phi = (\phi, \phi, \phi)$ and $\psi = (\psi, \psi, \psi)$.

Table 6. Strategic Design Game for Distributed Designs B

	$c_2 = \phi$		$c_2 = \psi$			$c_2 = \phi$		$c_2 = \psi$	
$c_1 = \phi$	0	-39	0	-288	0	-39	0	-288	-161
		30		30		-220		-161	
$c_1 = \psi$	-104	-39	83	141	33	-39	271	150	378
		30		30		391		378	
	$c_3 = \phi$					$c_3 = \psi$			

From the above data, the deviation losses can be calculated using Eq. (16) with results in Eq. (23).

$$[L_i(\phi)] = \begin{bmatrix} L_1(\phi) \\ L_2(\phi) \\ L_2(\phi) \end{bmatrix} = \begin{bmatrix} 104 \\ 249 \\ 250 \end{bmatrix}, \quad [L_i(\psi)] = \begin{bmatrix} L_1(\psi) \\ L_2(\psi) \\ L_2(\psi) \end{bmatrix} = \begin{bmatrix} 271 \\ 189 \\ 348 \end{bmatrix} \quad (23)$$

Pairwise deviation losses can be calculated using Eq. (17) with results in Eq. (24).

$$l_{ij}(\phi) = 0 \quad \forall i \neq j, \quad [l_{ij}(\psi)] = \begin{bmatrix} 0 & l_{12}(\psi) & l_{13}(\psi) \\ l_{21}(\psi) & 0 & l_{23}(\psi) \\ l_{31}(\psi) & l_{32}(\psi) & 0 \end{bmatrix} = \begin{bmatrix} 0 & 212.5 & 162.5 \\ 433.5 & 0 & 4.5 \\ 575 & 23 & 0 \end{bmatrix} \quad (24)$$

Next, the interaction matrix can be calculated using Eq. (10) with results in Eq. (25).

$$A = \begin{bmatrix} 0 & a_{12} & a_{13} \\ a_{21} & 0 & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0.566 & 0.434 \\ 0.989 & 0 & 0.011 \\ 0.961 & 0.039 & 0 \end{bmatrix} \quad (25)$$

Eigenvector analysis of the transposed influence matrix A^T yields the eigenvector corresponding to the eigenvalue of 1 (rescaled to unit norm) in Eq. (26).

$$w(A) = \begin{bmatrix} w_1(A) \\ w_2(A) \\ w_3(A) \end{bmatrix} = \begin{bmatrix} 0.494 \\ 0.288 \\ 0.218 \end{bmatrix} \quad (26)$$

Bringing these results together in Eq. (27), the result shows the distributed strategy ψ is risk dominant.

$$\begin{aligned} R_\psi(d_{1B}, d_{2B}, d_{3B}) &= w_1(A) \ln\left(\frac{L_1(\phi)}{L_1(\psi)}\right) + w_2(A) \ln\left(\frac{L_2(\phi)}{L_2(\psi)}\right) + w_3(A) \ln\left(\frac{L_3(\phi)}{L_3(\psi)}\right) \\ &= 0.494 \ln\left(\frac{104}{271}\right) + 0.288 \ln\left(\frac{249}{189}\right) + 0.218 \ln\left(\frac{250}{348}\right) = -0.46 \end{aligned} \quad (27)$$

Analysis of distributed design B shows the distributed strategy is more stable for players 1 and 3 and the monolithic strategy is more stable for player 2. The goals of reducing upfront costs and providing a second source of revenue successfully changed the strategic risk posture of player 1. However, the loss of the inter-satellite link (ISL) relay services tips player 2 towards a stable independent solution. Influence analysis still identifies player 1 as the most influential with weight 0.494. Similar to design A, this can be explained by the player's central role in providing shared downlink services via the ground station and the influence is greater than in design A because other players lack the relay components to interact with each other directly. Combining these factors, player 1's contributions outweigh player 2's contributions and the distributed strategy is risk dominant. In the event that player 2 disengages, players 1 and 3 still enjoy moderate returns from the distributed strategy.

3.5.3 DISCUSSION OF RESULTS

This analysis illustrates how "robust-yet-fragile" features can emerge from engineering design and how strategic risk assessment can mitigate fragility. If successful, distributed design A provides superior value for all three players by taking advantage of new technology and operational concepts. However, focusing on maximizing upside potential can yield unstable design solutions prone to failure. Player 1 experiences greater potential losses from distributed design A compared to others and is most likely to disengage from a distributed strategy. Distributed design B reduces the level of technological ambition and establishes an independent value stream. While its potential payoffs are smaller than design A, design B exhibits superior strategic stability and is robust to disengagement by player 2, the most likely to disengage from a

distributed strategy. These results echo Maier's (1998) principles for system-of-system architecting emphasizing stable intermediate forms and ensuring cooperating among all actors.

There are several extensions to this analysis which could be studied further. This case only investigates comparative analysis between two distributed architectures under a single collective strategy; however, similar analysis could be applied to a tradespace of a potential architectures *or* collective strategies. For example, the players could enumerate a set of alternative design architectures and several collective strategies with varying policies governing pricing, link availability, or other features to compare with the monolithic alternative which can be analyzed separately using existing systems engineering optimization methods. As a scalar metric, WALM risk dominance can compare the degree of stability (or instability) in a candidate architecture and strategy pair relative to a baseline monolithic or independent alternative.

A multi-actor value function is the key requirement for WALM risk dominance analysis. However, complete analysis requires $2^N - N$ function evaluations to assess each collective architecture/strategy pair. Thus, care must be taken to narrow the candidate architecture/strategy set to a minimum size to avoid lengthy computational burdens, especially when considering a stochastic model such as OFSPY.

3.6 EXTENSIONS AND OPPORTUNITIES

As demonstrated in the example application case, strategic design games and measures of risk dominance can inform concept selection in systems engineering. These methods help to assess inherent risks in decentralized architectures resulting from strategic instabilities of aligning decisions across independent actors. By applying WALM risk dominance analysis during conceptual design, systems engineers can identify, avoid, or rework distributed design alternatives with high but unstable payoffs compared to monolithic alternatives. This perspective may help to avoid costly development programs having structural problems likely leading to schedule and cost growth and, ultimately, cancellation. If scaled up to larger application cases, this research project could change how distributed systems are conceived and evaluated in conceptual design. Future work must proceed in two parallel directions.

First, additional theoretical work is required to assess the significance of violating assumptions of linear incentives or consider alternative metric definitions for cases with nonlinear incentives. Linear assumptions are unlikely to hold in most engineering projects because marginal contribution of each player tend to increase with collective size (i.e. there are increasing returns to scale in many distributed systems). Future work is also required to identify rigorous but practical means to quantify deviation losses in accordance with rational and behavioral economic theory. The simplified method of taking the value difference employed in this project may be adequate; however, additional exploration is required.

Second, additional practical or applied work is required to validate the proposed method in a realistic system context. In particular, a real-world system must be identified as being a bipolar game to benefit from risk dominance measures; however, many distributed systems are likely representable as bipolar games due to the framing of upside potential and downside risks in a strategic design game. Furthermore, a real-world application case would critically rely on a multi-actor value function to quantify value of varying levels of participation. Due to the relative novelty of this problem framing, a subsequent research project would likely require additional effort to either develop a suitable model or modify an existing one (e.g. using preprocessing and/or post-processing) to yield required outputs.

As a theoretically-grounded project, much of the risk of a future research project has been minimized through this incubator project. The overall theory of equilibrium selection appears internally consistent

and complicated but understandable to engineering managers. Quantities such as deviation losses, influence matrix elements, and weighting factors can be calculated for a general class of problems using the equations developed in this project. Remaining risks mostly deal with the framing of a particular design problem as a bipolar game, availability of a multi-actor value function, and combinatorial scaling. Given the successes of the formulation of WALM risk dominance in simplified but not trivial problems similar to *Orbital Federates*, the next step of research would scale up to a more realistic system.

Provided access to an existing or actively-developed value model, an appropriate level of support would total approximately \$350k over two years to support a full-time research assistant, travel for sponsor meetings and presentation of results at domain conferences, computation time on cloud platforms, and release time for the principal investigator. A midpoint review would establish the validity of a multi-actor value function to assess preference for multiple players and the overall framing of the design problem as a bipolar game. A final review would establish the validity of a game-theoretic risk analysis to assess a multi-architecture tradespace of candidate designs and collective strategies for distributed systems compared to a monolithic or independent alternative.

There exists potential synergy with a related project at NASA's Goddard Space Flight Center led by Dr. Jacqueline Le Moigne. The Tradespace Analysis Tool for Constellations (TAT-C) is a simulation modeling tool to quantify the performance of distributed spacecraft missions (Le Moigne et al., 2017). In the terminology of this project, it serves as distributed system value function similar to Eq. (2). While it presently addresses a subset of needs for strategic risk assessment, potential extensions may be able to transform it into a multi-actor value function to assess collective strategies for Earth-observing spacecraft missions. There exist some interesting existing application cases, such as the Afternoon Constellation, a collection of spacecraft which gather spatially and temporally correlated datasets, and potential coordination between U.S. and European medium-resolution land imaging platforms (Landsat and Sentinel-2) which gather overlapping data using similar orbital geometries to benefit from cross-validation. A strategic risk assessment method similar to that described in this project may help to identify stable designs among national and international government agencies and commercial partners.

Distributed system architectures provide an opportunity to achieve greater performance compared to monolithic alternatives; however, this project demonstrates a careful analysis of strategic risk is critical to identify stable design alternatives. Game-theoretic methods such as Selten's WALM risk dominance can help systems engineers assess and mitigate threats to distributed architectures during conceptual design. Understanding of both the upside potential of nominal operations as well as the downsides associated with failures or anomalous operations help assess sources of strategic risk. With additional support and development, this research has the potential to transform how distributed systems are perceived during conceptual design and avoid pursuing inherently unstable concepts.

3.7 REFERENCES

- [1] O. Brown and P. Eremenko, "The Value Proposition for Fractionated Space Architectures," *Space 2006*, San Jose, CA, 2006.
- [2] W.E. Oates, "On the Theory and Practice of Fiscal Decentralization," A.J. Auerbach and D.N. Shaviro (Eds.), *Institutional Foundations of Public Finance: Economic and Legal Perspectives*, Cambridge, MA: Harvard University Press, pp. 165-189, 2008.
- [3] S.M. Rinaldi, J.P. Peerenboom, and T.K. Kelly, "Identifying, Understanding, and Analyzing Critical Infrastructure Interdependencies," *IEEE Control Systems Magazine*, 11-25, 2001.

- [4] D.L. Alderson and J.C. Doyle, "Contrasting Views of Complexity and Their Implications for Network-centric Infrastructures," *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 40(4):839-852, 2010.
- [5] P.T. Grogan, K. Ho, A. Golkar, and O.L. de Weck, "Multi-actor Value Modeling for Federated Systems," *IEEE Systems Journal*, Early access, 2016.
- [6] R. Selten, "An Axiomatic Theory of a Risk Dominance Measure for Bipolar Games with Linear Incentives," *Games and Economic Behavior*, 8(1):213-263, 1995.
- [7] P.D. Collopy and P.M. Hollingsworth, "Value-Driven Design," *Journal of Aircraft*, 48(3):749-759, 2011.
- [8] A.M. Ross, D.E. Hastings, J.M. Warmkessel and N.P. Diller, "Multi-Attribute Tradespace Exploration as Front End for Effective Space System Design," *Journal of Spacecraft and Rockets*, 41(1):20-28, 2004.
- [9] G.A. Hazelrigg, "The Implications of Arrow's Impossibility Theorem on Approaches to Optimal Engineering Design," *Journal of Mechanical Design*, 118:161-164, 1996.
- [10] A.E. Abbas and A.H. Cadenbach, "On the Use of Utility Theory in Engineering Design," *IEEE Systems Journal*, 2016. Early access.
- [11] J.C. Harsanyi and R. Selten, *A General Theory of Equilibrium Selection in Games*, MIT Press, Cambridge, MA, 1988.
- [12] D. Schmidt, R. Shupp, J.M. Walker, and E. Ostrom, "Playing safe in coordination games: the roles of risk dominance, payoff dominance, and history of play," *Games and Economic Behavior* 42(X):281-299, 2003.
- [13] A. Tversky and D. Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty," *Journal of Risk and Uncertainty*, 5(4):297-323, 1992.
- [14] P.T. Grogan and O.L. de Weck, "Interactive Simulation Games to Assess Federated Satellite System Concepts," *2015 IEEE Aerospace Conference, Big Sky, MT*, 2015.
- [15] M.W. Meier, "Architecting principles for systems-of-systems," *Systems Engineering*, 1(4):267-284, 1998.
- [16] J. Le Moigne, P. Dabney, O. de Weck, V. Foreman, P. Grogan, M. Holland, S. Hughes, and S. Nag, "Tradespace Analysis Tool for Designing Constellations (TAT-C)," *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Fort Worth, TX, 2017.

Abstract. Mission assurance focuses on reducing uncertainty in the ability of systems to successfully complete missions despite disruptions. Current approaches to MA tend to rely exclusively on information available at design time. In other words, they do not take into account incoming situation awareness data from the operational environment. More recent approaches attempt to combine risk management with shared situation awareness data from the operational environment. However, these approaches tend to be inflexible when the system needs to operate in partially observable environments. This paper presents a closed-loop, model-based, approach to mission assurance. The approach employs: probabilistic models to account for uncertainty arising from partial observability, incremental information availability, and noisy sensors; a flexible contract construct that introduces flexible assertions into traditional contracts to adapt to changes in the operational environment while still supporting formal verification and testing to some degree; and reinforcement learning to increase confidence in probabilistic system models. The overall approach is discussed within the context of resilient multi-UAV swarm operations.

The proposed approach using Heilmeier Criteria is summarized below:

1. What are we trying to do?
 - a. Develop a formal model-based approach for closed loop assessment of mission assurance and reliability
2. How is it done today? What are the limits of current practice?
 - a. Mission assurance uses data available solely at design time (“open loop assessment”). Operational situational awareness data during mission execution is not used to update initial MA assessment.
3. What’s new in our approach? Why do we think it will be successful?
 - a. Based on resilience contract, a hybrid modeling construct, that combines formal and probabilistic models that supports model-based system verification and flexibility to cope with disruptions. Approach strikes a balance between system verification and flexibility, the two key requirements for mission assurance. Approach is based on a combination of two proven modeling approaches. We have mission assurance expert working with us from day one.
4. Who cares?
 - a. The space industry, DoD, automotive and aerospace industries
5. If we are successful, what difference will it make?
 - a. More accurate assessment of mission assurance and reliability leading to better decision making in the face of disruption.
6. What are the risks and payoffs?
 - a. Partial observability and noisy sensors pose a problem for system modeling. We overcome these problem through online reinforcement learning. Payoffs are accurate assessment of MA and reliability in safety-critical and mission-critical systems.
7. How much will it cost?
 - a. Cost to prototype will be \$260K per year for 2 years.
8. How long will it take?
 - a. Two years to demonstrate overall concept with progress demos along the way
9. What are the mid-term and final exams to check for success? How will progress be measured?
 - a. Mid-term: probabilistic system model using Markov Decision Process (full observability); final exam: probabilistic system model using Partially Observable Markov Decision

Process (POMDP). Demonstration of MDP and POMDP models on a DoD-relevant problem (e.g. autonomous ground vehicles).

4.1 INTRODUCTION

Mission Assurance (MA) is concerned with reducing uncertainty in the ability of systems to accomplish missions in the face of disruptions [1], [2], [3]. MA has historically been a part of a variety of engineering sub-disciplines including: high availability systems, failure analysis, performance engineering, quality assurance, reliability assessment, redundancy management, security engineering, hazard identification, software engineering, systems engineering, and safety engineering [4], [5].

Existing methods for MA are based on information available exclusively at design time (Jabbour and Muccio, 2010). Recently, various MA approaches have been proposed to improve on existing methods. Some of the more popular approaches for MA in federated environments combine standardized risk management with shared situation awareness among entities involved in executing operational missions [4]. However, these approaches tend to be inflexible when the system needs to operate in partially observable environments.

This paper presents a model-based MA approach that accounts for uncertainties arising from partial observability of the operational environment, and employs reinforcement learning based on incoming data (i.e., observations) from the operational environment to progressively determine system states.

This paper is organized as follows. Section 2 presents the current approach to MA. Section 3 presents the flexible contract construct for real-time MA based on operational situation awareness data. Section 4 presents a closed-loop approach to MA based on flexible contracts. Section 5 presents real-time MA concept of operations for a simple system. Section 6 discusses the technical underpinnings of the overall approach. Section 7 presents concluding comments and future developments to realize the proposed approach in operational settings.

4.2 CURRENT MA APPROACH

MA today is a sequential, design time process (Figure 1). In other words, it does not reflect real-time *situational* awareness data that could potentially contribute to more accurate MA assessment. Figure 1 presents a simplified, high level view of the current approach to mission assurance. Several key considerations such as reliability, safety, performability, quality assurance, parts, materials, processes, and domain-specific considerations such as radiation effects in space missions which are all part of MA, are not explicitly represented to prevent cluttering the figure.

The traditional MA approach, based on the analysis of dynamic system characteristics, employs methods such as: bottom up evaluation of component faults (e.g., failure modes effects analysis); worst case circuit analysis; single event effects analysis; and reliability analysis. MA predicts how likely the system will provide the required level of service in its operational environment.

Thus, the key question is how valid can predictions be that are based solely on information available at design time. While these predictions are useful for conducting trade-studies and identifying potential risk areas, it is questionable whether they reflect the realities of the operational environment. To begin with, system behavior cannot be reliably predicted because the system's environment and operational context cannot be fully controlled. Furthermore, it is difficult to discern all the states that a system might be in. As a result, it is difficult to tell whether the system is performing satisfactorily (i.e., within safe states regime), is in trouble (i.e., in unsafe states), or heading into potential trouble (i.e., about to enter unsafe states).

Thus, online assessment of MA becomes essential to understand a system’s overall state and health status, and guide decisions about its operational use.

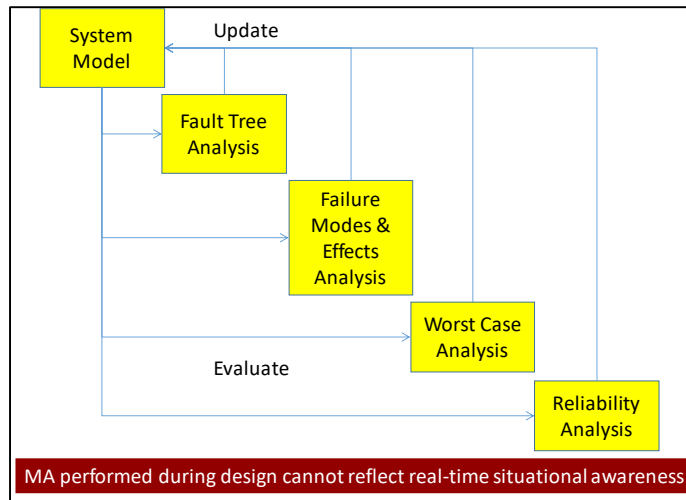


Figure 1. Current Approach to Mission Assurance

An illustrative example of MA for a simple system (Figure 2) is presented next. The simple system comprises a prime computer (P) and a redundant computer (R) that are cross-strapped to sensor and actuator interfaces.

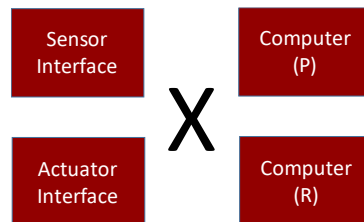


Figure 2. A Simple System

For this example, traditional MA typically involves analyzing a static block diagram (e.g., fault trees that are used to evaluate potential causes of mission failure in a top down fashion). A fault tree for the system in Figure 2 is shown in Figure 3.

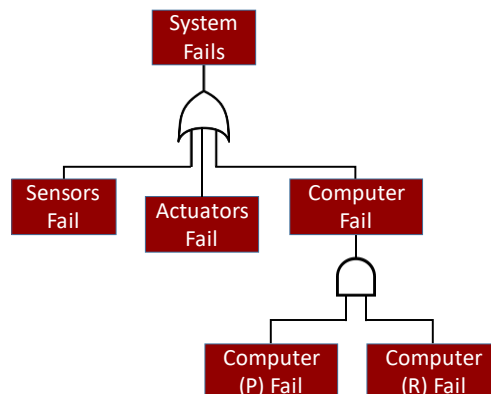


Figure 3. Fault Tree for simple system

With respect to this problem, reliability is defined by the equation:

$$R_{sys} = R_{sens} * R_{act} * (2 * R_{comp} - R_{comp}^2)$$

In this equation, R_{sys} = reliability of system
 R_{sens} = reliability of sensors
 R_{act} = reliability of actuators
 $(2 * R_{comp} - R_{comp}^2)$ = reliability associated with one of 2 computers not working

However, such traditional MA methods do not exploit sensed data from the operational environment during the conduct of the mission. The latter is needed for accurate, real-time, MA assessments. Specifically, what is needed is a flexible, closed loop approach that reduces uncertainty by progressively learning system states from sensed data. This recognition led to the creation of the Flexible Contract (FC) construct [6], [7], [8], [9], [10].

4.3 THE FLEXIBLE CONTRACT CONSTRUCT

A traditional (i.e., inflexible) design contract is a formal modeling construct defined by: assert-guarantee pairs, and rigorous concepts such as composition, abstraction, and refinement. The assert-guarantee construct states that system/component properties are guaranteed under a set of assumptions about the environment. A traditional contract enables rigorous step-wise refinement and supports modularity and hierarchy. It is capable of representing conditions on continuous and discrete components. Design contracts are used for requirements validation, conflict detection and resolution, and identification of redundant or incomplete requirements.

A flexible contract (FC) is a system modeling construct based on Markov Decision Process [6], [7], [8], [9], [10]. It extends the deterministic design contract through flexible assertions. It is suitable for probabilistic modeling and is capable of handling both observable and unobservable states. Implemented as a Partially Observable Markov Decision Process (POMDP), a FC supports in-use learning, uncertainty handling, and pattern recognition. While developed at design time, the FC is trained during actual use (“learning”).

More formally, FCs extend the concept of invariant contracts which are defined by a pair of assertions, $C = (A, G)$, in which A is an assumption (pre-condition) made on the environment and G is the guarantee (post-condition) a system makes if the assumption is met. More precisely, invariant contracts describe a system that produces an output from set $o \in \{o_0, o_1, \dots, o_{o-1}\} \subseteq O$ when in the state $\sigma \in \{s_0, s_1, \dots, s_{s-1}\} \subseteq \Sigma$ for an input $i \in \{i_0, i_1, \dots, i_{i-1}\} \subseteq I$ where O is the set of all outputs, Σ is the set of all system states, and I is the set of all inputs. Systems defined by invariant contracts are compatible with formal analyses methods that enable rigorous design and validation. However, invariant constructs are not well-matched with unknown and unexpected disruptions that might result from unpredictable swarm environments, internal faults, prolonged system usage, and previously undiscovered interactions with the operational environment. Flexibility is introduced within the “sense-plan-act” construct shown in Fig. 4. This construct comprises iterations of: sensing the environment and system status (Sense \equiv assumption); planning action that maximize the likelihood of achieving a goal (Plan), and executing those actions (Act \equiv guarantee). The environment and system health are sensed and assessed after each action. The planning function determines whether to continue with the current plan if the actions accomplish the desired outcome, or otherwise make changes.

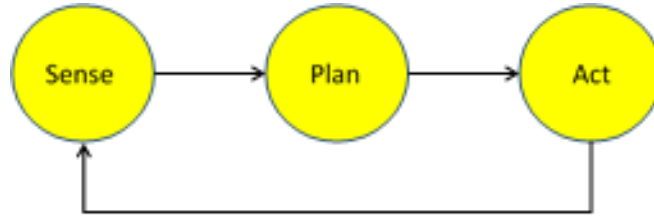


Figure 4. Flexibility is implemented with Sense, Plan, and Act Cycle

From a computational perspective, flexibility is introduced through POMDP, which accommodates observable, unobservable, and unknown states. A POMDP models a decision process in which system dynamics are assumed to be a belief Markovian Decision Process (MDP), a memoryless decision process with transition rewards. A belief MDP comprises the 4-tuple:

β = infinite set of belief states

α = finite set of actions

$(b, a) = \sum_{s \in S} (s, s' | a)$ Expected reward at b (s) on transition from s to s' given a

$\Lambda(b' | b, a) = \sum_{o \in O} \Lambda(b' | b, a, o) \Lambda(o | a, b)$ Transition function

In the transition function, a belief represents an understanding of a system state, $s \in S$, with uncertainty. The MDP has a policy, π , which describes how to select actions for a belief state based on maximizing a goal defined by the reward function, ρ , within some time period, that is, $\pi: s \in S \rightarrow a \in \alpha$. We define the expected utility of executing π when started from s as $U^\pi(s)$. An optimal policy $\pi^* = \operatorname{argmax}_\pi U^\pi(s)$ maximizes the expected *utility of an action*.

Expected utility may be computed iteratively using a number of methods including a dynamic programming approach in which a discount factor, γ , where $0 \leq \gamma < 1$, is used to penalize future rewards and is based on the “cost” for not taking immediate action. The k th value of (s) is computed iteratively using Eq. (1). The optimal policy is determined by finding the policy that maximizes $U^\pi(s)$ for each policy π_i .

$$\begin{aligned}
 U_0^\pi(s) &= 0 \\
 U_1^\pi(s) &= R(s, \pi(s)) \\
 U_k^\pi(s) &= R(s, \pi(s)) + \gamma \sum_{s'} \Lambda(s' | s, \pi(s)) U_{k-1}^\pi(s')
 \end{aligned}$$

As noted earlier, in a POMDP model, some states are not observable (hidden) because of uncertainties about the current system state and the outcomes of actions due to imperfect information (e.g., noisy sensors). The FC approach begins with a naïve model of system behavior comprising known (designed) states and transitions, as well as predicted anomalous states and transitions. For the UAV swarm, the naïve model is refined over time by observing UAV swarm behavior as actions are taken.

Post deployment, swarms perform one or more missions defined by mission scenarios. Each scenario comprises a set of mission phases that are further refined into a collection of detailed task behaviors that are allocated to the vehicles in the UAV swarm to accomplish mission objectives. Mission scenarios are defined by instantiating meta-states and transitions shown in Fig. 5. After initialization, the swarm transitions to a Planning state within mission operations. Planning takes stock of swarm health, mission objectives, risks, and prior knowledge of the operational environment to create an initial set of mission

phases and individual vehicle tasks. Thereafter, the swarm transitions to Cruise in which each vehicle positions itself for making observations. Vehicles typically make observations during Cruise and provide the information collected back to Planning. The swarm may encounter faults that require deciding how to best to accomplish the mission or select a secondary (less ambitious) mission when the primary mission becomes unachievable. The swarm enters the Observation state when it arrives at its desired locations and begins intelligence gathering and operational mission data collection.

Fig. 5 shows that the swarm may return to Cruise between observations, or may transition to degraded or unsafe operation. Fig. 5 includes restoration transitions that may return the swarm to a fully operational capability after a fault, or may place the swarm into a degraded state by marshalling surviving swarm resources.

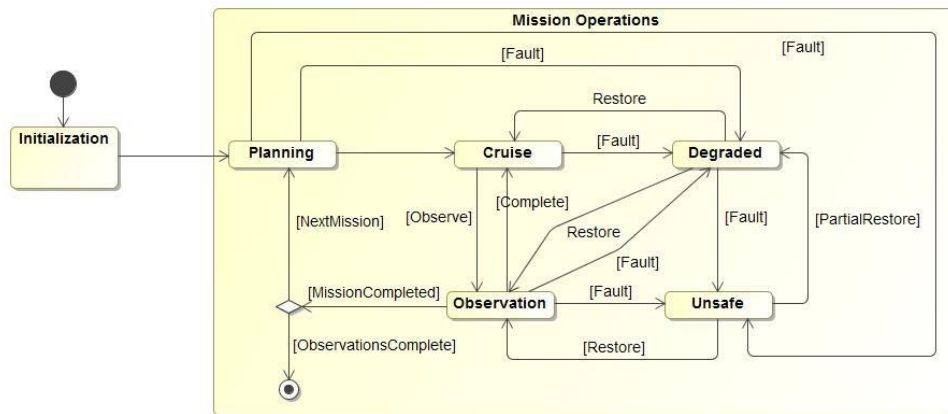


Figure 5. Mission meta-states

The swarm may return to the Planning state after concluding the mission, if there is adequate time and resources remaining to prosecute another mission. Additionally, Planning may be needed when restarting a disrupted or an aborted mission depending on when the disruption occurs during task execution, and where the swarm is in relation to ongoing observations. The Planning function may also modify missions as a result of “interesting” observations. For example, a vehicle that detects a threat may request confirmation from another vehicle that has been tasked with observing a different geographic sector. Similarly, a vehicle might request support from another vehicle if a sensor needed for a particular observation has failed, or is found to be untrustworthy.

Fig. 6 shows the structure of a FC that implements the transitions shown in Fig. 5. A belief state estimate is derived from environmental and health sensors and used to evaluate which actions to take as previously described. The actions are then mapped onto tasks.

A POMDP is a MDP, in which the system state is not directly observable, but probabilistically inferred from available sensory measurements. Thus, next action determination is not based on the current state, but on the current probability distribution. Thus, computationally there is a major distinction between MDP and POMDP. MDP assumes perfect knowledge of state, is relatively easy to specify, and is computationally tractable. POMDP, on the other hand, does not require perfect knowledge of state, is a bit more difficult to specify, and computationally intractable with respect to optimality. However, POMDP treats all sources of uncertainty uniformly, and allows for information gathering actions. Several approaches have been proposed in the literature to solve POMDP [12] [13] [14] [15]. One class of solutions is locally optimal in Gaussian belief spaces [16] [17].

In sum, with POMDP, each belief is a probability distribution. Thus, each value in a POMDP is a function of an entire probability distribution. This property is problematic, because probability distributions are continuous, and belief spaces are large and complex. So, the real value of POMDPs is in *finite* worlds with *finite* state, action, and measurement spaces, and *finite* time horizons. In these cases, the value functions can be effectively represented by piecewise linear functions.

A POMDP transitions from one state to the next by evaluating a belief state and determining the “best” action to take in the most likely system state. It is initialized by invariant assertions that represent traditional design contracts for a system. As the system operates, the emission and transmission probabilities and actions are updated (get “trained”) to reflect real world operational dynamics. Figure 6 presents a graphical depiction of a generalized FC.

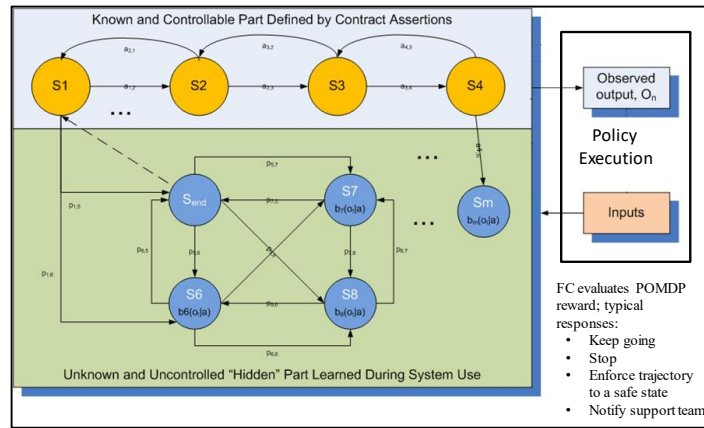


Figure 6 Generalized Flexible Contract

As shown in this figure, the upper light blue portion represents the deterministic (i.e., known) parts of the system. These are defined by invariant (i.e., traditional) contracts such as “if X, then Y.” The bottom light green portion depicts the unknown and uncontrolled “hidden” part of the system, which is represented probabilistically by a POMDP. This portion of the model, which is learned during actual system use, is akin to learning on the job. The initially postulated hidden states may or may not exist, and their probabilities are initially unknown. As observations are made during operation, these hidden states are “filled in” along with their emissions (outputs) that occur when the system is in those states. There are a few techniques to perform these calculations [18].

Intuitively, one can surmise how these techniques work. If the system is not in state S1 or S2 or S3 or S4, then it must be in one of the hidden states. The frequency with which the system goes into a hidden state determines the transition probability and the observations made in that state determine the emission probabilities.

The action policy determines the needed action at each state by evaluating a reward or penalty function. For example, the options might include: continue, stop, take an action that avoids an obstacle, put the system into a safe operational mode, notify the support team, and so forth.

4.4 CLOSED LOOP MA APPROACH

The closed-loop, FC-based, MA concept is depicted in Figure 7. A FC comprises a state estimator, a MA evaluation system (which evaluates factors such as reliability, availability, safety, and risk), and a response policy. The state estimator determines the belief state probability distribution from observations made and actions taken. MA is then evaluated from the belief distributions. The Response Policy determines the optimal action to take as a function of the state estimation and MA evaluation. Each belief estimate is associated with the system’s MA metrics (e.g., reliability, safety, risk). The probabilities are then evaluated for each belief estimation, for example:

- $P(\text{system is sound}) = \sum \text{Prob}(\text{system is in a known "good" state})$
- $P(\text{system has failed}) = \sum \text{Prob}(\text{system is in a known "failed" state})$
- $P(\text{system is in a risky state}) = \sum \text{Prob}(\text{system is in an unknown or un-designed for state})$

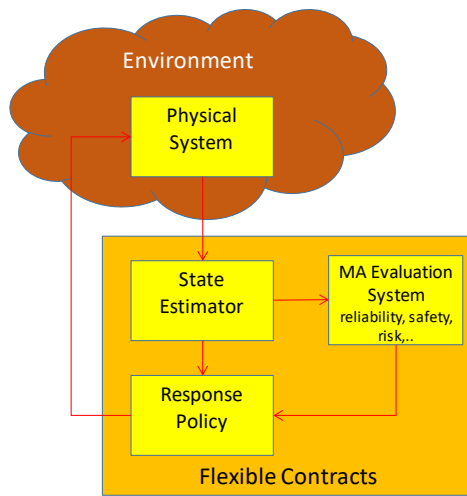


Figure 7. Flexible Contract Approach to Mission Assurance

An exemplar hypothetical progression of belief states based on observations and actions taken is visually depicted in Figure 8.

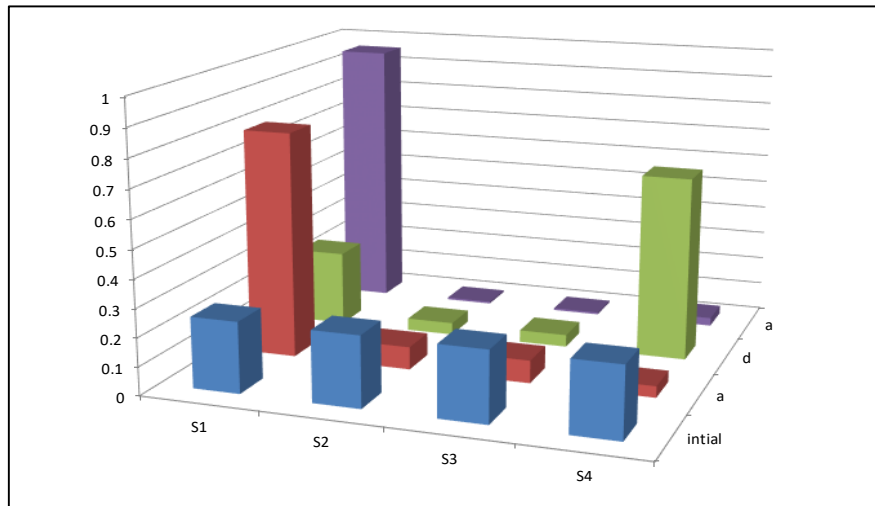


Figure 8. Graphical Depiction of Belief States Progression

As shown in this figure, initially all belief states are equally probable. When an observation “a” is made, the State Estimator updates the belief states making state S1 the most probable. When an observation “d” is subsequently made, the belief evaluation function makes state S4 the most likely. Finally, an observation “a” is made which makes S1 again the most likely. If state S4 = “failed,” for example then the reliability $R = 1 - P(S4)$.

4.5 POMDP CONCEPT OF OPERATIONS FOR EXEMPLAR PROBLEM

Returning to the simple system (Figure 2), we now examine the state model which is used for calculating reliability. The simple system comprises a prime computer (P) and a redundant computer (R) that are cross-strapped to sensor and actuator interfaces. An initial Markov model for a repairable system derived from invariant contracts is shown in Figure 9.

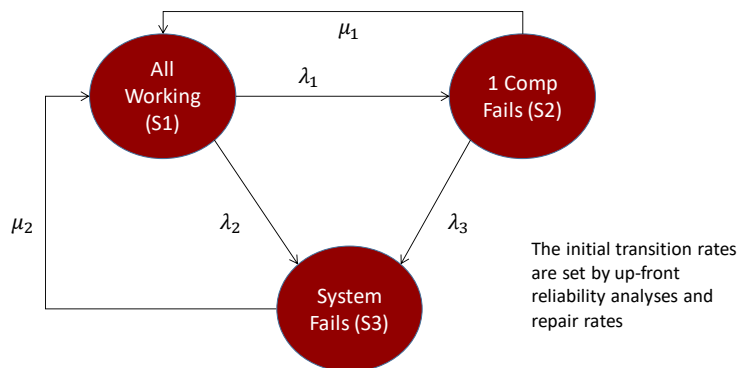


Figure 9. Markov Model for Exemplar Problem

In this model, the system is initially in state S1 (i.e., all systems working). From this state, the system can transition to other states based on failure and repair rates, denoted by λ_i and μ_j . For example, the system transitions from state S1 to S2 if one of the computers fail with the failure rate λ_1 . The system transitions from state S1 to state S3 at the rate $\lambda_2 = (1 - \lambda_1)$ if either the sensor, actuator, or both fail. The system transitions from state S2 to state S3, at the rate if the remaining computer fails or the sensor, actuator, or both fail. The *reliability* of the system is the sum of the probabilities of being in S1 or S2, or equivalently, the probability of not being state S3. *Availability* is the ratio of time when not in S3 to the total time.

We now address the problem of incorporating a hidden state into the Markov model. For simplicity, we add a single hidden state (H1) to the system model as shown in Figure 10. S1 now has a failure rate of λ_4 to H1 and H1 has a failure rate λ_6 to S2, and λ_5 to S3. Because H1 is hidden, the transitions rates into and from H1 are learned during system operation and/or test.

The “step-through” of states occurs as follows. Once the POMDP model is trained, we observe system outputs, evaluate the state we believe the system is most likely in, and take the action having the highest reward (or least penalty) for that state. For example, if the system is most likely in state S1, then it continues operating in that state in the absence of component/system failure. If the system is in state S2, it continues operating and requests a computer repair. An emergency repair is requested if system is in state S3. If the system is in the hidden state, H1, then it might continue monitoring its health for a period of time. If the system transitions to S1, S2, or S3 during the wait period, then the above actions are taken. If the system stays in H1, during wait period then maintenance personnel are called to investigate. If maintenance personnel determine that H1, is “good,” then the state diagram and actions are updated,

otherwise the action in H1 is changed to “schedule repair.” If the system state is ambiguous (i.e., two or more states are equally probable with the highest probability of occurrence), then the policy could either assume the worst, and take the associated action, or continue sampling outputs in the hope that the system state disambiguates. Table 1 summarizes the logic associated with stepping through the states.

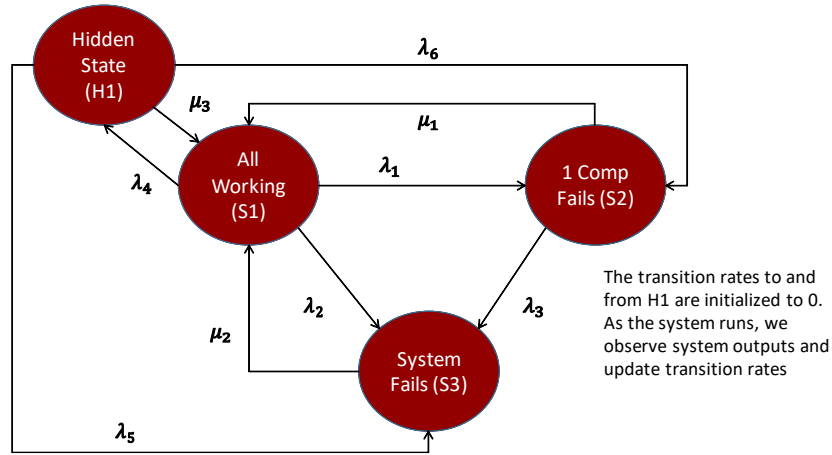


Figure 10. Introducing Flexibility in System Model

Table 7. Stepping Through States

- After training, we observe system outputs, evaluate the state we most likely believe system is in and take appropriate action, e.g.:
 - if system is in S1, then continue operating
 - if system is in S2, then continue operating and schedule a computer repair
 - if system is in S3, then schedule an emergency repair
- If system is in H1, then monitor what occurs for a period of time
 - if system transitions to S1, S2, or S3 during wait period, take above actions
 - if system stays in H1 during wait period, then schedule a maintenance check
 - if the check indicates H1 is “good,” then update state diagram and actions: otherwise, change action in H1 to schedule repair
- If system state is ambiguous (i.e., two or more states are equally probable and have highest probability), then:
 - collect additional observations and retrain state model – which might require adding new states, and schedule a maintenance check as above, or
 - assume the worst and take the associated action

In sum, incorporating flexibility in MDP consists of: introducing hidden states in MDP representation thereby making it a POMDP model; relaxing time-invariance restriction of the state space and action space; adding an evaluation metric to determine best action; and introducing the concept of time (e.g., temporal constraints/changes in probabilities that impact occurrence of events).

4.6 AN ILLUSTRATIVE EXAMPLE

We now discuss the different system modeling elements within the context of multi-UAV swarm control. These include: traditional contracts; flexible contracts, control flow model; swarm control architecture; and iterative Bayesian belief update.

Traditional (i.e., inflexible) Contracts. These are “assert-guarantee” pairs derived from system requirements and expected system behavior. Table 2 shows two invariant contracts using linear temporal logic terminology.

Table 8. Invariant Contracts

- Contract #1: At the next instant, if obstacle ahead, then turn left
- Contract #2: At the next instant, if no obstacle ahead, then continue path

Flexible Contracts. Invariant contracts can be made flexible by relaxing the assertions to account for uncertainty. The assumptions in a flexible contract are represented by belief states, while the action policies replace guarantees in the traditional contract. Accommodating disruptions is accomplished by including belief states that represent environmental hazards and fault conditions and including contingency or mitigation policies. For example, Table 3 shows a simple penalty function that is associated with a threat being possibly to the right or to the left of a UAV. If belief estimation determines that the probability of the threat on the left is 0.95 then the penalty for making a left turn is 8.5 but the penalty for making a right turn is -96, meaning there is a reward for turning right and a penalty for turning left.

Table 9. Penalty Function Calculation and Decision

- A simple penalty function:
 - $P(\text{leftThreat}) * 10 + P(\text{rightThreat}) * (-100) + (1 - (P(\text{rightThreat}) + P(\text{leftThreat}))) * (-1)$
- If $P(\text{leftThreat}) = 0.95$, $P(\text{rightThreat}) = .01$, and $P(\text{can't decide}) = 0.04$, then:
 - penalty = $.95 (10) + .01 (-100) + .04 (-1) = 9.5 - 1 - .04 = 8.49$, if turn left
 - penalty = $.01 (10) + .95 (-100) + .04 (-1) = 1 + 95 - .04 = -95.94$, if turn right
- Want to **turn right** because penalty for turning right is negative (i.e., **reward**), but penalty for turning left is positive

Control Flow Model. System behavior is captured using the control flow model shown in Figure 11. Fig. 11 shows transition “guards” or conditions that must be true for the transition to occur, e.g.:

$b(\text{failed}) \geq 0.95$ is the threshold to transition from “normal motors” to “failed motors.”

Belief state estimates are derived from observations about the system, and used by the Auto Plan function that determines next actions to take.

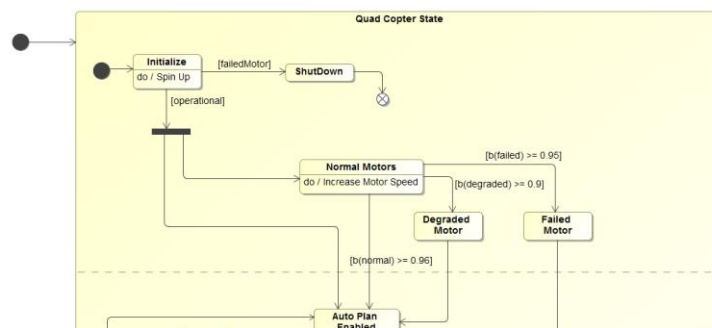


Figure 11. Control Flow Model

An exemplar architecture that implements Figure 11 using software agents is shown in Figure 12. This figure shows that environment sensor outputs, the MDP model, and the policy feed the state estimator which produces an updated set of belief values, and triggers an action for the UAV swarm.

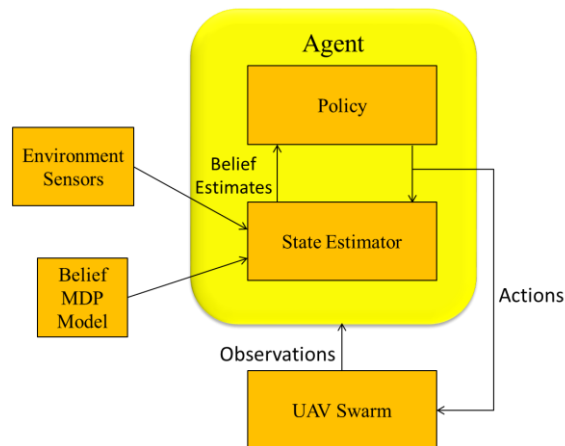


Figure 12. Multi-UAV Swarm Control Architecture

Iterative Belief Update. The system's behavior reflects its beliefs based on latest observations. For example, suppose an UAV swarm finds itself in an environment in which it needs to avoid obstacles while being mindful of unknown threats. In this instance, the UAV swarm needs to veer left or veer right to avoid an obstacle which may be either to the left or the right of the swarm. If the UAV swarm guesses correctly, it can avoid the threat. If the UAV swarm guesses wrong (i.e., it decides to veer left and the threat is located on the left, or it decides to veer right and the threat is located on the right), then the swarm is likely to suffer damage. The potential actions that the swarm can take are: veer left, veer right, or continue

on the current path making further observations of the threat. The algorithm for summing potential outcomes based on path taken is rooted in Bayesian Belief update with normalized rewards and penalties. Figure 13 presents a pictorial of iterative Bayesian Belief update and the formula for computing beliefs based on Bayesian Belief Network. The system starts with a 50-50 belief that the threat could be to the left or the right. The system makes an observation which suggests that a potential threat is to the left. So, the system’s Bayesian engine moves the belief to the left as depicted in this diagram. This move indicates that there is greater belief that the threat is to the left. Eventually, the system concludes that the belief that the threat could not be to the right because it does not observe anything to the right. Belief updating occurs in accord with Bayesian analysis using observation and current state. The state history is contained in the current state. The “a” are actions, the “o” are observations, the “b” are beliefs, and the “S” are states. Belief state changes results from observations and actions taken.

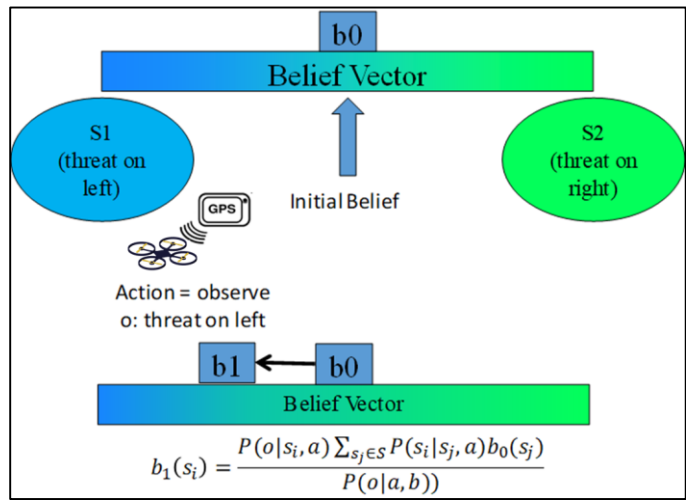


Figure 13. Iterative Belief Update

4.7 CONCLUDING COMMENTS

The MA function is concerned with reducing uncertainty in the ability of a system (or SoS) to successfully complete its mission despite running into disruptions (Madni and Jackson, 2009; Goerger et al., 2014; Madni et al., 2017b). Current approaches to MA tend to rely exclusively on information available at design time. This is a clear limitation of the traditional approach. This paper has presented a closed-loop, model-based approach based on flexible contracts and reinforcement learning. Specifically, the approach employs probabilistic modeling to account for uncertainty arising from partial observability, and incremental availability of information. It employs a flexible extension of invariant contracts from contract-based design to accomplish some level of system verification and testing while having the ability to adapt to changes. It employs reinforcement learning to reduce uncertainty in the knowledge of system state and health status. The key concepts associated with this approach were presented in this paper within the context of multi-UAV swarm operations.

4.8 REFERENCES

1. Madni, A.M., Jackson, S., "Towards a conceptual framework for resilience engineering." Systems Journal, IEEE 3.2 (2009): 181-191.

2. Neches, R., and Madni, A.M. "Towards affordably adaptable and effective systems." *Systems Engineering* 16.2 (2013): 224-234.
3. Goerger, S.R., Madni, A.M., and Eslinger, O.J., "Engineered resilient systems: a DoD perspective." *Procedia Computer Science* 28 (2014): 865-872.
4. Grimaila, M.R., Mills, R.F., Haas, M., and Kelly, D., *Mission Assurance: Issues and Challenges*, 2010 International Conference on Security and Management (SAM 10), 2010.
5. Jabbour, K. and Muccio, S. *The Science of Mission Assurance Journal of Strategic Security*, Vol. 4, No.1, 2011:61-74
6. Madni, A.M., D'Ambrosio, J., Sievers, M., Humann, J., Ordoukhanian, E., Sundaram, P. "Model-Based Approach for Engineering Resilient System-of-Systems: Applications to Autonomous Vehicle Network" 2017 CSER, March 23-25, Redondo Beach, CA, 2017a.
7. Sievers, M., and Madni, A.M., "A flexible contracts approach to system resiliency." 2014 IEEE SMC International Conference, San Diego, CA.
8. Sievers, M. and Madni, A.M. *Contract-Based Byzantine Resilience for Spacecraft Swarm*, 2017 AIAA SciTech, Grapevine, Texas, Jan 9-13, 2017.
9. Madni, A.M., Sievers, M., "A Flexible Contract-Based Design Framework for Evaluating System Resilience Approaches and Mechanisms", 2015 ISERC, Nashville, Tennessee.
10. Madni, A.M., Sievers, M., Humann, J., Ordoukhanian, E., "Model-Based Approach for Engineering Resilient System-of-Systems: Application to Multi-UAV Swarms," 2017 CSER, March 23-25 2017, Redondo Beach, CA, 2017b.
11. Madni, A.M. and Sievers, M. *Model Based Systems Engineering: Motivation, Current Status and Needed Advances*, 2017 CSER, Redondo Beach, CA, Mar 23-25, 2017
12. Monahan, George E. "State of the art—a survey of partially observable Markov decision processes: theory, models, and algorithms." *Management Science* 28.1 (1982): 1-16.
13. Sondik, Edward J. "The Optimal Control of Partially Observable Markov Decision Processes." PhD thesis, Stanford University (1971).
14. Cheng, Hsien-Te. *Algorithms for partially observable Markov decision processes*. Diss. University of British Columbia, 1988.
15. Cassandra, Anthony R., Leslie Pack Kaelbling, and Michael L. Littman. "Acting optimally in partially observable stochastic domains." *AAAI*. Vol. 94. 1994.
16. Patil, Sachin, et al. "Scaling up gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation." *Algorithmic Foundations of Robotics XI*. Springer International Publishing, 2015. 515-533.
17. Platt Jr, Robert, et al. "Belief space planning assuming maximum likelihood observations." (2010).
18. Eddy, Sean R. "Hidden markov models." *Current opinion in structural biology* 6.3 (1996): 361-365.

5 DATA SCIENCE APPROACHES TO PREVENT FAILURES IN SYSTEMS ENGINEERING - KAREN MARAIS, PURDUE UNIVERSITY

Summary: Heilmeier Criteria	
1. What are you trying to do?	Develop automated ways of tracking risk that are based on the real reasons of systems engineering failures.
2. How is it done today, and what are the limits of current practice?	Current approaches are ineffective at best. Organizations, despite following best practices, cannot consistently achieve success. Imposing more rules and reporting requirements has not increased success probability.
3. What's new in your approach and why do you think it will be successful?	Our approach is based on automatically tracking the real reasons and root causes of systems engineering failures so that we can provide practitioners actionable data.
4. Who cares?	If successful, our work will help reduce the probability of systems engineering failures, freeing engineers on all types of complex systems to innovate and create truly revolutionary systems.
5. If you're successful, what difference will it make?	
6. What are the risks and the payoffs?	The primary risk is that we do not obtain data. Therefore, we propose to use student project data in the first year, to prove our concept. In subsequent years, versions of our tool can be deployed inside organizations in such a way that the research team does not have to see the failure data. The potential payoff is a tool that can help organizations predict incipient failures.
7. How much will it cost?	We propose one year of effort to develop the prototype, using one computer science student, and one systems engineering student. This effort will cost approximately \$120K.
8. How long will it take?	
9. What are the midterm and final "exams" to check for success? How will progress be measured?	Since we will be using student project team data, the academic year provides an appropriate set of milestones. By the end of the first semester of the project, our goal is to have collected three months' worth of input signals, interim failure data (e.g., missed milestones), and final failure/success data (e.g., did the team project satisfy performance objectives?). Over the next two semesters, we will continue collecting data and also predicting failures. Our project will be successful if we can predict at least half of interim and final failures.

5.1 OUR VISION AND ROADMAP

Anecdotes and statistics on the failures of systems engineering have become a sure-fire way of attracting attention and lamentation during presentations. No-one is immune to the failure disease and in particular past success is no guarantee of future performance—organizations that have succeeded spectacularly in one project will fail just as spectacularly in the next project.

In response to these dire statistics, new methods, processes, and tools are continuously proposed and implemented, including numerous new methods of risk identification, tracking, and management. Yet the frequency of failures shows no signs of decreasing, and, meanwhile, engineering creativity in large complex systems seems to be stifled. Rather than the revolutionary creations our 20th century counterparts foresaw appearing in the 21st century, we have limited ourselves to evolutionary improvements.

Why do these methods not help as much as we hoped? One possible reason, is the reliance on extensive data creation, collection, and tracking. When projects are under pressure, activities that are seen as non-essential to the core task will not be performed, or, worse, will be performed in a cursory compliance-oriented fashion, potentially leading to misleading data and erroneous conclusions about the state of risk.

What if we could, instead, do all this risk tracking and reporting with existing information? Our proposed effort leverages two main ideas: (1) risk assessment based on the “real reasons” for systems engineering failures, and (2) combining existing data with Wisdom of the Crowd (WoC) indicators to uncover the correlations between various (unreliable) traditional and crowd-derived measures and the measurable outcome (success, failure, or delay).

Figure 1 summarizes our vision of a tool to help organizations track and manage the risks of project failures. The tool is built around state-of-the-art relational deep learning (Meng et. al., 2018), together with contextual bandit techniques (Lin et al., 2010), using a combination of enterprise data, and “Wisdom of the Crowds” data that employees enter into a mobile device app. The code is continually refined using each organization’s own data.

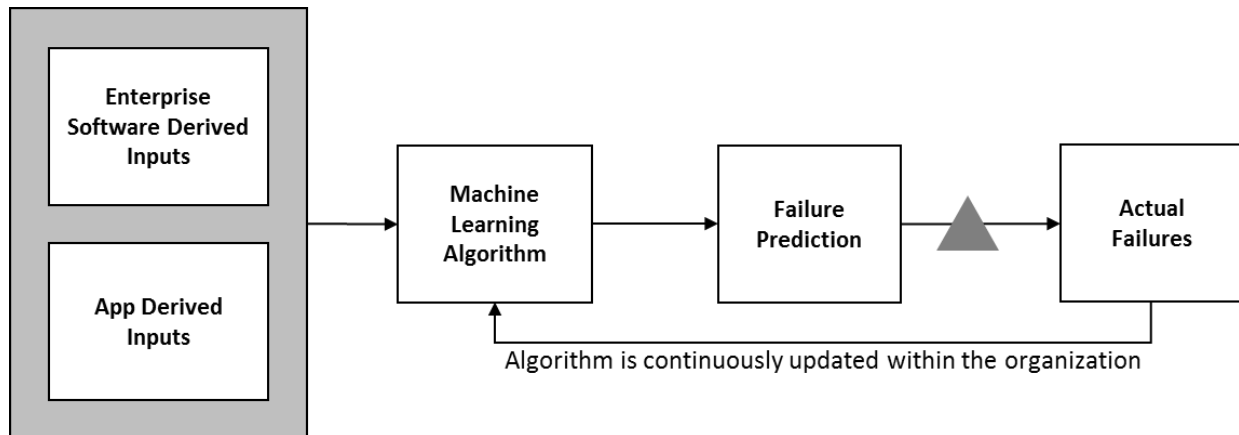


Figure 1: Envisioned Final Product

Figure 2 shows our roadmap for achieving this vision. The red text represents the activities for the first year. During the first year, we propose to identify potential input data (see Section 5.3), develop an initial version of the WoC app (see Section 5.2), and train the first generation of the machine learning software (see Section 5.4). We propose to collect training data from student projects at Purdue, ranging from design-build-test classes to student run organizations such as the Solar Car Challenge. Student projects provide a convenient microcosm of “the real world” while insulating us from the risk of being unable to obtain data.

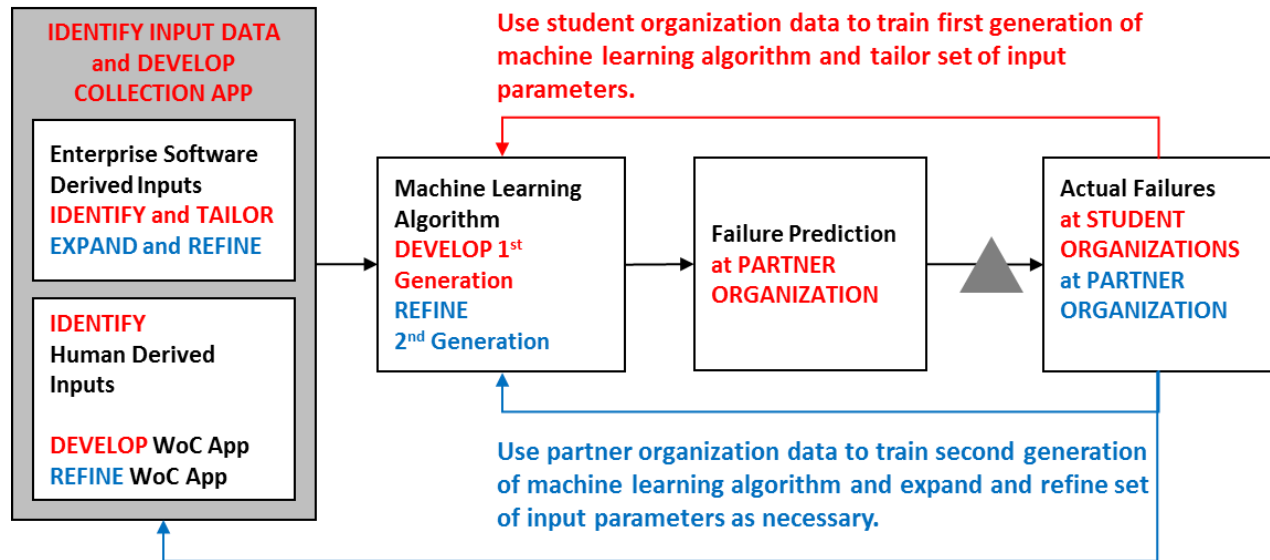


Figure 2: Overview of our proposed approach to develop the failure prediction tool

The blue text represents activities for future work, should the first year be successful. Assuming we have shown that we can successfully predict failures in student organizations, we will use this success to “market” our idea to potential industrial or other (e.g., DoD, NGO) partners. In the first part of our planned future work, we will expand the input data set, to reflect the broader range of data available in actual organizations (see Section 5.3), as compared to student organizations. Then, we will deploy our app in our partner organization(s) to collect data, based on which we will predict failures (thus providing value to our partner(s)), and subsequently refine the machine learning code and input data set.

A note about terminology: Both accidents and project failures are “undesired and unplanned (but not necessarily unexpected) event[s] that result in (at least) a specified level of loss” (Leveson, 1995). Here, we use the term “accident” to refer to those events that directly result in loss of life, injury, or damage to property. We use the term “project failures” for all other undesired project events, such as failure to achieve mission objectives, budget or schedule overruns, cancellations, and quality or performance issues.

The remainder of this chapter describes the work we have completed thus far, and next steps.

5.2 RELATIONAL DEEP LEARNING WITH FEW DATA POINTS

In complex cutting-edge projects, neither traditional data tracking nor even Big Data analysis is able to consistently and accurately pinpoint issues. Failures, even though they may appear simple on the surface (the team should have had a contingency plan!), are often the result of a complex network of decisions, many of them locally and temporally rational. Modeling and predicting such complex events requires complex models; and complex models need large amounts of historical data to be able to produce accurate predictions and insights; cutting-edge projects do not have an abundance of historical data (and even when data is available, it may be hard to collect and put into appropriate formats). Project complexity also limits the usability of methods to ameliorate these data availability issues (e.g., transfer learning).

In such complex scenarios, one can augment existing data with “wisdom of the crowd” *side information*. Wisdom of the crowd (WoC) refers to the hypothesis that the collective opinion of a large number of non-

experts (e.g., a novice engineer) is better than the opinion of a single expert (say, an experienced manager). For instance, employees give their best assessment of the timeline and budget of a project, given their knowledge of the system. These assessments are then combined with a machine learning algorithm to predict the probability that the project will be successful or the system will fail.

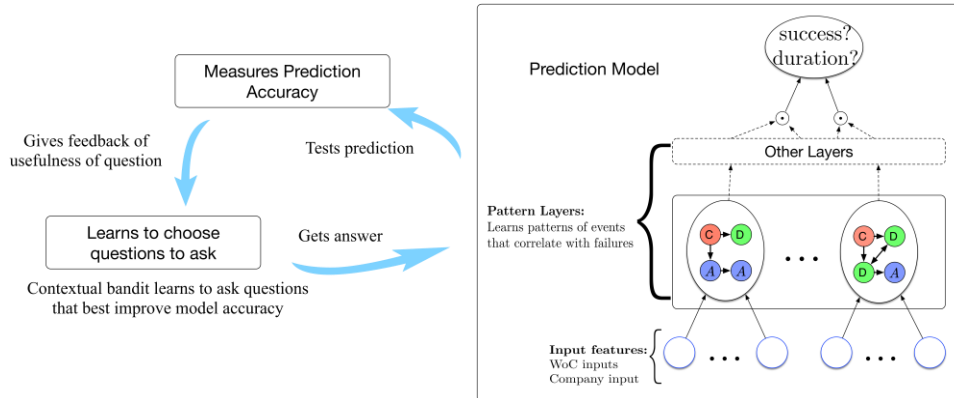


Figure 3: Neural network prediction model with contextual bandit algorithm for question selection. Our interpretable neural network (Meng et al. 2018) learns which patterns of answers related WoC questions and measurable company data predict failures. The particular structure of our model requires a orders of magnitude less data than traditional neural networks (tens of examples will be enough to train a first-generation model).

Hard-to-game WoC: Unfortunately, with bonuses and salaries depending on contracts, it is challenging to ensure employees truthfully report their project estimates (private information). Moreover, employees that have close relationships might give similar assessments and the collected data may be incomplete, further increasing the influence of these correlations. Correlations and biases must be accounted for in the final predictions. We address these limitations in two ways: (1) by asking some questions that are highly correlated with the predicted outcome but are hard to “game” (see Section 5.3), and (2) using new techniques developed by PI Ribeiro (Meng et al. 2018) to make predictions using complex relational patterns (Figure 3). The machine learning model can automatically learn which patterns in the WoC answers, combined with company data, tends to predict failure.

Initially, too many potential WoC questions: We show in Section 5.3 how we can identify a large pool of potential questions. However, no team member is willing to answer hundreds of questions every week. We need to balance the number and frequency of promising questions with our need for information. Here, we will borrow from a widely successful technique used for online testing and advertisement: **contextual bandits** (Lin et al., 2010). A/B testing is a technique for comparing two different versions of, say, a web page. The current web page design is the null hypothesis, and the proposed design is the alternative hypothesis (so it’s like between-subject design). So, the default (“A”) webpage might have the shopping cart icon in the lower right corner, while the alternative webpage (“B”) has the icon on the top right corner. The test then is to see which placement results in higher buying rates. However, when faced with thousands of potential questions (i.e., thousands of hypotheses to test), A/B testing is prohibitively expensive, as it requires too many answers (trials). Contextual bandits, on the other hand, integrate the learning algorithm (e.g., our relational deep learning method (Meng et al., 2018)) with a dynamic question-asking mechanism. This approach allows us to have a large pool of potentially useful questions, but dynamically predict the few questions that better help our model estimate outcomes (and, similarly, adapt the pool of questions for different organizations). In online advertisement, this boils down to showing online ads that will more likely result in purchases. In news organizations, this amounts to showing front-page news items that are more likely to be clicked. In our setting, this will result in deciding which questions we should ask. Our contextual bandit will use off-the-shelf Natural Language Processing

tools to learn a model that translates the question sentences into the improvement in accuracy obtained when the question is asked.

5.3 IDENTIFYING POTENTIAL SYSTEMS ENGINEERING FAILURE SIGNALS

Machine learning models are powerful but also data-hungry, so we will need a large set of signals. Generating this set of signals is the primary challenge of this work. We propose identifying these inputs using three different approaches: (1) identifying the factors underlying the real reasons for failures, (2) using systems archetypes to identify dysfunctional cases of local rationality, and (3) using cognitive biases to identify potentially irrational and destructive actions.

In related work on an NSF CAREER grant, PI Marais found that most systems engineering failures, even those in new, one-of-a-kind high-tech systems, do not involve previously unknown phenomena, or black swans [Sorenson and Marais, 2016]. As appealing as the black swan metaphor is, the *real reasons* for most failures are, in fact, rather prosaic and predictable white swans, as shown in Table 1.

Table 1: The real reasons for systems engineering failures. Adapted from Aloisio and Marais (2017)

Cause	Definition
Failed to supervise	Actor(s) in the organization failed to supervise people or a process properly.
Lost tacit knowledge when employee departed	Personnel quit, were moved to a different project, or retired, and the organization failed to sustain the knowledge base without these persons.
Failed to provide resources	Actor(s) in the organization failed to provide adequate resources to a department; for instance, maintenance, marketing, or safety.
Failed to consider design aspect	Actor(s) in the organization failed to consider an aspect in the system design. In many cases, this causal action describes a design flaw, such as a single-point failure or component compatibility.
Used inadequate justification	Actor(s) in the organization used inadequate justification for a decision.
Failed to form a contingency plan	Actor(s) in the organization failed to form a contingency plan to implement if an unplanned event occurred.
Lacked experience	Actor(s)' lack of experience or knowledge led to the failure. For example, an inexperienced manager who was placed in charge of a large project.
Kept poor records	Actor(s) in the organization kept poor records of a process, such as maintenance.
Subjected to inadequate reviews	Actor(s) in the organization did not review documentation or other work sufficiently to capture errors and deficiencies.
Inadequately communicated	Actor(s) in the organization failed to communicate with each other such that personnel were confused with the information they were given, had to "fill in the gaps" in the information they were given, or not notified about important information at all.
Conducted poor requirements engineering	Actor(s) in the organization did not lay out the needs, attributes, capabilities, characteristics, or qualities of the system well.
Failed to consider human factor	Actor(s) in the organization failed to consider a human factor in system development. This causal action describes, for example, failing to consider human factors in specifying procedures or physical design.

Cause	Definition
Failed to inspect	Actor(s) in the organization failed to inspect a crucial component.
Subjected to inadequate testing	One or more actors in the organization subjected a component or subsystem to inadequate testing. This causal action captures inadequate tests as well as adequate tests performed inadequately.
Managed risk poorly	Actor(s) in the organization failed to identify, assess, formulate, or implement a proper mitigation measure.
Violated procedures	Actor(s) in the organization violated a procedure pertaining to the system, such as a maintenance or operation procedure.
Violated regulations	Actor(s) in the organization violated a regulation pertaining to the system.
Did not learn from failure	Actor(s) in the organization did not take past failures into account and a similar problem occurred.
Created inadequate procedures	Actor(s) in the organization developed a deficient procedure, for instance maintenance, manufacturing, or emergency procedures.
Did not allow aspect to stabilize	Actor(s) in the organization did not allow a system aspect like personnel, design, or requirements to stabilize before moving forward with the project.
Conducted maintenance poorly	Actor(s) in the organization failed to perform maintenance on a component or subsystem.
Enforced inadequate regulations	A regulator (e.g., the FAA) enforced deficient regulations. This causal action captures writing deficient regulations as well as implementing regulations poorly.
Failed to train	Actor(s) in the organization failed to train other actors in the organization, such as operations personnel or maintenance personnel.

These findings mean that we already know in part what we need to look for—we just need to determine which data, or **signals**, will alert us to the presence of these real reasons. **Figure 4** shows how we propose to use the real reasons to seed our initial set of inputs, eventually discarding them as the code learns to use the input data to predict failures. We differentiate here between **factors** and **signals**. Factors are units of information that we hypothesize may lead to one or more of the real reasons. Signals are pieces of data that may alert us to the presence of the factors. Signals may already be collected by organizations, or, may be obtained using our WoC app.

Use student and partner organization data to train first and second generations of machine learning algorithm and tailor set of input signals.

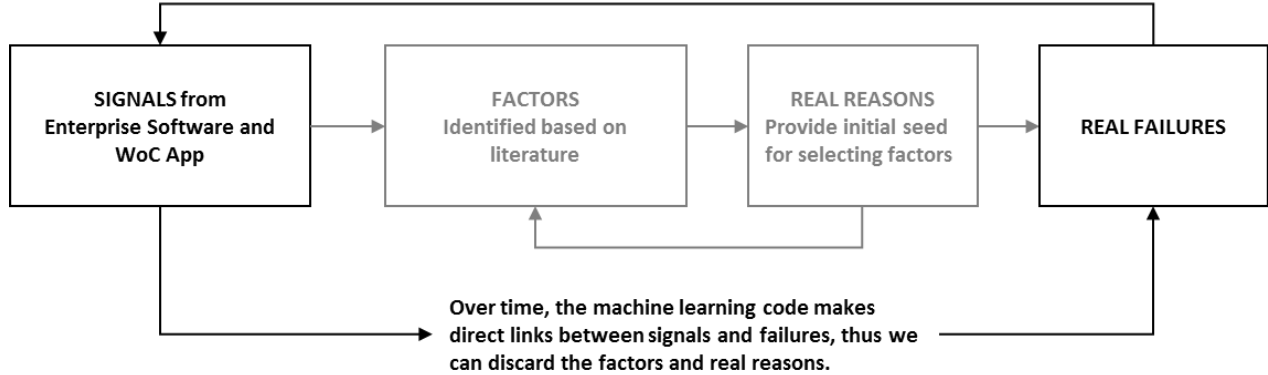


Figure 4: Our approach uses the “real reasons” to help identify potentially useful input data. Over time, the code will make direct links from the input data to the real reasons.

Figure 5 shows two examples of potential signals of poor requirements engineering, one of which we propose to obtain using the WoC app, and the second of which is already tracked by organizations.

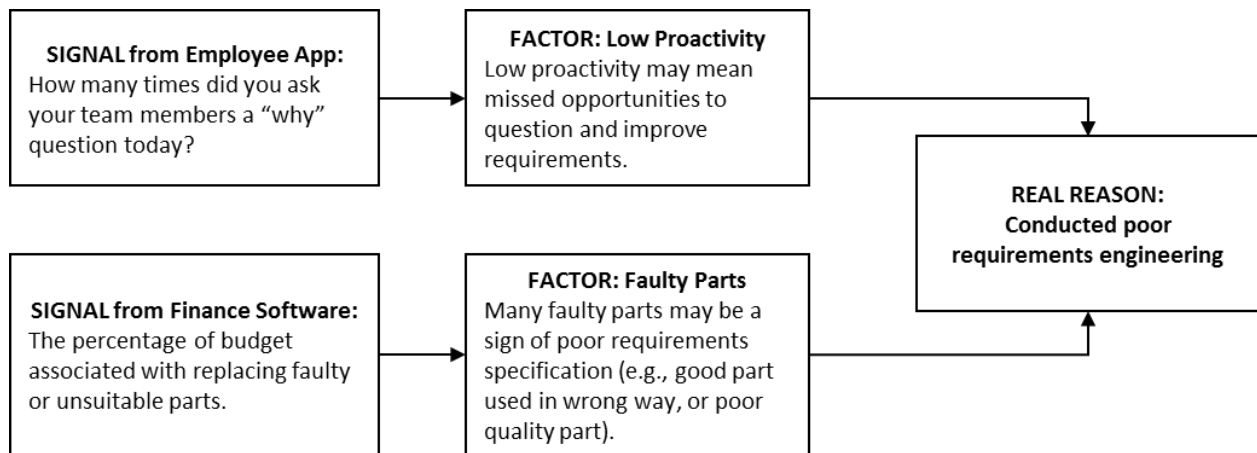


Figure 5: The machine learning code uses signals to predict the presence of real reasons. We identify potential signals by working backwards from the real reasons to factors to signals.

We propose to identify a set of potential factors and signals by surveying the organizational and human factors literature, as well as surveying enterprise software tools. For example, Table 2 shows examples of factors we identified from the literature, and how they might relate to the “created deficient requirements” real reason.

Table 2: Potential factors affecting “conducted poor requirements engineering” identified from the literature

Factor and Definition	Mechanism
Proactivity (Kirkman and Rosen, 1999): Proactive individuals show initiative, are willing to take action and affect their environment, and show perseverance.	Proactive team members may put in more effort to identify missing requirements that would otherwise be missed.

Factor and Definition	Mechanism
Neuroticism (Virga et al., 2014): Neurotic individuals are associated with low emotional stability, experience frustration, anxiety, depression, and negative emotions.	General teamwork and ability to solve problems can be negatively impacted by neurotic members.
Stress level (Dietz et al., 2017): High level of stress is associated with increased anxiety, negative emotions, distraction, conflict, and loss of team orientation.	Stress may negatively impact cognitive performance and result in incomplete tasks.
Individual experience and experience working together (Reagans et al., 2005): The level of proficiency of employees as well as the collective ability to exchange knowledge.	People with a lot of experience may be better at identifying hidden requirements.
Team Tenure (Smith et al., 1994): The duration a team has been working together.	An engineering team that has been working together for a long time may be more thorough and methodical during the requirement generation process.
Team size (Smith et al., 1994): The number of members in a team.	Larger team sizes may correlate with more thorough requirements generation.
Budget Uncertainty (Yan, 2005): Refers to deviation from the initial budget estimate, due to external funding sources and associated risk.	Uncertain budget can affect time allocated to a phase of a project and thus may result in deficient requirement generation.
Leadership style (Cummings et al., 2010): Distinction between leadership that focuses on the relationships and people, and leadership that focuses on the tasks.	Certain leadership styles might improve teamwork in the organization, which in turn may help the team create better requirements (an activity best done in teams).
Parts expenditure (Mathieu et al., 2006): The percentage of budget associated with replacing faulty or unsuitable parts.	Many faulty parts may be a sign of poor requirements specification (e.g., good part used in wrong way, or poor quality part).
# Parts failing in operation (comparing to other design teams/products)	More parts failing may be correlated to deficient requirement generation during the design phase.
# Design iterations per unit time (in comparison to previous/similar projects)	More iterations may be correlated with having to update requirements and hence update designs.

Next, we will identify potential signals for each factor. Some factors are already tracked by organizations, while others are known but not necessarily tracked in a convenient manner. **Table 3** shows examples of potential signals for several factors related to “conducted poor requirements engineering”. Some signals are obtained directly from existing data (e.g., “number of changes to estimated budget”, while others we will attempt to elucidate in a more subtle manner (e.g., “How many times did you ask your team members a “why” question today?”).

Table 3: Potential signals for factors for “conducted poor requirements engineering”

Factor	Signal
Proactivity	How many times did you ask your team members a “why” question today?
Neuroticism	Are you more excited or more worried about this project? How many times today did you feel frustrated by your team members or by other members of our organization?
Stress level	Were you able to focus on this project as much as you would like today?
Individual experience and experience working together	How many times today did you or your team have to look outside your team for advice or guidance on how to do a particular item? Did you learn anything new today in your technical discipline?
Team Tenure	How many of the people that you worked with today have you known for more than 1 year? (less than 1/3, 1/3 to 2/3, more than 2/3)
Team size	How many members are there in your team?
Budget Uncertainty	Number of changes to estimated budget to date.
Leadership style	Does your team leader know what you did for fun this weekend? Does your team leader know which tasks you are most proud of? Does your team leader know which tasks you worry about the most?
Parts expenditure	Percentage of parts budget spent to date.
Number of parts failing in operation	Number or percentage of failed parts.

However, questions such as these will not provide enough or good enough data. People may answer inaccurately for understandable reasons (for example, you may not want to tell your boss that you are feeling stressed). Can unorthodox questions, such as “did you have breakfast this morning?”, help determine whether a project is in potential trouble? Online companies, such as Google, Facebook, and Amazon, rely on unorthodox signals to predict user behavior. For instance, (De Choudhury et al. 2013) shows how the structure of someone’s Twitter follower/followee network structure can help predict depression.

The root causes of failures can often be related to perfectly rational work incentives introduced for various reasons (see, for example, Braun, 2002; Marais et al., 2006). For example, if a team is given bonuses for coming in under budget, the team, might, consciously or unconsciously, decide to buy cheaper low-quality parts or skip steps in a process. Such a response though clearly undesired, is arguably rational, given the incentives. Organizational behavior archetypes provide a potentially useful starting point for identifying these locally rational but ultimately destructive behaviors. Table 4 shows examples of safety archetypes identified by PI Marais and potential Signal Questions.

Table 4: Examples of safety archetypes and potential signal questions.

Archetype	Description	Signal Question
Eroding Goals	When performance does not reach a stated goal, managers may move the goal over time to one that appears to be more attainable, rather than determining why the organization did not reach the particular goal.	If your team redefined any goals today, did that make you feel better about meeting your overall project objective?
Complacency	When things have been going well, people tend to become complacent.	Did you worry more today than yesterday?
Stagnant Risk Management Practices	When technological advances are not accompanied by concomitant understanding of the associated risks, risk may increase.	Did you update any of your risk assessments today?
Unintended Side Effects of Fixes	Poorly thought out fixes may have unintended side effects.	Did your team experience new problems today as a result of a previous fix to a problem?
Fixing Symptoms Rather Than Root Causes	Fixes to problems that only address the symptoms may worsen or prolong the original problem.	Were you disappointed today because a problem that you thought had fixed, had instead continued or gotten worse?
The Vicious Cycle of Bureaucracy	When organizations respond to problems with more rules and bureaucracy, employees may become apathetic or alienated.	How many times today were you frustrated by rules or bureaucracy?

Rationality therefore doesn't always guarantee desired behavior, but, clearly irrationality is not the answer either! Behavioral economists have identified a vast array of cognitive biases (Tversky and Kahneman, 1975), which we will use to design simple easy-to-answer questions that aim to ferret out potentially harmful instances of irrationality. More recently, researchers have shown how these biases can affect engineering decision making (e.g., Smith and Bahill, 2010; Bohlman and Bahill, 2013). Behavioral economics can also help get better signals, by asking questions in clever ways. For example, we might ask "What do other team members think about X", where the third person perspective is used as a proxy for the subject's own expectations. Table 5 shows examples of how we will use these cognitive biases to identify potential signal questions. One of the advantages of this approach to uncovering signals is that many of the questions do not have a "correct" answer, thus making it harder for employees, perhaps well-meaning, to put a rosy tint on things, or to intentionally or subconsciously "hide" information.

Table 5: Selected cognitive biases and potential signal questions.

Bias	Description	Signal Question
Ambiguity effect (Baron, 2000)	The tendency to avoid options for which missing information makes the probability seem "unknown".	How many times today did you say or think "I don't know"?
Anchoring or focalism (Zhang et al., 2007)	The tendency to rely too heavily, or "anchor", on one trait or piece of information when making decisions (usually the first piece of information acquired on that subject).	Is your first activity in the morning related to your main project?
Automation bias (Goddard et al., 2011)	The tendency to depend excessively on automated systems which can lead to	From 1 (almost never) to 5 (all the time), how often does the key

Bias	Description	Signal Question
	erroneous automated information overriding correct decisions.	automated system needed in your project fail?
Bandwagon effect (Coleman, 2003)	The tendency to do (or believe) things because many other people do (or believe) the same.	How many arguments did your team have today?
Confirmation bias (Oswald and Grosjean, 2004)	The tendency to search for, interpret, focus on and remember information in a way that confirms one's preconceptions.	How many things did you learn today that surprised you?
Courtesy bias (Ciccarelli and White, 2012)	The tendency to give an opinion that is more socially correct than one's true opinion, so as to avoid offending anyone.	How many times today did you "bite your tongue"?
Focusing effect (Kahneman, 2006)	The tendency to place too much importance on one aspect of an event.	What is the most important decision your team made today?
Irrational escalation/sunk cost fallacy (Drummond, 1998)	The phenomenon where people justify increased investment in a decision, based on the cumulative prior investment, despite new evidence suggesting that the decision was probably wrong.	Over your career, what proportion of ideas have you thrown out? (less than 1/3, 1/3 to 2/3, more than 2/3)
Law of the instrument (Brislin, 1980)	An over-reliance on a familiar tool or methods, ignoring or under-valuing alternative approaches. "If all you have is a hammer, everything looks like a nail."	Looking at just today, the methods and tools you have on this project are (1) insufficient, (2) adequate, (3) more than good enough.
Loss aversion (Kahneman, 1991)	The disutility of giving up an object is greater than the utility associated with acquiring it.	Roughly how many ideas did your team discard today?
Normalcy bias (Kuligowski and Gwynne, 2010)	The refusal to plan for, or react to, a disaster which has never happened before.	On a scale of 1 (not at all) to 5 (all day) how much time did your team spend thinking about how things might go wrong?
Not invented here (Katz and Allen, 1982)	Aversion to contact with or use of products, research, standards, or knowledge developed outside a group.	How many ideas did you find today from outside your team?
Optimism bias (Baron, 2000)	The tendency to be over-optimistic, overestimating favorable and pleasing outcomes.	What kind of TV shows do you like to watch? [comedies/dramas/detective]
Overconfidence effect (Hilbert, 2012)	Excessive confidence in one's own answers to questions.	How confident do you feel that your answers to the questions in this app are "right"?
Status quo bias (Baron, 2000)	The tendency to like things to stay relatively the same.	When was the last time you ordered something new at your favorite restaurant?
Parkinson's Law of Triviality (Forsyth, 2009)	The tendency to give disproportionate weight to trivial issues.	How many discussions about trivial matters did you have today?

5.4 CASE STUDY: STUDENT TEAMS

In Section 1.3 we showed how we will identify a large pool of potential signal question, and in Section 1.4 we discussed how we will use machine learning techniques to use these signals to predict failures. One

crucial ingredient for the machine learning code is a set of linked signals and failures. Failure data can be difficult to obtain, because, for understandable reasons, organizations do not like to share such information. However, they may be more willing to do so if we can demonstrate that our approach has the potential to help. Therefore, we propose here to demonstrate the potential of our approach using student projects at Purdue University. Purdue offers many opportunities for students, either via large technical organizations such as the American Institute of Aeronautics and Astronautics (AIAA) or via the technical curriculum, to participate in smaller-scale engineering projects that include designing, manufacturing, testing, or operating engineering equipment. Through these phases students get exposed to real-life situations and work in teams to solve problems, while making weekly progress with meetings and sessions. Others have also used student courses or teams to obtain useful information about how engineers work. For example, Bohlman and Bahill (2013) used data from systems engineering courses to identify mental mistakes engineers make while creating tradeoff studies.

This approach offers several advantages:

1. We have ready access to such teams (pending IRB approval).
2. There are several potential teams with which we can work (reducing risk of not getting any data)
3. The teams are doing “real engineering”.
4. The students have diverse backgrounds (educational, national, and personal), reflecting what industry looks like.

Purdue has several potential teams each year, for example:

- Most years, Purdue’s AAE department runs a two-semester undergraduate course where students design an experiment that will fly on NASA’s “vomit comet”. This year, the team will develop an experiment to participate in NASA’s Micro-g Neutral Buoyancy Experiment Design (Micro-g NExT) Challenge. The students will design, build, and test a tool or device that addresses a current space exploration challenge. Test operations are conducted in the simulated microgravity environment of the NASA Johnson Space Center Neutral Buoyancy Laboratory (NBL). The project proposals are due in November 2017 with the teams testing their equipment in late spring 2018.
- Purdue’s AAE department also presents a varying range of design-build-test and design-build-fly classes every semester, including several offerings at undergraduate and graduate levels. This coming Spring, students in the aircraft senior design class will build model airplanes.
- Purdue Solar Racing is a student-run organization at Purdue University that designs, builds, and races solar-powered vehicles in national and international competitions. The multidisciplinary team consists of a diverse set of students who work in business, operations, and engineering functions. The team is currently working on their tenth vehicle, named Renatus.
- Purdue’s Chapter of the Association for Computing Machinery (ACM) offers students the opportunity to participate in Special Interest Groups (SIGs) which specialize in different areas of computer science, such as mobile applications, artificial intelligence, game development, high performance computing, multimedia, operating systems, or security. Each year, the SIGs recruit new members as returning ones give guidance and lead team projects. The students work on a project year-round before presenting it to corporate partners. One of the groups allows students to work on projects with the goal of participating in collegiate-level robotics competitions.

5.5 CONCLUSION

Systems engineering failure and more broadly project management failures continue to plague industry and government. Proper upfront systems engineering (e.g., thorough requirements engineering) has resulted in significant improvements, but, unfortunately, organizations often deviate from these practices, often for locally and temporally rational reasons (e.g., to meet scheduled milestones). Even organizations with many past successes are not guaranteed future success. Many suggested methods for managing risk are based on closely monitoring the organization or development process, but these same methods may exacerbate the problem by diluting employee resources from the main task. Here, we have proposed an alternative approach to both the data collection and failure prediction tasks. First, rather than asking people to fill out time-consuming data such as detailed spending projections, we ask them a series of relatively simple questions that do not require looking anything up (e.g., roughly how many ideas did your team discard today? Many of our questions do not have a “correct” answer; we anticipate that such questions will result in greater transparency, because they are harder to game, intentionally or unintentionally. Second, rather than trying to explicitly model the relationships between these input signals and project risk, we propose to use deep relational learning to predict failures. This code requires an initial set of training data. Because it can be difficult to obtain such data, we propose to train the first-generation model using student project data. Subsequent generations can be trained within organizations. Our approach will allow organizations to partner with us while keeping sensitive data private, and, as an added benefit, the code can be automatically tailored to best predict failures within a particular organization.

5.6 REFERENCES

- Aloisio, D.C. and Marais K. (2017). Preventing slips, overruns, and cancellations: Application of accident investigations and theory to the understanding and prevention of engineering project. Working Paper, Purdue University.
- Baron, J. (2000). *Thinking and deciding*. Cambridge University Press.
- Bohlman, J. and Bahill, A.T. (2013). Examples of mental mistakes made by systems engineers while creating tradeoff studies. *Studies in Engineering and Technology*, 1(1), pp.22-43.
- Braun, W. (2002). The system archetypes. *System*, 2002, p.27.
- Brislin, R.W. (1980). Cross-cultural research methods. *Environment and Culture*, pp. 47-82. Springer US
- Child, J. (1972). Organization structure and strategies of control: A replication of the Aston study. *Administrative Science Quarterly*, pp.163-177.
- Ciccarelli, S.K. and White, J.N. (2012). *Psychology*. 3rd ed. New Jersey: Prentice Hall, 201.
- Coleman, A.M. (2003). *Oxford dictionary of psychology*. New York: Oxford University Press.
- Cummings, G.G., MacGregor, T., Davey, M., Lee, H., Wong, C.A., Lo, E., Muise, M. and Stafford, E. (2010). Leadership styles and outcome patterns for the nursing workforce and work environment: a systematic review. *International Journal of Nursing Studies*, 47(3), pp.363-385.
- De Choudhury, M., Gamon, M., Counts, S. and Horvitz, E. (2013). Predicting Depression via Social Media. *International Conference on Weblogs and Social Media 13*, pp.1-10.
- Delmar, F. and Shane, S. (2006). Does experience matter? The effect of founding team experience on the survival and sales of newly founded ventures. *Strategic Organization*, 4(3), pp.215-247.
- Dietz, A.S., Driskell, J.E., Sierra, M.J., Weaver, S.J., Driskell, T. and Salas, E. (2017). *Teamwork under Stress*. The Wiley Blackwell Handbook of the Psychology of Team Working and Collaborative Processes, pp.297-315. John Wiley & Sons.

- Drummond, H. (1998). Is escalation always irrational?. *Organization Studies*, 19(6), pp.911-929.
- Forsyth, D.R. (2009). *Group dynamics*. Cengage Learning.
- Goddard, K., Roudsari, A. and Wyatt, J.C. (2011). Automation bias: a systematic review of frequency, effect mediators, and mitigators. *Journal of the American Medical Informatics Association*, 19(1), pp.121-127.
- Griffin, A. and Hauser, J.R. (1992). Patterns of communication among marketing, engineering and manufacturing—A comparison between two new product teams. *Management Science*, 38(3), pp.360-373.
- Hilbert, M. (2012). Toward a synthesis of cognitive biases: how noisy information processing can bias human decision making. *Psychological Bulletin*, 138(2), p.211.
- Katz, R. and Allen, T.J. (1982). Investigating the Not Invented Here (NIH) syndrome: A look at the performance, tenure, and communication patterns of 50 R & D Project Groups. *R&D Management*, 12(1), pp.7-20.
- Kahneman, D., Knetsch, J.L. and Thaler, R.H. (1991). Anomalies: The endowment effect, loss aversion, and status quo bias. *The Journal of Economic Perspectives*, 5(1), pp.193-206.
- Kahneman, D., Krueger, A.B., Schkade, D., Schwarz, N. and Stone, A.A. (2006). Would you be happier if you were richer? A focusing illusion. *Science*, 312(5782), pp.1908-1910.
- Kirkman, B.L. and Rosen, B. (1999). Beyond self-management: Antecedents and consequences of team empowerment. *Academy of Management Journal*, 42(1), pp.58-74.
- Kuligowski, E.D. and Gwynne, S.M. (2010). The need for behavioral theory in evacuation modeling. *Pedestrian and Evacuation Dynamics 2008*, pp.721-732.
- Marais, K., Saleh, J.H. and Leveson, N.G. (2006). Archetypes for organizational safety. *Safety Science*, 44(7), pp.565-582.
- Mathieu, J.E., Gilson, L.L. and Ruddy, T.M. (2006). Empowerment and team effectiveness: an empirical test of an integrated model. *Journal of Applied Psychology*, 91(1), p.97.
- Meng, J., Chandra Sekar, Ribeiro B., Neville J. (2018) Predicting Subgraph Evolution in Heterogeneous Dynamic Networks, AAAI (accepted).
- Pohl, R. ed. (2004). *Cognitive illusions: A handbook on fallacies and biases in thinking, judgement and memory*. Psychology Press.
- Reagans, R., Argote, L. and Brooks, D. (2005). Individual experience and experience working together: Predicting learning rates from knowing who knows what and knowing how to work together. *Management Science*, 51(6), pp.869-881.
- Smith, E.D. and Terry Bahill, A. (2010). Attribute substitution in systems engineering. *Systems Engineering*, 13(2), pp.130-148.
- Smith, K.G., Smith, K.A., Olian, J.D., Sims Jr, H.P., O'Bannon, D.P. and Scully, J.A. (1994). Top management team demography and process: The role of social integration and communication. *Administrative Science Quarterly*, pp.412-438.
- Sorenson, D. and Marais, K. (2016), April. Patterns of causation in accidents and other systems engineering failures. *In Systems Conference (SysCon)*, 2016 Annual IEEE, pp. 1-8.
- Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(Nov), pp.45-66.
- Tversky, A. and Kahneman, D. (1975). Judgment under uncertainty: Heuristics and biases. *In Utility, probability, and human decision making*. pp. 141-162. Springer Netherlands.
- Yang, I.T. (2005). Impact of budget uncertainty on project time-cost tradeoff. *IEEE Transactions on Engineering Management*, 52(2), pp.167-174.

Vîrgă, D., Curşeu, P.L., Maricuţoiu, L., Sava, F.A., Macsinga, I. and Măgurean, S. (2014). Personality, relationship conflict, and teamwork-related mental models. *PloS One*, 9(11), p.e110223.

Zhang, Y., Lewis, M., Pellon, M. and Coleman, P. (2007). A Preliminary Research on Modeling Cognitive Agents for Social Environments in Multi-Agent Systems. *AAAI 2007 Fall Symposium, Emergent Agents and Socialities: Social and Organizational Aspects of Intelligence*, pp. 116-123.

6.1 OBJECTIVES OF THE RESEARCH

Defense systems in operation and in development today are increasingly what we call cyberphysical in nature. Cyberphysical systems (CPS) combine sensors and actuators to perceive and act in the physical world with communication to enable information and data flow and computation to drive decision making and control the physical actuation. While CPS offer the potential for tremendous new capabilities, their ‘cyber-ized’ computation and communication backbone coupled with readily available technological advances makes them vulnerable to classes of threats previously not relevant for many defense systems. Cyberattacks are now a tremendous concern for the future of military operations, and this has spawned a drive to intentionally design “cyber resilience” into these systems at the early stages in ways that are amenable to comparative analysis and verification within the systems engineering process. Recent studies from Rand [1] and a 2016 Defense Science Board [2] are especially relevant to the design and evolution of CPS for defense. Both noted similar limitations in current processes, namely that current policies, guidance, and practice ***still assume stable and predictable operational environments and lack methods to consider the dynamics between rapidly changing threats and system configurations.***

The overarching goal of the proposed research program is to advance the theory and practice of systems security design and analysis for cyberphysical systems in ways that will specifically address the concerns noted above. We distinguish security from the broader concept of resilience in that ***security focusses on protecting defense systems from sentient adversaries.*** Cyber systems are generally designed by initially specifying critical and other necessary functionality. The high-level functionality is decomposed into specific functional capabilities, and system requirements derive from these functional needs. Boehm and Kukreja [3] distinguish between functional and non-functional requirements as what the system does and how well it does those things, respectively. The –ilities, or system qualities (SQs) of a system such as maintainability, changeability, survivability, etc. are best understood through their relation to the non-functional (i.e., performance) requirements. From this perspective, security is another non-functional quality. Security is assessed based on how well a given security design pattern protects the system as intended – without adversely impacting the critical functional capabilities.

With that understanding, this research proposes to develop a holistic approach integrating the CPS, attack vector(s), and security implementation(s) into a unified ecosystem whereby we may evaluate how well security design choices preserve critical system functionality necessary for mission success. Further, the frameworks and methods created and matured in this effort will serve as a direct compliment to existing model-based systems engineering (MBSE) processes and tools and, in turn, themselves be executable within a toolset that enables systems engineers to produce, navigate, and understand the complexity and scope of the problem. The penultimate goal is to extend our executable ecosystem model to assurance test framework and patterns. Such extension will enable the community to maintain explicit knowledge of vulnerabilities and corrective patterns in design models and begin to build standard libraries of test strategies reusable across different security design and evaluation efforts.

6.2 CURRENT PRACTICE AND ITS LIMITATIONS

As described in the SERC System-Aware Cyber Security effort [4], cyber system protection may be achieved through any number of implementable techniques that provide capabilities to respond to certain types of threats (e.g., detect when a system asset has been compromised, isolate a compromised asset,

restore an asset to an original state, etc.) that are similarly applicable to CPS. These techniques may be represented as design patterns, each contextually relevant to a given threat type, and are ideally reusable from one system to another. Security is realized by incorporating these design patterns into the system, thereby generating an adapted system architecture with new or improved, threat-specific capabilities. Together, multiple security design patterns form the security architecture.

There are two primary reasons why selecting security measures and evaluating their effectiveness are so difficult. Firstly, each of these implementations provides distinct security capabilities at a cost (e.g., implementation cost, additional resource cost, collateral impacts cost, etc.). For any given CPS, multiple security design patterns may be implemented, which may alter the original system architecture minimally or substantially depending on individual characteristics. The decision problem posed by security design is very non-linear and quite complex. Secondly, and directly related to that last point, CPS are heterarchical in nature. They are comprised of numerous, heterogeneous elements acting both independently and interdependently. Unlike traditional defense systems, they are spatial and logical in scale and complex in their behavior dynamics. Because of this complexity, traditional decomposition and predictive methods are insufficient. Further, the interdependency makes a threat to a critical system function inseparable from the original system. The system, the threat, and the security implementation change the initial system structure and functional behavior, producing an entirely new system with new dynamics.

6.2.1 EMPHASIS ON STRUCTURE OVER FUNCTION AND A CONFLATION OF -ILITIES

The extensive focus on structural system representations alone is a critical weakness in current methods. Even a recent DARPA BAA [5] emphasizes “structural design patterns” to help meet cyber requirements, calling out physical elements such as load balancers and redundant elements as examples. In all complex systems, however, form and function are intrinsically linked. A system’s structural characteristics and what processes and behaviors are possible within and as produced by that system are not separable. The structural bias results in an overt conflation of resilience (in the broader sense) with robustness or reliability. Reliability theory evaluates the impact of failures due to independently occurring natural events or system component failures and their cascading effects typically based on intrinsic system measures such as mean time to or between failure (MTTF or MTBF, respectively). Resilience metrics in related fields such as homeland security, however diverse [6], commonly treat resilience as an explicit relationship between system performance and time, thus conflating notions of reliability with resilience. This approach is also applied to systems of systems (SoS) analyses treating disruptions as a function of constituent system reliability [7].

Even many graph-based methods evaluate resilience of complex systems based on structural properties. For example, approaches like random node or edge removal are used to determine the impact on structural properties such as connectedness [8]. There has been a huge emphasis on how structural statistical properties of a network topology will be affected by additions (growth or augmentation) and removals (failures or attacks) of nodes and links, and particularly how networks can be more robust against the latter [9]. Those additions and removals are typically assumed as perturbations coming from external sources, not incorporated into the dynamics of the network itself. The literature is also replete with studies on *dynamics of networks*, meaning how a network changes its structure over time. This is distinct from studies of *dynamical processes on networks* that are most common in immunology to study disease propagation or idea dissemination in social sciences [8, 10, 11]. These latter studies will be addressed as foundational to our approach in Section 6.3.

6.2.2 LACK OF INTEGRATION BETWEEN CURRENT MBSE, THREAT ANALYSES, AND SECURITY

Engineers traditionally view threats and uncertainties as stability and/or robustness issues instead of security concerns, and current MBSE methods consequently evolved to support these perceptions. The International standard OMG Systems Modeling Language (SysML), for example, was specifically developed to support function-oriented systems development. SysML and its parent Unified Modeling Language (UML) provide for structural component descriptions and generalized functional blocks with the functional view often in the form of an activity diagram. They have a foundational basis, however, that is the key challenge most relevant to our discussion: current model-based system design paradigms are based entirely on notions of decomposition and composability. These models are also system-centric, meaning they contain no notion of external threats. While natural threats may loosely be addressed via measures of reliability, sentient threats (i.e., security) must be defined as constructs completely external to the system in question and are not captured in SysML or UML system views.

Security is concerned with coordinated attacks by an intelligent, adaptive, and rational adversary. CPS may be vulnerable if they are reachable logically, not just physically. Attack tree analysis is the most widely accepted method to assess how an attacker can gain access via the cyber aspect of these assets and subsequently exploit system functions. An attack tree represents adversary perspectives and decomposes a high-level objective (i.e., “compromise the system”) into possible paths through which an attacker could exploit a specific system feature. By themselves, however, attack trees – even with human-supported feasibility assessment and prioritization – do not merge likely threat vectors with the system in question.

Fundamentally, the objective of a cyber-attack is to gain unauthorized access to, modify, or disable a system. CPS attacks are distinguished by a primary focus on regulating (controlling) the state of physical processes. This notion is not normally exploited and has only recently been considered by most security analyses. Most security efforts for cyber-physical systems have followed an analogous paradigm to that of cyber systems, focusing on protecting the perimeter of supervisory networks controlling the embedded processors. Security measures include typical IT security firewalls, anti-virus packages, data encryption, access control, and user authentication. Yet once an attacker penetrates the supervisory network, they often find little or no security or validation schemes on the embedded process controller. This allows the attacker to modify the behavior of the embedded processor and consequently the physical process [12].

As a community, we need a modeling paradigm that holistically unifies traditional MBSE with outputs of attack tree analyses and design alterations resulting from security implementation. Complimenting these existing processes with methods that embrace the natural complexity of our resultant ecosystem may reveal emergent behaviors, economies and diseconomies of scale, and consequences we would otherwise not see. Additionally, augmenting our design and analysis process to elucidate structure-function relationships in a formal modeling paradigm will traceably capture how the functional dynamics of our system are changed by considering the entire ecosystem. In turn, this formal model will enable us to explore consequences of attacks that penetrate our perimeter protection in un-anticipated ways, moving security analyses beyond the limits of approaches described above. Finally, the model can inform and be augmented with system assurance test models that capture the context of the attack for formal test.

6.3 RESEARCH APPROACH

Our challenge is to ***define a methodology and process that is repeatable*** for each unique CPS representation that enables us to rationally compare and select security implementations (a) ***in the early stages of design***, ‘designing in security’, and (b) ***is also applicable to already designed systems*** for which a security solution is needed. The inextricable structure-function connection is widely appreciated in

research pertaining to living systems, and will be a core premise of our approach. Namely, by adding consideration of the threat(s), and protection(s), we create a new system structure with new functional dynamics. A system’s structural characteristics and what processes and behaviors are possible both within and as produced by that system are not separable. System security analysis via heterarchical models that unify these topological-functional dependencies may better inform CPS protection strategies than hierarchical WBS structures and component lists.

Our research will consequently make two central assumptions. Firstly, we can create functional abstractions of attacks and countermeasures, whereby attacks reduce or disable desired system functionality and protection methods increase or enable functionality. Secondly, modeling the original system together with these abstractions as a directed graph supportive of simulation will reveal important behavioral dynamics across the CPS, threat, and protection elements. The original system components, functional representation, and assets critical to realizing those functions will derive from existing MBSE system component models, functional architectures, and activity and/or N² diagrams. However, graph theoretic measures based on aggregate statistics alone do not adequately characterize a system in terms of its throughput, performance, and vulnerabilities. Networks may have identical degree distributions, for example, and yet fundamentally different structures functional performance [13]. Key to the ability of our directed graph to elucidate relationships and performance behaviors will be its amenability to dynamic simulation (i.e., *dynamic processes on graphs*). To achieve this, we will borrow concepts developed for ecological physics, neurology, and other living sciences.

6.3.1 PHASE 0 – PILOT BACKGROUND REVIEW AND FOUNDATIONAL CONCEPT DEVELOPMENT

The Phase 0 pilot effort began by investigating the potential for a model-based approach to CPS protection using heterarchical models to capture threats and associated system vulnerabilities. In contrast, to structural topologies where size represents spatial extent or physical dispersion CPS scale will encompass logical and flow-based (e.g., signal, power, etc.) extent across the numerous assets required to realize the necessary functions. The original notion was to cast CPS graphs as directed acyclic graphs (DAGs), a common approach for flow-based functional models. CPS differ very much from the simple-pairwise relationships traditionally modeled for many systems as their flow and logic relationships are directional. DAG representations were also used in [14] to investigate the potential for graph models to serve as a structural framework for system-aware cyber security architecture selection. CPS, however, are not acyclic.

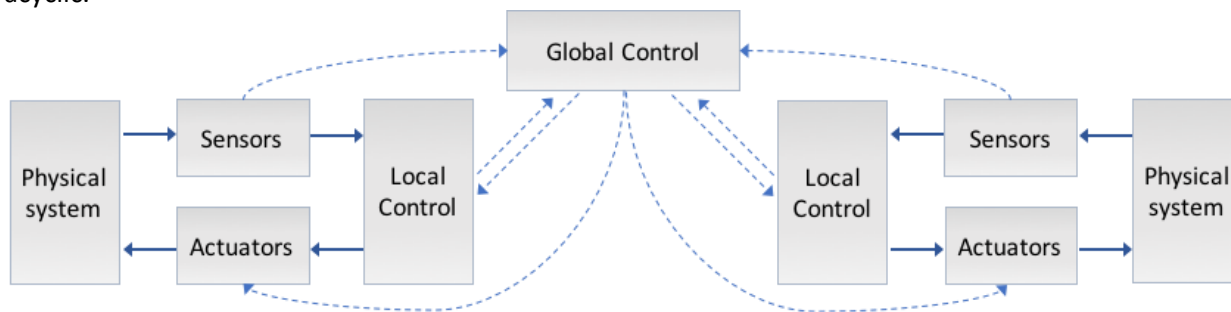


Figure 6-1 illustrates the cyclic nature of an individual CPS and group configuration. Higher-level functional capabilities are often achieved through local network structures that, while still directed, also contain cycles. While we can transform a cyclic structure into an acyclic one by duplicating node types and adding them in appropriate places, this will not produce a functional topology representative of our problem. In fact, the cycles in CPS are often important sources of vulnerability (e.g., an attacker tricks the system into staying locked into a given functional state or cycle). Moreover, the pilot research found that unlike most

studies investigating attacks in uncorrelated, scale –free graphs or assume all nodes are identical [8, 10, 11], our final graph models are not random or statistically evolved and should exhibit different dynamic characteristics. Phase 0 findings provided refined direction for the research, which will be expanded upon in the following sections. Phase 1 will contain the most detail, and the ensuing phases will be described as necessary maturations of this work.

6.3.2 PHASE 1 – DEVELOP METHODOLOGICAL FOUNDATIONS

The aim of Phase 1 is to develop the methodological foundations whereby we may begin to answer, “How do we connect the functional representations into the same ecosystem model in a way that produces meaningful consequences in the modes when executed upon – did our security design pattern implementation work or not?”. Our research must therefore address three primary challenges.

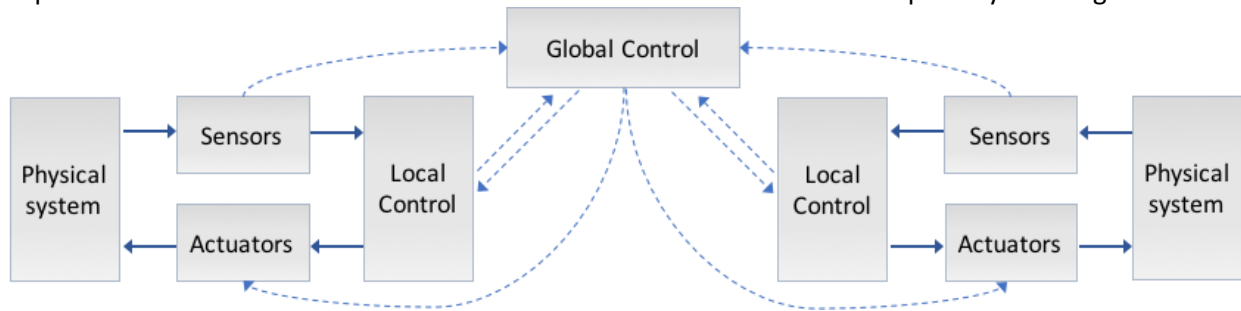


Figure 6-1: Abstract representation of a CPS group configuration

Task 1. *We need to effectively extract necessary system threat, and protection information from the respective MBSE or attack tree structures in a way that enables repeatable, automated generation of a graph structure with the correct functional specification and relationships.* The goal for this task is to automatically extract semantic descriptions from each entity type (cyber system functional component or asset, protection pattern functional capability or asset, threat functional capability) that will be included in the graphical model with respect to entity class, relationships with other entities of any class, and type of relationship (i.e., logical flow, information flow, causal dependency). With these characteristics derived from the functional model extracted from the system description, attack graphs, and a library of protective functional patterns, this will serve as the basis for a repeatable, directly executable model modification approach. The result of this task is a complete ecosystem model that is now comprised of the original (unprotected) cyber system, the threat functional capabilities and attack vectors revealing the critical cyber assets they will target, and the security functional architecture via one or more security design patterns showing how (functional capabilities) and where (cyber assets or threat assets). Our ecosystem model is contextual with respect to the threat and protection spaces.

We envision this process to begin from a CPS description including its functional capabilities and associated critical assets already specified and annotated in an MBSE paradigm such as SysML or UML. These assets are those elements required to realize a given functional capability, and may be logical elements, software elements, data elements, gates, etc. Designers will frequently add hardware components as necessary to this picture as needed to support the system assets. Systems are decomposed via standard work breakdown structures (WBS) into a structural model that defines a hierarchy of system, hardware, and software configuration items and components. As a first step, we need to semantically extract the original CPS functional information from the relevant activity and/or block diagrams into a form that may be used to create a directed graph. The nodes will represent the functional capabilities and their critical cyber assets. The edges (links between nodes) will represent a

logical flow, information flow, or causal connectivity. Both direct and indirect connections can be consequential, together producing cascading effects in the event of a disruption.

Similarly, our approach will build from an existing attack tree generation and analysis process already conducted against the critical cyber assets. We envision the effort as a compliment and consumer of attack vectors and their libraries as well as security design patterns already being investigated by the SERC System Aware Cybersecurity effort. For this research, we need to develop a semantic mapping of attack vector descriptors to targeted assets to create a modified version of our original graph. Threat functions will be new nodes and edges will connect the threats to the targeted CPS functional asset. Security design patterns are subsequently implemented alongside and on selected CPS functions to counter or mitigate the threat posed by an attack. They are also specified and annotated in the MBSE tool. We need to extract a semantic mapping of security design patterns to (a) their functional capabilities, (b) the cyber assets they require to achieve their functional capabilities, (c) the critical cyber assets and/or functions they will protect, and potentially if applicable (d) the specific threat functional capabilities and/or threat cyber assets they are designed to detect or counter through direct connective action. In graph form, (a) and (b) will be captured as nodes while (c) and (d) will be captured as edges. Notice that these edges, by definition, require the original cyber system functional model to exist.

Task 2. We must develop node and edge functional/property motifs that make our ecosystem graph executable and meaningful with respect to answering our research question. These motifs should be detailed enough to produce the needed overall system behavior but generalizable enough by type to be reusable and allow us to produce similarly executable graphs at much greater scale. Unlike the random or statistically generated graphs studies in other fields, we must develop techniques best suited and modifiable for simulating the impacts of highly correlated threats and protection implementations on functional capabilities of a system. This research task will develop ways to specify parameters in our graphical model that, when executed via discrete simulation, produce “consequences” most relevant and meaningful to the CPS security problem. A key take away from the science fields mentioned previously is that different graph topologies radically alter the dynamics possible. Our different graph topologies may analogously alter the effectiveness of establishing security for our system under threat(s). The difficulty in modeling CPS is the number of different node types or classes more so than the number of nodes. To make executable models of CPS scalable, we need to devise motifs, algorithmic representations that represent the behaviors and decision processes of each distinct class. Various graph properties may be assigned in ways that capture dynamic impact to best represent the problem, specifically node and/or edge state descriptions and weights.

- Node states and capacities. Nodes may be in one of a certain number of possible states during a simulation depending on the input they receive from their neighbors and pre-specified growth or decay characteristics of those states. A node capacity can be expressed as a direct function of its state to represent how much of something (e.g., signal strength, data quality, etc.) a given node possesses. State and capacity can modulate the level of influence a given node propagates to its dependent nodes, producing a cascading effect. Final capacities of the original CPS nodes (perhaps as a function of their state), for example, may quantify “how well” that functionality is preserved.
- Edge states and weights. Edge state is analogous to node state, except that edge state represents on or off connectivity, the degree of connectivity, and any latency, decay, or growth associated with the connectivity. For example, an edge may change its strength (weight) at a certain rate once its parent node has expressed a certain state or capacity. Edge weights represent edge capacities or proxies for measures of connection strength. Together these concepts help shape

the propagation dynamics of the cyber system and can potentially capture impact of a connective vulnerability type as well.

These techniques will, together with the graph’s final structure, govern the spreading dynamics of the impact of the threat(s) and security implementation(s) and, consequently, the final state of the cyber system’s functional capability. Rules governing node and edge states will derive from the node or edge type (i.e., node class) and relational nodes defined by a Markov blanket [13]. A key aspect of the work will be to simplify the relational state rules as much as possible to still reflect an abstraction of the contextual cyber system behavior over time. Figure 6-2 illustrates these concepts.

Task 3. We need to identify and design an approach using a real system, likely via hardware-in-the-loop emulation, whereby we may validate the research concept. While it is impractical to test the full operating envelope of our simulation backed by hardware test and evaluation, simulation like what we propose does not eliminate the need for testing using physical systems. Complex systems such as CPS ecosystems are not explicitly predictable, even when using first principles approaches. We are not trying to produce quantitatively accurate models in terms of expressing voltages, gains, inductances, etc. Rather we are striving to create functional abstractions of our system, using final node and edge states after graph simulation, that can accurately reveal whether critical system functions are preserved or not in the face of defined threats. Initial validation will be logical, conducted using designed models to verify they perform as expected. Second-level validation will need to compliment a CPS design or analysis effort using hardware-in-the-loop emulation. We propose two options that together would validate the conceptual development. Firstly, the research team has had discussions with SERC member University of Virginia about the possibility of supporting their existing RT on smart munition systems for the US Army. This effort will encompass system description, threat attack vector specification, and protective patterns. It is also envisioned to include HWIL emulation. Additionally, GTRI has in-house cyber emulation capabilities that may also serve effectively as a validation testbed.

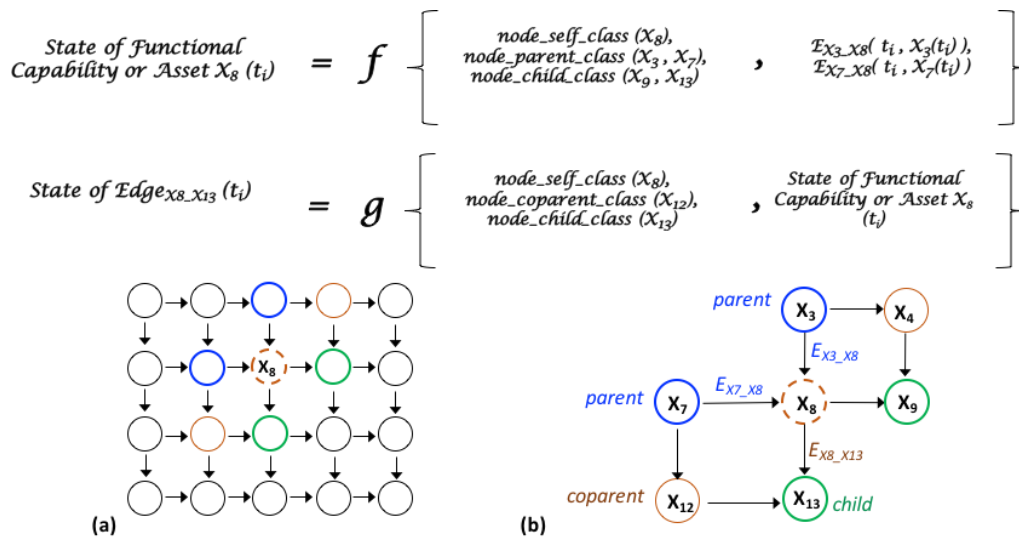


Figure 6-2: Illustration of directed graph (here acyclic) in a 2D lattice. In (a), node X8 is conditionally dependent on its parents (in blue), children (in green), and co-parents of those children (in orange). In (b), the state of X8 and an outgoing edge are expressed as functions of its inputs with functional rules based on node classes within its Markov blanket.

6.3.3 PHASE 2 – MATURE AND EXPAND METHODOLOGICAL FOUNDATIONS

Phase 2 will build from and mature the research developed in Phase 1 across three task areas.

Task 1. We need to mature the processes and structures developed in tasks 1 and 2 from Phase 1 to become semi-automated so that we may scale our approach across various CPS problems. Even with humans in the loop of the process of identifying attack vectors, prioritizing attacks, and selecting defensive protection patterns to implement, the constructs with which we represent our system and the corresponding analytical methods must be executable in a computational environment to produce the efficiencies at scale that are necessary for this problem. For this task, we will mature our methods to accomplish that vision. Additionally, the conceptual validation proposed in Phase 1 will be continued and matured under this task.

Task 2. To elucidate the structure-function relationship further, we can derive functional representations directly from structural (i.e., component) topologies that may reveal different behavioral dynamics or vulnerabilities. We need to understand and characterize how these insights might be applied to the overall problem to make a difference in our security approach. In a structural cyber topology, the nodes will represent hardware, software, or logical components. Edges will represent a logical flow, information flow, or causal connectivity. Both direct and indirect connections can be consequential, together producing cascading effects in the event of a disruption. We need to understand how the structure of our cyber system and its security implementation constrains (or promotes) the dynamics that can occur on that network. Does our structural design architecture produce the necessary functionality we need? We may produce these insights using community detection in graphs based on information theoretic approaches frequently applied to social and biological systems. In these methods, relationships (edges) represent “flow” and the focus is on discovering the dynamic interdependence of system elements connected by the edges. A known structural representation already exists as a starting point for these methods. As a baseline we follow previous work [15, 16] using random walks as a proxy for information flow to discover the essence of a graph flow patterns induced by the structural architecture. There are some key aspects of the approach we should address and will be vital to our problem: 1) What is the relationship between this functional mapping and that derived from the MBSE process? The mapping view is derived directly from the synthesis of structure and directional flow. We need to explore the differences that may arise and explain how they relate to produce a meaningful interpretation. 2) What can our mapping reveal about vulnerabilities that we did not know before? This structure-function mapping is not a dynamic simulation of our graph but a discovery of functional communities and their relationships driven (or constrained) our directional structure. This may reveal very different perceptions of fault or failure modes not discernible in the original topology. 3) Do structural design changes preserve functionality for our system? As we change security implementations or create different designs to satisfy the security requirements, we change our system topology. We may evaluate different design attributes such as redundancy, diversity, segmentation, distributed-ness, etc., and their impact to our security effectiveness. Tuning our mapping process may reveal whether we retain the same functional mapping or produce a different one. If different, we need to relate this to what we changed. This latter concept relates directly to a measure of resilience. Did our design decisions preserve functionality for our system? 4) Can we use this approach to determine “how much” of a design attribute we may need? To make design decisions, we need to know how much redundancy, diversity, distributed-ness, etc. produce a meaningful difference in our security performance. This may aid decision trades. Adding redundancy, for example, should not change our end functionality. Does it, via our community discovery? Diversity will add structural components and, consequently, new edges to our graph. Distributed-ness will add new edges. Our approach to functional community discovery may reveal what impacts on functional characteristics that the degree of these attributes included in a design may produce.

Task 3. We need to associate the inputs and outputs of our research with current processes to prioritize threats and security implementations through a decision tool where we can perform trades on the effectiveness, ease, and “cost” parameters associated with each. Each of the protective implementations provides distinct security capabilities, but at a cost. This cost is reflected in implementation cost, resource cost (i.e., the cost of any additional resources needed by the system for that implementation), and a more abstract cost reflecting the set of collateral impacts to the rest of the system. Whereas Information technology systems have few functional-structural constraints, cyberphysical systems are likely to be more constrained with a more tenable set of protections. For any given CPS, multiple security design patterns may be selected for implementation, altering the original system architecture minimally or substantially as discussed earlier. It will be imperative that we prioritize threats and security implementations through a decision tool where we can perform trades on the effectiveness, ease, and “cost” parameters associated with each. This will enable us to narrow down our threat and security implementation spaces for further analysis. Otherwise our architectural possibilities may be innumerable. We need to identify specific outputs of our research process and how they may feed into existing decision analysis methods and tools so that the “cost” of security may be evaluated.

6.3.4 PHASE 3 – EXTENSION TO ASSURANCE AND TRANSITION

Phase 3 is contingent upon the success of the previous phases. It will focus on two key aims:

Task 1. How can we formally extend and implement the process into an integrated toolset or family of related, modular tools that manage and assemble in a semi-automated way the necessary ecosystem information extraction, model construction, and node and edge property assignment (i.e., discrete functional models) to enable simulation? The goal is to produce reusable system models and specify the associated function libraries, component libraries, and standards necessary to support the semi-automated process. As part of this task, we need to evaluate and – if deemed feasible – pilot a library search capability that can partially automate the assembly of a new meta-model from existing component libraries containing key CPS components and/or associated functions, common threat vectors and associated attack libraries, and libraries of security design patterns.

Task 2: How can we extend the process to support model-based system assurance (MBSA)? The same functional model building activity developed by the previous Phases can also be used to build a test framework, essentially a separate functional “harness” that can be attached to portions of the functional model. Here the attacks are replaced by actual test functions (fuzzers) that evaluate vulnerabilities of the system. Threat patterns can be evaluated by test inputs that actually propagate through the system, and the test framework can simulate multiple attack inputs. In contrast to the system view, current methods typically test only one software component at a time. While this specific task will not sufficiently address the inherent complexity of the system, it may provide a foundation from which the community can gradually build libraries of more complex system tests.

6.3.5 RESEARCH TEAM QUALIFICATIONS TO ENSURE PROJECT SUCCESS

GTRI excels at advancing systems engineering and analysis of complex defense systems using open-source technologies to build integrated toolsets based on modular open architecture software frameworks. The GTRI team includes researchers well versed in complexity, MBSE standards, processes, and methods development, discrete simulation, and cybersecurity. To complete this effort, we envision the funding will directly support collaboration with the cyber-security efforts and analytic workbench efforts ongoing at SERC members University of Virginia and Purdue University respectively.

6.4 RELEVANCE OF THE RESEARCH

New approaches to address cybersecurity and program protection planning in development and operational use of military CPS are an obvious concern and a top priority for all DoD agencies. However, this research specifically focuses on the complex interactions between threats, well-intended protection patterns, and the resultant system behaviors that may or may not be what was intended. Our ***overarching goal is to create methods that link to existing processes and answer how well the original functional capabilities of the cyber are preserved in the face of the threat(s) given the augmentation with the security design pattern(s)***. Our dynamic graphical model will permit us to explore and understand the structure-function relationships inherent in our ecosystem that produce those outcomes and link our security choices to a trades-based decision process relating cost and level of security success. Additionally, our approach provides the structure to visibly or quantifiably reveal the inherent diseconomies of scale that can result from over protection. Succinctly, the research ***offers an approach to help answer ‘How can we evaluate the “best” security designs for a cyberphysical system?’ in a way that builds from and compliments existing MBSE and security processes***. “Best” in this sense refers to providing the necessary security to the overall CPS – a dynamic concept within an operational context – within available resources to do so. This is a cross-cutting project across three SERC Thematic Areas: Systems Engineering and Management Transformation, Trusted Systems, and Enterprise Systems-of-Systems.

6.5 RISKS, MITIGATIONS, AND PAYOFFS

Our central hypothesis is that a functional abstraction of the entire security ecosystem (i.e., the initial CPS design, attack vectors, and security design pattern implementations) can effectively produce an executable graph that will traceably reveal whether and how well a given security implementation serves its purpose as well as any unintended consequences it may produce. ***Risk***: The risk is that the high-level of abstraction inherent to functional models is not sufficient to accomplish this purpose. ***Mitigation***: This is the fundamental research question for this effort, and the hypothesis will be confirmed or refuted at the end of Phase 1. ***Risk***: The consistency and completeness of system descriptions in SysML vary by practitioner. Annotation detail specifying critical functional assets and relationships may not be captured to a sufficient level in a given use case. ***Mitigation***: The research team has extensive experience with SysML tables, query structures, and plug-ins to commercial packages that will enable us to (a) extract the necessary system information to support our approach, and (b) guide the MBSE descriptive process in ways that directly support security design and evaluation. ***Risk***: The research needs a real problem or emulation testbed to perform conceptual validation. ***Mitigation***: GTRI has in-house resources that may help address this need, and we have approached the lead SERC partner for smart munitions is agreeable to our assistance on that effort pending Army approval. If realizable, then that effort provides a real system with real data and a targeted emulation task to assist with conceptual validation

In terms of payoffs, the methods we propose are applicable to both early stage and more mature designs for which a security analysis is needed. They are similarly extensible to use cases where CPS systems are networked together to provide a greater physical capability extent. Our approach has the potential to answer critical questions that are not adequately addressed through the currently disparate processes.

6.6 BUDGET

Although precise figures have yet to be calculated, the estimated budget is \$345K for Phase 1 and a total of \$345K for a comprehensive Phase 2 program consisting of targeted efforts for maturation across the multilevel framework that are driven by the results from Phase 1. An optional Phase 3, to be undertaken based on the success of Phases 1 and 2, consists of extending the processes and tools into existing

processes and application for wider use. The projected budget for Phase 3 is \$290k. Thus, the total budget for the overall research program, including Phase 3 and the \$20K for Phase 0, is approximately \$1 million.

6.7 TIMELINE

Phase 0 of this effort was completed during the incubation project period and lasted 2 months from funding was received until this report was produced. Phase 1, Phase 2, and Phase 3 are each projected to last 12 months. The total timeline outside of the pilot is projected at 3 years.

6.8 PROJECT EVALUATION AND MEASUREMENT

By the end of Phase 1, we expect to produce a minimal viable demonstration of our approach and a comprehensive description of our underlying conceptual advances for the SERC and DoD sponsors. By the end of Phase 2 – within 2 years - we aim to produce and validate strong theoretical foundations, and prototype processes and tools capable of being demonstrated on realistic problems and integrated into existing complimentary MBSE and attack modeling processes. Within 3 years, at the end of Phase 3, we expect to see significant validation of both our theory and tools in pilot applications at realistic scales and associated complexity as well as extend our approach to support CPS assurance.

Progress will be measured by annual technical review assessments made by key stakeholders across the SERC, DASD(SE), and complimentary RT stakeholders such as AARDEC. These reviews will include a description of the conceptual advances and methods, processes, and tools that can support their application to real problems as well as relevant demonstrations. The findings and results will be vetted with the stakeholder community for any necessary course-alteration and transition planning.

6.9 REFERENCES

- Snyder, D., Powers, J. D., Bodine-Baron, E., Fox, B., Kendrick, L., & Powell, M. H. (2015). *Improving the cybersecurity of US Air Force military systems throughout their life cycles*. RAND Corporation, Air Force Project, Santa Monica, CA.
- Defense Science Board (September 2016). *Task force report on Cyber Defense Management*. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, DTIC number AD1023639.
- Boehm, B., and Kukreja, N. (2015). *An Initial Ontology for System Qualities*. In INCOSE International Symposium, vol. 25, no. 1, pp. 341-356.
- Jones, R. A., & Horowitz, B. (2012). A System-Aware Cyber Security architecture. *Systems Engineering*, 15(2), 225-240.
- Defense Advanced Projects Agency (DARPA) (May 2017). *Cyber Assured Systems Engineering (CASE)*, Broad Agency Announcement, Solicitation number HR001117S003.
- Dessavre, D. G., Ramirez-Marquez, J. E., & Barker, K. (2016). Multidimensional approach to complex system resilience analysis. *Reliability Engineering & System Safety*, 149, 34-43.
- Uday, P., & Marais, K. (2015). Designing Resilient Systems-of-Systems: A Survey of Metrics, Methods, and Challenges. *Systems Engineering*, 18(5), 491-510.
- Newman, M., Barabasi, A. L., & Watts, D. J. (2011). *The structure and dynamics of networks*. Princeton University Press.
- Geard, N. (2010). *Adaptive Networks: Theory, Models and Applications*. T. Gross and H. Sayama (Eds.), Springer-Verlag.
- Barrat, A., Barthelemy, M., & Vespignani, A. (2008). *Dynamical processes on complex networks*. Cambridge University Press.

- Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., & Hwang, D. U. (2006). Complex networks: Structure and dynamics. *Physics reports*, 424(4), 175-308.
- Lyn, K. G., Lerner, L. W., McCarty, C. J., & Patterson, C. D. (2015, October). The Trustworthy Autonomic Interface Guardian Architecture for Cyber-Physical Systems. In *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015 IEEE International Conference on* (pp. 1803-1810).
- Li, L. (2007). *Topologies of complex networks: functions and structures*. California Institute of Technology.
- Lockett, B. (2013). Integration of Graphical Modeling Techniques as a Structural Framework for System-Aware Cyber Security Architecture Selection. Thesis from <http://libra.virginia.edu/catalog/libra-oa:3720>.
- Rosvall, M., Axelsson, D., & Bergstrom, C. T. (2009). The map equation. *The European Physical Journal-Special Topics*, 178(1), 13-23.
- Ziv, E., Middendorf, M., & Wiggins, C. H. (2005). Information-theoretic approach to network modularity. *Physical Review E*, 71(4), 046117.

*Model-Based Assurance Arguments for Verification and Validation of Autonomous Behavior
Requirements for Autonomy-Enabled Systems*

7.1 PROBLEM AND NEED

Autonomy-enabled systems (AES) are a DoD priority within the technology offset strategy.

AES present new challenges for Systems Engineering. In AES, some traditionally human cognitive operations - choices, interpretation, adaptation, interaction etc. – are transferred from a human operator to a machine. Human operators are trained and selected, after an initial 18 to 22 years of life experience, operators are able to recognize unusual situations, drive in traffic, and adjudicate conflicting guidelines such as “drive slower in the rain” and “drive faster under fire.” They are expected to adapt and apply the unit’s tactics, techniques and procedures (TTP) to execute the mission plan and accomplish the mission objectives – as appropriate for the situation as it develops. In AES, requirements substitute for training and selection.

Trusted autonomy and a trustworthy AES requires that the system is able to integrate and operate as an effective as part of the autonomy-enabled unit. Trustworthiness is more than simply being safe and effective. The team leader must employ the system effectively as part of a manned-autonomous team, and the manned and autonomous elements must interact smoothly to execute the TTP to accomplish the mission. Military operations are inherently unsafe. A faster, more effective operation is a safer operation.

- The team leaders, operators, and adjacent team members understand the capabilities and limitations of the AES, how it will act and perform over the range of missions and situations
- The AES does not limit the team’s ability to execute operations employing the unit’s TTP, or limit the team leader’s ability to assign tasks assuming that the appropriate TTP will be used and used effectively
- The AES is competent to perform its autonomous functions effectively, and improves the team’s operational effectiveness over the range of missions, operation plans, and particular situation

AES are machines, sophisticated machines, but machines nonetheless. Their capabilities and behaviors must be expressed as system requirements and acceptance criteria in system acquisition. This includes the autonomous “cognitive” functions that will enable the AES to operate as part of an autonomy-enabled unit with humans and AES working together.

The SE community currently lacks MPT to *develop and verify system requirements* for the “cognitive” autonomy capabilities, and lacks MPT to assess designs to *inform technology-selection and source-selection decision makers* vis the acceptance criteria. Despite this, DoD, and TARDEC in particular, recognize that AES will enter the acquisition process. Acquisition agencies will be required to develop requirements and acceptance criteria, and to assess designs to inform technology-selection and source-selection decisions.

Historically, in software intensive systems, errors in requirements – incorrectness, inconsistency, incompleteness – are the greatest source of development time and cost overrun (see figure 7.1). This relationship is likely to be even more skewed for AES. Although not well-documented, unsuitable requirements – not the ability of engineered solutions to satisfy the requirements – has been a major cause of program terminations (the EFV, the DIVADS gun, the Shillelagh missile, the Sheridan tank, and others are examples worthy of study as to what went wrong).

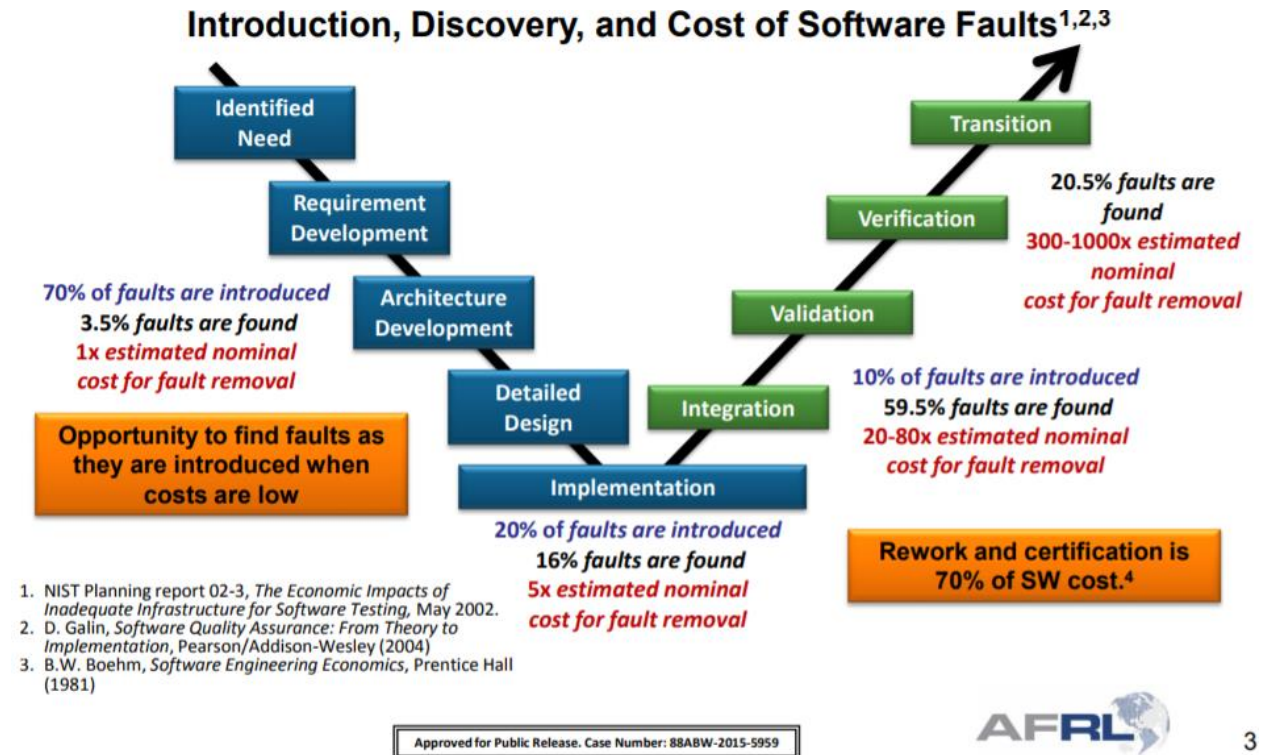


Fig. 7.1: Software Fault Introduction and Rework Costs over the Development Lifecycle. From “Test and Evaluation of Autonomous Systems in a Model Based Engineering Context,” by M. Nolan (Raytheon), A. Fifarek and J. Hoffman (USAF AFRL), March 2016

Effective SE MPT for AES autonomous capability requirements TEVV should have the following characteristics:

- A “user-friendly” framework to express behavior requirements (the term “behavior” is used broadly to refer to any action taken by the AES to include goal selection and adaptation as well as physical and communication actions)
- Behavior requirements can include both prescriptions and prohibitions
 - Prescriptions specify what the AES should do under what conditions – including internal goal setting actions, so the prescriptions can represent desired outcomes
 - Prohibitions specify what actions should not be taken under what conditions, and what outcomes are to be avoided
- The behavior requirements can be refined and expanded over the acquisition lifecycle
- The requirements for situation awareness and underlying perception capabilities are derived from the behaviors
- The requirements for execution and underlying actuation and communication are derived from the behaviors

- The behaviors representation is such that the behavior set can be analyzed to identify conflicts, inconsistencies, ambiguities, and gaps
- The behaviors representation is suitable to be incorporated into dynamic models of the military operations, including the behaviors of other friendly and unfriendly agents, in order to assess operational outcomes
- The individual behavior requirements, the assessment of the set of behaviors, and the assessment of the operational outcomes all feed a rigorous method to produce an aggregate evaluation of the AES trustworthiness (effectiveness, robustness, and compatibility with unit TTP), and support diagnosis of the evaluation

7.2 TECHNICAL SOLUTION

The solution framework for autonomous behavior requirements is illustrated in figure 7.2.

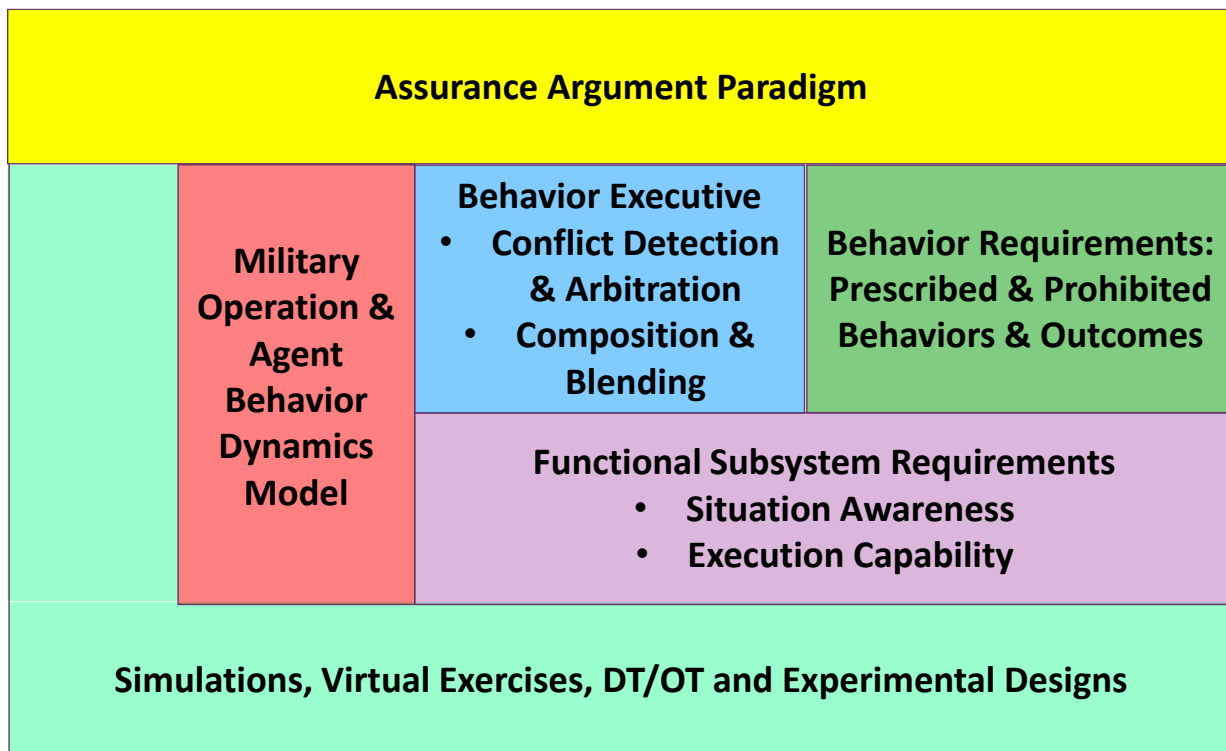


Fig. 7.2: Model-Based Autonomous Behavior Requirements TEVV Framework

Assurance Arguments

The assurance argument paradigm is the integrating framework that “rolls up” the hierarchy of claims and sub-claims with their supporting model-based evidence and assumptions. At the top level, claims are about the effectiveness and team compatibility over some range of operations and situations.

Assurance arguments are an increasingly-used approach to assess safety, effectiveness, security, reliability and similar properties that cannot be proven by formal means and for which statistical data and/or test data are unavailable or insufficient for the range of conditions. They are especially suitable

early in the development process when decisions must be made and justified prior to engineering development.

Properly formulated, assurance arguments identify key assumptions, as well as information that is needed early in the acquisition lifecycle to make informed capability choices, requirements, and source selection decisions. Extended assurance arguments can potentially provide way to integrate diverse types of evidence accrued over the development lifecycle.

Various forms of the assurance argument approach are being developed and applied: by the FAA for safety critical systems [1], by NASA for “one-shot” satellite and launch systems [2] and for systems using Artificial Nets [3], by medical device manufacturers and the FDA [4], cybersecurity development [5], and hazard and safety analysis [6]. DARPA has released a solicitation for “Assured Autonomy” to develop assurance arguments for adaptive learning in real-time [7].

The assurance argument approach has been criticized for lack of a rigorous formalism for combining evidence of different types, sub-claims, and assumptions in the argument. There have been some attempts to bring greater rigor by linking evidence to models [8], and by formalizing the types of assurance arguments to use for different types of claims [9].

The proposed approach brings enhanced rigor to the assurance argument model by explicitly tracking the *scope* and *credibility* of sub-claims, evidence and assumptions (figure 7.3). The scope is the range of conditions of the claim, and credibility is the confidence that the claim is true over the range of conditions. The scope and credibility of a claim are composed from the scope and credibility of its support, according to the calculus appropriate to the logical nature of the decomposition.

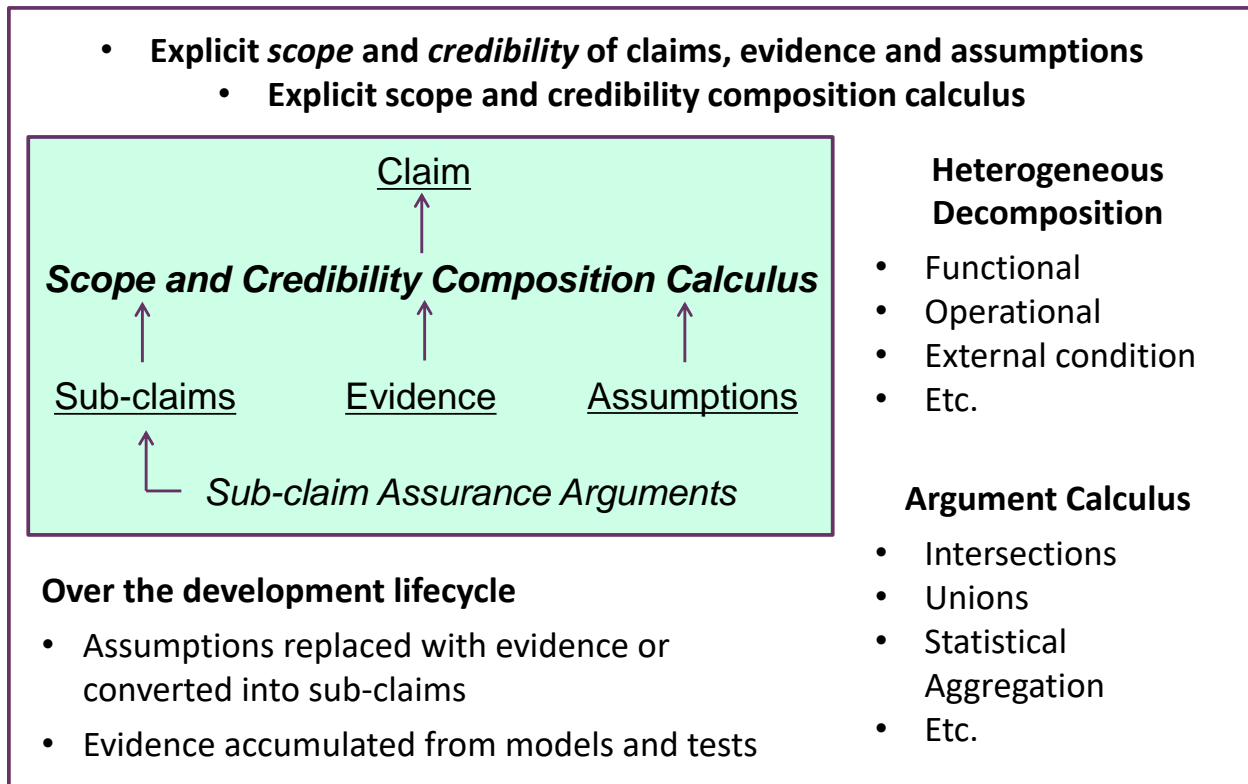


Fig. 7.3: Extended Assurance Argument Paradigm

Evidence comes from multiple sources: (1) detailed simulations, virtual exercises, subsystem testing, etc., (2) an operational dynamics model, (3) a model of the behavior deconfliction, arbitration, and composition, and (4) the set of behavior requirements. The operational dynamics model framework, the behavior executive model, and the behavior requirements format are within the scope of the proposed SE MPT. The detailed simulations etc. are not. The experimental designs used to generate evidence from virtual and constructive simulations and tests are key factors in characterizing the scope and credibility of the experimental results.

Generic AES Functional Architecture

The technical solution is based on an implementation-agnostic functional architecture for the autonomy component of an AES (figure 7.4). The architecture shows the central role of the behavior executive responsible for arbitrating among conflicting behaviors, composing compatible behaviors, and detecting situations for which the AES has no appropriate behavior. The behavior executive is a central part of an operating AES. *The behavior executive is also the key tool to analyze inconsistencies, gaps, ambiguities and incorrect behavior requirements.*

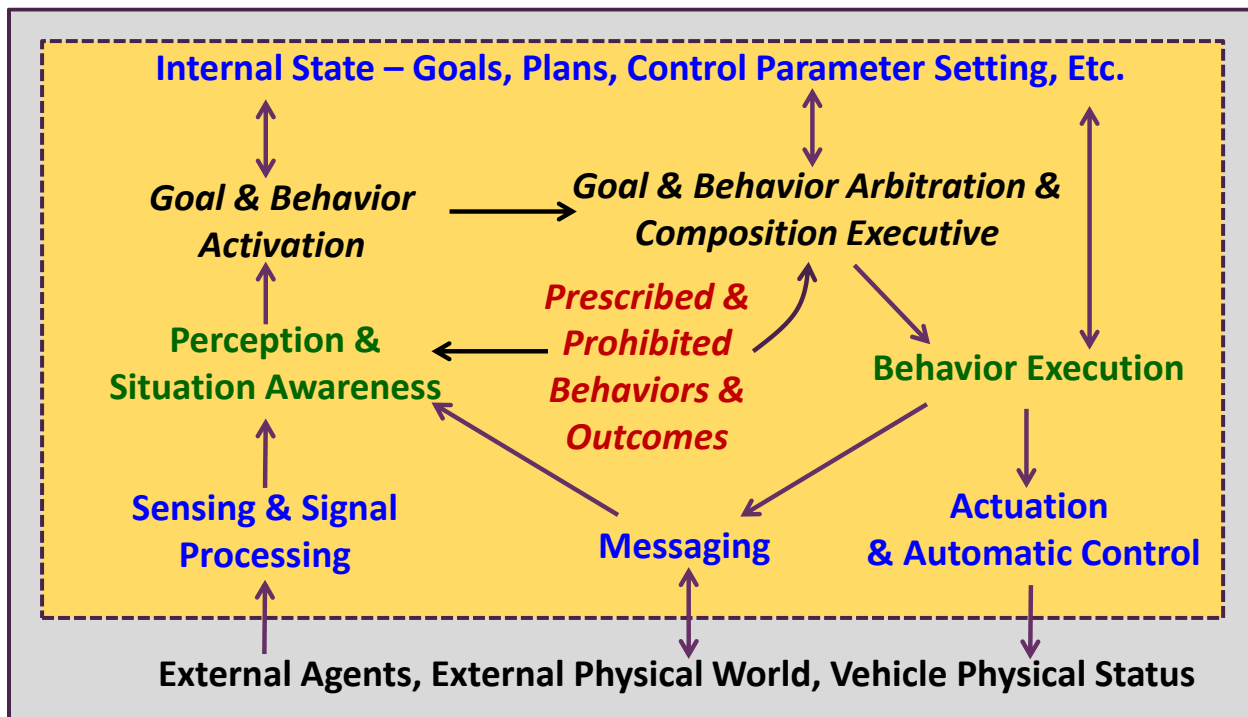


Fig. 7.4: Implementation-Agnostic AES Functional Architecture

The behavior requirements specify what the AES should do when (prescriptions for actions and outcomes) and what it should not do when (prohibitions on actions and outcomes). Behaviors have activation conditions which define precepts of situation awareness, and consequence actions or activities, which are controlled based on feedback from situation awareness precepts. The behavior specification drives the perception and situation awareness needs. The behaviors define the activities and control limits for the AES. A behavior specification has three parts

- A conditions template specifying the situation in which the behavior should be applied

- A distance metric specifying the activation level as a function of the distance between the template condition and the perceived situation
- A consequence action expressed in terms of the conditions, so that the actions will be automatically tailored to the perceived situation

This two-level structure with an outer constraint loop (prohibitions) and an inner prescriptive loop is an emerging best practice for safety assurance in complex and autonomous systems [10]. The behavior that the AES will exhibit, and its operational effectiveness, depends on how behaviors are composed and conflicts are resolved, and cannot be predicted from behaviors requirements individually.

The behavior requirements will almost certainly be incomplete and inconsistent. It is not reasonable to expect that all possible combinations of situations will be explicitly accounted for in the behavior requirements. The conditions for the prescription and prohibition behaviors will specify when they should and should not be applied, but the conditions as specified will not have all the details. Different behaviors can be activated at the same time by different aspects of the situation (e.g., it can be raining and sunny at the same time). The *activation level* of a behavior is a measure of how close the current situation matches the template for the behavior. The behavior executive uses the activations levels of the behaviors and their relative importance to arbitrate and compose the action that the AES will take.

The behavior executive is used for *behavior checking* – exploring the space of conditions to detect conflicts, ambiguities, gaps and incorrect behaviors. The SE MPT will have a behavior checking tool that explores the space of conditions, examining (1) combinations of conditions drawn from different behaviors, and (2) situations that have intermediate values between different behaviors.

- Conflict is when two behaviors are activated that produce conflicting consequences (either in immediate action, or outcome consequence)
- Ambiguity is when multiple behaviors have nearly equal activation levels and importance
- A gap is when no behaviors have significant activation
- Incorrect behaviors are when the outcome conflicts with prescribed or prohibited outcomes

The behavior executive is the key tool to find conflicts, gaps, and ambiguities among the behavior requirements. It is used inside a behavior checking loop. An operational model is used inside the behavior executive to forecast outcomes.

Operational Dynamics Model

The operational dynamics model is where the actions of the AES and other agents are played out to assess conflicts between outcomes (prescribed or prohibited) and actions/activities (prescribed or prohibited).

Behavior requirements include prescribed and prohibited combinations of actions/activities subject to the tactical situation. This does not need an operational dynamics model. When behavior requirements are expressed in terms of outcomes to seek or avoid, i.e. goals not actions/activities, then an embedded operational dynamics and outcome model is needed.

The operational dynamics model is needed to identify conflicts in of action behaviors with outcome goal & constraint behaviors. (In the implementation-agnostic framework, behaviors are situationally dependent, prescribing and proscribing actions, combinations of actions, and outcomes of executive adjudication of actions vis outcomes).

Detecting when a set of behaviors leads to a prohibited outcome requires an embedded model of the operational dynamics and outcomes. The operational model is used inside the behavior executive to detect conflicts between outcomes, prohibitions and goals.

The behavior executive is the central tool used by the behavior-checking function to identify inconsistency, gaps, ambiguity, etc. among the requirements. At the simplest level, it does not project outcomes, and excludes behavior goals and constraint activation in reference to outcomes.

The operational dynamics model is needed when some behavior requirements are expressed in terms of prescribed and prohibited outcomes. For the purposes of behavior requirements TEVV, the operational dynamics model needs to reflect the behavior requirements with expected “mechanical” or “deterministic” physical response. It is not a detailed simulation of performance, but rather an “expected value” model of “what happens when A does X, and how does it fit into the operation TTP.”

Even in “simple” convoy operations there are complex situation awareness and decision behaviors related to lane change, maintaining formation, etc., over the variations of road and weather conditions, mission priorities, neutral and adversarial actors, breakdowns and mutual support, with heterogenous units and formations (manned, weaponized, etc.), the challenge of actions to expect when (behavior requirements) is hard. Behavior requirements will be in conflict. How requirements are expressed and conflicts resolved is essential to asking for a system that will be operationally effective.

Finite-state machines (FSM, sometimes also called state-transition models) can be an effective approach for operational dynamics modeling. Agents – friendly, unfriendly, and automated – are engaged in activities or actions. The actions take time to complete. Multiple actions occur in parallel, and the probability that one action or another completes next depends on the relative rates. FSM are well-suited for simple operational concepts, threat interactions, TTP and behaviors.

A criticism of FSM is that can become unmanageable when there is an attempt to have broad scope and high resolution simultaneously. Complexity can be managed with extended hierarchical FSM. A state at one level of the hierarchy is decomposed into a more detailed FSM network, whose inside is invisible at the higher level. Another approach to managing complexity for *multi-agent operations* is to use hierarchical FSM to model agent behaviors, and let a computational engine rather than a specific model, generate the states of the world from the actions and activities of the individual agents.

The use of a first order model, or any model for that matter, relies on assumptions and has its own credibility and scope of application.

In work in this area, most recently supporting military utility analysis on the DARPA Ground Vehicle Experimental Technologies (GVX-T) program, a version of hierarchical FSM modeling proved effective and useful (figure 7.5).

Stateflow models are a further extension in which the rate of transition processes depends on the levels of resources, and processes & events change the levels of resources. This formulation requires a first-order model of the rate of transition events based on the levels of resources, and the consumption/production of resources in states and at transitions. Stateflow models have been effectively used to represent opponent populations with discrete commanders’ decisions for operational effectiveness analysis [11]. The level of detail of stateflow models may be much greater than what is needed for behavior requirements TEVV.

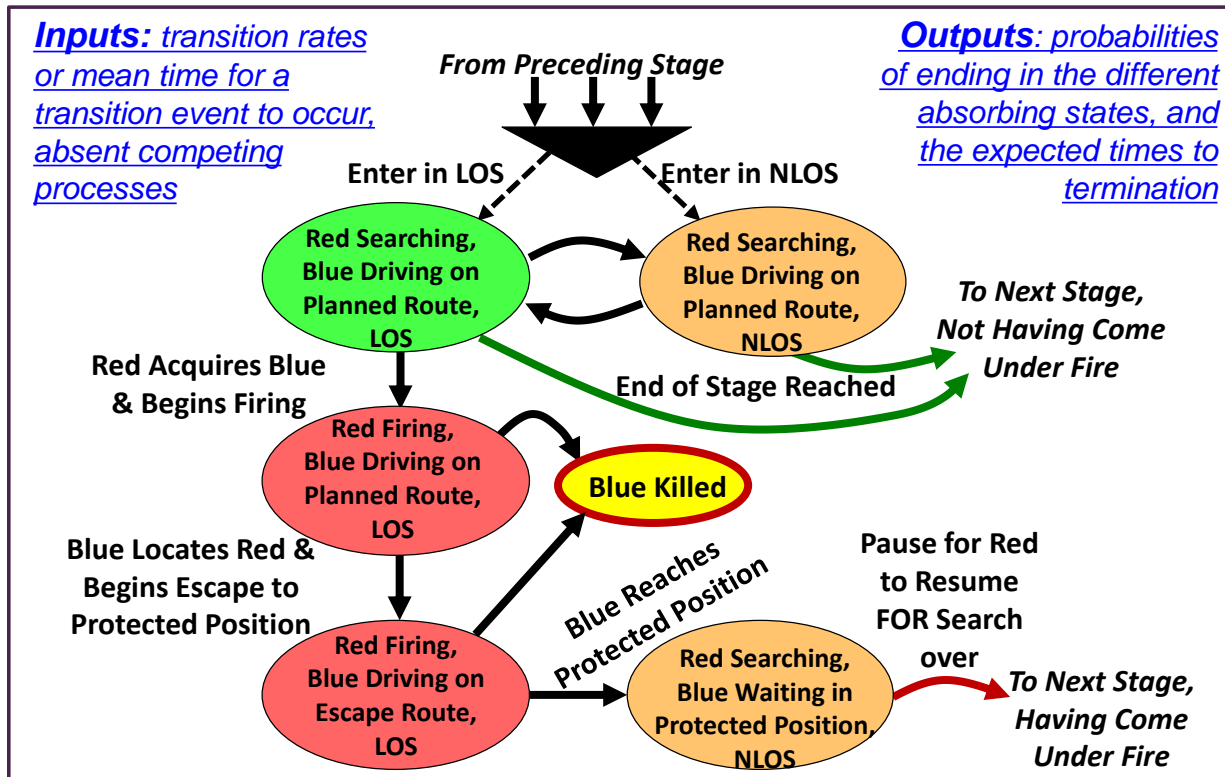


Fig. 7.5: Example Node in a Hierarchical Stateflow Model of a Military Operation

FSM and stateflow models can become complex if there is an attempt to have both high resolution and broad scope in the same model. In some cases, complexity can be managed by hierarchical modeling. In team and adversarial relations, parallel multi-agent modeling reduces the complexity to single agent behavior.

Additional research and development is needed to specify the appropriate format for an operational dynamics model. The Ptolemy II system provides a sound theoretical basis for hierarchical heterogeneous models of computation [12].

7.3 PROGRAM PLAN

The project will refine and integrate the assurance argument calculus model, the behavior requirements model, operations dynamics model, and the behavior executive model. It will produce an automatic behavior-checker using the behavior executive, behavior requirements and operations dynamics models to identify inconsistencies, gaps, ambiguities, and incorrect behaviors.

Co-Development Partner

We propose to work with TARDEC as co-developer and transfer partner for the SE MPT, with developmental testing application in their autonomous convoy project. TARDEC has endorsed the collaboration.

TARDEC has a current Science and Technology (S&T) project for an autonomy-enabled convoy system. The project has dual objectives to (a) produce a demonstration system, and (b) to develop and test SE MPT for autonomy-enabled systems. TARDEC recognizes that development of SE MPT for AES is needed in parallel with AES technology development. The US Army RDECOM S&T has allocated additional funding to this project beyond what has been committed from TARDEC core funding. In RT148, TARDEC's autonomous convoy project funded a project on MPT to trace and analyze modeling and simulation (M&S) in model-based engineering workflow for AES.

In the autonomous convoy project, TARDEC is developing and testing M&S requirements and capabilities to support system concept and requirements development – the turquoise region of figure 7.2 (detailed simulations, virtual exercises, developmental testing, etc.)

TARDEC is using a version of “Early Synthetic Prototyping” (ESP) warfighter experiments to explore how warfighters will interact with and use autonomous systems and work as part of an autonomy-enabled-units. The goal is to use early virtual experiments to test and inform operational concepts for AES, requirements, and benefits. The SE challenges include (a) design of virtual experiments, (b) data mining from virtual experiments, and (c) understanding the credibility and limitations of findings.

Complementary Funding

In September, we submitted a proposal to the Automotive Research Center (ARC) – TARDEC's university arm, led by University of Michigan, of which WSU is an active member. The ARC is focused on M&S – what M&S is needed to address acquisition challenges and opportunities, and how to integrate it into system acquisition. At the beginning of the summer, TARDEC gave the ARC the challenge of developing research proposals to support development and acquisition of autonomy-enabled ground vehicles.

ARC proposals always include Government and Industry oversight participants to help ensure military and dual-use relevance. Our proposal to the ARC has oversight participation from *DASD-SE*, the *TARDEC autonomous convoy team*, and a *Tier 1 automotive supplier* with an R&D program in SE for autonomous safety functions.

The proposed WSU project is to finalize and implement a rigorous and extended assurance argument model (the yellow top level of figure 7.2) for AES. *The scope of the ARC proposal does not include anything below the yellow “Assurance Argument Paradigm” in figure 7.2.* The scope was to develop and implement the MPT rigorous assurance arguments, and demonstrate them, in the autonomous convoy project context.

The scope focused on developing the hierarchy of claims and considerations needed to:

- Help develop RFP solicitation packages to ensure the Government gets information needed to make source-selection decisions, and inform bidders how the information will be used (in order to avoid protests)
- Probe limitations of the behavior requirements
- Compare competing bids and diagnose strengths and limitations on a level playing field

Over the Spring/Summer/Fall we participated in a series of four full-day workshops hosted by the ARC on M&S for AES. The workshops were attended by TARDEC, academic researchers from the six ARC universities, and the automotive industry. We continued our engagement with the TARDEC autonomous convoy team on SE for autonomy-enabled convoys and convoy vehicles.

Based on the presentations and comments by TARDEC, and the “brainstorming” of M&S opportunities and needs by the academics, we are confident that the research in our proposal to the ARC and in this to the SERC are

- Highly relevant
- Not duplicative or overlapping with other ARC research topics

Schedule and Milestones

- Year 1
 - Produce & demonstrate initial software implementation of rigorous assurance argument paradigm in Matlab
 - Develop high-level of the assurance arguments for major capability claims for trustworthy AES, using TARDEC’s autonomous convoy project as a case study, to include sub-claims, evidence, and assumptions with corresponding composition calculus
 - Define information interfaces between the assurance argument model and the operations dynamics model; select the initial paradigm for the operations dynamics modeling
 - Specify the initial behavior requirements expression “language”
 - Produce initial requirements for the behavior executive
- Year 2
 - Refine/extend software implementation of rigorous assurance argument paradigm as needed
 - Demonstrate and assess high-level of the assurance arguments for major capability claims for AES, using TARDEC’s autonomous convoy project as a case study, using the software implementation of rigorous assurance argument paradigm
 - Implement operations dynamics model framework, with and demonstrate it for selected convoy operations
 - Implement behavior requirements expression model, with example behaviors for selected convoy TTP
 - Illustrate derivation of situation awareness requirements
 - Produce draft guidelines and pseudo-code for the behavior executive and behavior checking software
- Year 3
 - Finalize initial versions of software tools
 - Conduct and document TEVV analysis of the behavior requirements
 - Hold a workshop to assess value, practicality and suitability of the MPT
 - Plan and execute transfer of software tools and user’s manuals, MPT documentation

Funding Request

The total funding request for the full project is \$160K/year to WSU, over 3 years.

The WSU ARC proposal is for \$80K/year over 3 years. Funding review is in process, with announcements expected in March 2018.

If the ARC proposal is funded by TARDEC, its scope can be excluded from the proposal to the SERC.

If the ARC/TARDEC provides \$80K/year to WSU, then the need from SERC/DASD-SE is for \$80K/year to WSU.

7.5 CONCLUSIONS AND RECOMMENDATIONS

AES an important area for the technology offset strategy. It is an area of high interest to our long-term co-development partner, TARDEC.

SE MPT to reduce AES development time is vital to effective technology offset. Incorrect, incomplete, and inconsistent autonomous behavior requirements are an expected source of time and cost overrun. TEVV of the autonomous behavior requirements is a leverage point to minimize rework and delay.

We have a co-development partner, TARDEC, and an S&T demonstration project on which to apply the SE MPT.

There is potential cost-sharing from TARDEC through the WSU proposal to the ARC.

The proposal is practical and relevant. It addresses current needs of acquisition agencies, with TARDEC as an example, to develop MPT for “cognitive” behavior requirements TEVV, in consideration of the difficulties in specifying AES behaviors.

The proposed project build on R&D we have been doing across multiple projects, and is an opportunity to bring the lessons and techniques together into a coherent whole. The technical approach extends MPT that we have used effectively on the DARPA Ground Vehicle Experimental Technology (GVX-T) program. The MPT under proposed development are evolutionary, step-wise improvements on and integration of existing tools and foundations.

The bottom line: Shorten AES development time by use SE MPT to assess the inconsistencies, gaps, ambiguities, incompatibilities, and incorrectness in the autonomous behavior requirements

7.6 REFERENCES

- [1] Chris Wilkinson, Jonathan Lynch, Raj Bharadwaj, and Kurt Woodham. (2106) *Verification of Adaptive Systems*. DOT/FAA/TC-16/4.
- [2] David Rinehard, John Knight and Jon Rowanhill. (2015) *Current Practices in Constructing and Evaluating Assurance Cases With Applications to Aviation*. NASA/CR-2015-218678.
- [3] John Rushby. (2105) *The Interpretation and Evaluation of Assurance Cases*, SRI International, Technical Report SRI-CSL-15-01.
- [4] Charles Weinstock and John Goodenough. (2009). *Towards and Assurance Case Practice for Medical Devices*. CMU/SEI-2009-TN-018.
- [5] Ben Calloni. (2015). *System Assurance and Related Standards*. Object Management Group.
- [6] Mario Gleirscher and Carmen Carlan. (2017) *Arguing from Hazard Analysis in Safety Cases: A Modular Argument Pattern*, Proc. Conf. High Assurance Systems Engineering (HASE), 2017 IEEE 18th International Symposium.
- [7] Sandeep Neema. (2017). *Assured Autonomy*. DARPA.

- [8] Richard Hawkins, Ibrahim Habli, Dimitris Kolovos, Richard Paige, Tim Kelly. (2015) *Weaving an Assurance Case from Design: A Model-Based Approach*. High Assurance Systems Engineering (HASE), 2015 IEEE 16th International Symposium.
- [9] Ewen Denney, Ganesh Pai. (2015) *A Methodology for the Development of Assurance Arguments for Unmanned Aircraft Systems*. NASA Ames Research Center, Proc. 33rd International Safety Systems Conf.
- [10] A. Abdulkhaleq, P. Bluehr, D. Lammering. (2017). *Using STPA in Compliance with ISO26262 for Developing a Safe Architecture for Fully Automated Vehicles*. Proc. STAMP Workshop, MIT
- [11] Seth Bonder. (2002). Army Operations Research—Historical Perspectives and Lessons Learned. *Operations Research*. 50(1):25-34.
- [12] C. Ptolemaeus, ed. (2014) *System Design, Modeling, and Simulation Using Ptolemy II*, Ptolemy.org.

8.1 PROBLEM STATEMENT, MOTIVATIONS, AND OBJECTIVES

In recent years, cyber-physical systems (CPS) have achieved widespread application in diverse areas including: civil infrastructure, energy, healthcare, transportation, automotive, smart appliances, and others [Rajkumar et. al. 2010]. Usually cyber-physical systems are composed of diverse subsystems consisting of both physical and software components developed by different vendors. These components are deeply intertwined at various levels of abstraction (e.g. data flow, physical connections, and logical connections etc.) under changing contexts to achieve the desired functionalities and the respective quality attributes, such as performance, security, scalability, maintainability, etc. With the advance of technology, the recognition of new consumer needs, and the detection of deficiencies in current systems, components need to be upgraded, replaced, and fixed---frequently, in many domains. The key question is: can the upgrade, replacement, or problem fix happen quickly and without disrupting the rest of the system?

To address this issue, stakeholders, such as the US Department of Defense, have increasingly emphasized modular and open approaches to system development. According to Baldwin and Clark [2000], modularity allows for both independence of structure and integration of functions in large and complex systems. The goal of modularity is to improve interoperability, facilitate system evolution and technology insertion, and foster competition. Requirements such as the Modular Open Systems Approach (MOSA) have been imposed on acquisition efforts. However, it can be difficult for the stakeholders to assess whether the resulting architectures and systems are truly modular. In other words, can modules in a CPS actually be upgraded, replaced, and fixed in a plug-and-play manner without affecting one another to maximally leverage the benefits of modularization? In the traditional software engineering field, it has been observed that it is often not the case. Modules without any explicit structural dependencies were seen frequently changing together when new features or bug fixes were implemented. Wong et. al. [2009] call this phenomenon a *Modularity Violation (MV)*. Studies of real-life open source projects indicate that up to 85% of bug-fixing efforts in a software system involve modularity violations [Xiao et. al. 2016]. An in-depth analysis revealed that modularity violations usually result from and imply latent connections among software modules. For example, it could be an implicit assumption regarding the usage of time units in two modules [Schwanke et. al. 2013]. In software systems, the consequences of modularity violations could be higher bug-rates (when the assumptions are not consistent among different parties) [Mo et. al. 2015] and increased maintenance costs (in order to keep the assumptions consistent) [Xiao et. al. 2016]. It is possible that that modularity violations could be even more harmful in cyber-physical systems. Modularity violations, derived from the latent relationships among software and hardware components, prevent the long-term evolution, maintenance, and success of cyber-physical systems. In particular, due to the inflexibility and high cost of evolving hardware modules, modularity violations in cyber-physical systems could more easily lead to vendor-lock-in compared to a pure software environment.

The objective of this research is to develop techniques, metrics, and models that would allow stakeholders to detect, measure, and understand modularity violations in developed and acquired cyber-physical systems. We organize modularity violations in cyber-physical systems into three different types: 1) software vs. software, 2) software vs. hardware, and 3) hardware vs. hardware. The first type of modularity violation has been extensively investigated in prior work [Wong et. al. 2009; Schwanke et. al. 2013; Xiao et. al. 2014; Ran et. al. 2015; Xiao et. al. 2016]. There, the approach to identify and measure modularity violations relies on the analysis of source code and operational data such as maintenance activity records. These are automatically and comprehensively tracked in version control systems. The

challenge to extending this type of analysis to identify modularity violations involving hardware is the lack of systematic records of operations involving hardware components. Unlike, software projects that use standard version control systems to keep track of who made what changes to which parts at what time and for what reason, changes to hardware components are not always tracked in similar detail or are not as easily accessible.

Given these limitations, the question naturally follows, could a subset of hardware modularity violations be inferred from an analysis of records of software changes? More specifically, some software changes may be a consequence of hardware related modularity violations. If that were the case, analysis of version control systems could be used to detect some hardware related modularity violations outright and flag other potential violations for further investigation. The issue is whether or not there is actually enough information in a version control system to infer a hardware modularity violation. To evaluate the feasibility of this idea, the incubator phase of this project involved the analysis of two open source cyber-physical systems: OpenWrt and MD-PnP, which aim to develop a common software infrastructure for the Internet of Things (IoT) and medical systems respectively. To analyze these systems, we developed a keyword-based heuristic to help us identify software-hardware modularity violations, where the software artifacts change due to hardware related issues. We identified 70 software source files that are potentially involved in software-hardware level modularity violations because the changes made to these files involved hardware related concepts. We conjecture that hardware-related concepts that propagate changes to software artifacts imply a potential modularity violation at the software-hardware level. Consequently, the incubator demonstrated the feasibility of identifying potential modularity violations in a cyber-physical system by analyzing a software version control system.

Given that the incubator established the plausibility of the analysis, several additional steps are required to achieve the ultimate goal of developing methodologies, metrics, and tools to detect and assess modularity violations in cyber-physical systems. First, the incubator focused on source files as the unit of analysis. However, in practice, the definition of a module will depend on the nature of system and the goals of different stakeholders. In some systems, stakeholders may be willing to trade modularity for performance. In other cases, it may be a matter of perspective. For example, low-level developers may view modules as hands-on and manageable working elements to implement functional details (e.g. a particular source file and a particular hardware board). In comparison, a project owner views modules as cohesive functional components to deliver the product value and competitiveness. Thus, there is a need to be able to handle modules of different granularity in the analysis. Second, the keyword-based heuristic developed for one project domain may not be applicable to a project in another domain. Third, there is not yet an explanatory model that helps stakeholders to understand the reasons and context behind the modularity violations. Without such explanatory model, it is difficult for stakeholders to mitigate current and future violations.

To achieve the ultimate goal of allowing the stakeholders to detect, measure, and understand modularity violations in cyber-physical systems, we propose the following activities:

- Develop strategies to decompose a CPS into modules of different granularities. The investigation of modularity violations first relies on the understanding of what is a “module”. We argue that the scope/granularity of modules is determined by the stakeholder who views the system for addressing a particular concern [Kruchten 1995].
- Develop a systematic approach to identify modularity violations in CPS. This approach should be able to scale up or scale down to different module granularity levels. This approach should also be adaptive and generally applicable to different problem domains.

- Develop a software-based proof-of-concept demonstrator to evaluate the effectiveness of the approach developed above.

8.2 CURRENT PRACTICE AND ITS LIMITATIONS

8.2.1 MODELING CPS AND ITS LIMITATIONS

The term “cyber-physical systems” emerged around 2006, when it was coined by Helen Gill at the National Science Foundation [Lee 2015]. Gill defined a cyber-physical system (CPS) as an integration of computation with physical processes. However, one of the challenges to designing and managing cyber-physical systems is that there are techniques to represent either the cyber processes or the physical processes, but not both [Baheti and Gill 2011]. From the “cyber” perspective, there are different techniques to represent the architecture of software systems, such as architecture description languages, UML models, and component models. From the physical perspective, there are different traditional engineering techniques to model the development of physical systems. One study on architecting cyber-physical systems found that, currently, researchers tend to focus on a specific attribute of interest rather than assessing the cyber-physical architecture as a whole [Malavolta, et al. 2015].

However, there are some instances of research that take a more holistic approach to architecting cyber-physical systems. Rajhans, et al. [2009] contributed a new cyber-physical architectural style to present the interconnections and interactions between physical and cyber components. They defined three related families of general components and connectors pertaining to the cyber domain, the physical domain, and their interconnections. From the architectural assessment perspective, Sinha [2014] developed a theoretical framework for structural complexity quantification and its implications for the design of cyber-physical systems. Derler et al. [2012] focused on the challenges of modeling cyber-physical systems that arise from the intrinsic heterogeneity, concurrency, and sensitivity to timing of such systems. They described some promising approaches that use domain-specific ontologies to enhance modularity and jointly model of the functional and implementation architectures. Finally, Cristalli et al. [2016] provides a representative example of a modular approach to developing a cyber-physical system through their modular design for a Smart Robotic Cell, composed by several interconnected sub-systems.

What all of the above have in common is a focus on designing cyber-physical systems to be modular. However, there is no guarantee that the realized system will exhibit the intended modularity. The need to assess the actual modularity of a cyber-physical system is particularly acute. As Lee [2008] points out, cyber-physical systems have always been held to a higher reliability and predictability standard than general-purpose computing. Without reliability and predictability, cyber-physical systems will not be applied in safety-critical domains like traffic control, automotive safety, and healthcare. While a modular design is just one approach to improve reliability, predictability, and maintainability; when it is employed, one would like to know how well that was achieved. While approaches to assess modularity in pure software systems have been developed, to the best of the investigators’ knowledge no approaches have been developed that specifically address the unique challenges of assessing the modularity of cyber-physical systems. Thus, the question is whether existing approaches from the software domain can be adapted to the cyber-physical domain.

8.2.2 MODULARITY VIOLATION DETECTION IN SOFTWARE SYSTEMS AND ITS LIMITATIONS

The term modularity violation was first proposed by Wong et al. [2009] to describe the phenomenon where independent software modules frequently change together during the evolution of a system in the process of fixing bugs or adding features. Methodologies and tools [Xiao et al. 2014] have been built for analyzing modularity violations in software systems. The identification of a modularity violation follows three steps:

1. *Reverse-engineering the modular structure of a software system from the code base:* The modular structure of a software system can be calculated based on the structural dependencies among software entities. Xiao et al. developed a new architectural representation, called Design Rule Space (DRSpace) modeling to capture the modular structure of a software system as multiple overlapping design spaces [Xiao et al. 2014], applying Baldwin and Clark's [2000] design rule theory. The modular structure can be visualized in the form of a DSM (Design Structure Matrix). The items in the DSM represent software entities, the cells represent the structural dependencies among entities. The entities can be clustered into modules based on selected dependency types for supporting analysis of different focuses.
2. *Extracting the evolutionary coupling (i.e. the number of co-changes) between software modules:* This can be calculated based on the data recorded in version control systems, which automatically keep track of all the changes made to software entities, in terms of who at what time made what changes to which entities/modules for what reason. Analyzing such data helps to extract the evolutionary coupling between any two software entities: how many times they are modified together in the course of maintenance activities. The more frequently two entities are modified together, the stronger is their evolutionary coupling---implying latent connections.
3. *Calculating the discrepancy between the above two data sets to point to modularity violations among software modules:* If two software entities/modules are structurally independent according to step 1, but they have high evolutionary coupling according to step 2, there is a potential modularity violation.

According to prior empirical studies of industry software systems, modularity violations usually indicate implicit assumptions shared among software modules. For example, it could be an implicit assumption regarding the usage of time units in two modules [Schwanke et al. 2013]. In software systems, the consequences of modularity violations could be higher bug-rates (when the assumptions are not consistent among different parties) and increased maintenance costs (in order to keep the assumptions consistent) [Mo et al. 2015; Xiao et al. 2016].

However, the above-mentioned approach is limited in that it only addresses software systems implemented in a single programming language. Complex hybrid systems are composed of multiple heterogonous hardware and software sub-systems. Such systems are particular difficult to analyze because of data inconsistency and heterogeneity among the various sub-systems. This incubator phase of this research tested the feasibility of leveraging software maintenance data to infer hardware related issues that are indicative of software-hardware modularity violations. We propose to formalize and extend this approach to be applicable to a wide range of domains.

8.3 RESEARCH APPROACH

The research effort is divided into two phases. The first phase is to evaluate the feasibility of inferring software-hardware modularity violations from software maintenance actions. This incubator phase,

lasting from July 2017 to November 2017, has been completed. The second phase is to formalize and generalize the concepts developed during the incubation phase. This phase consists of 3 parts: 1) examine the criteria to decompose a CPS into modules of different levels of granularities for addressing separate concerns of stakeholders; 2) build a domain concept learning approach, leveraging natural language processing techniques, to identify CPS modularity violations in different domains; and 3) Incorporate the results from the previous two parts into a decision framework and develop a software-based demonstrator system to identify CPS modularity violations.

8.3.1 RESULTS OF THE INCUBATOR STUDY

As discussed earlier, the identification of modularity violations in a CPS is a challenge because maintenance operations on hardware components are not as comprehensively recorded as in traditional software systems. However, we hypothesized that software components that are changed frequently due to hardware related concepts are indicative of potential modularity violations.

As a case study for feasibility, we chose two real life software infrastructures, OpenWrt and MD-PnP, for cyber-physical systems. OpenWrt is a Linux based core that is commonly used to support the Internet of Things (IoT). The MD PnP provides the software infrastructure for plug-and-play medical devices to improve patient safety. The following table summarizes basic facts of these projects. Table 10 shows the scale (measured by number of files, methods, and lines of code), the development team size, and the age (measured by the number of revisions and starting time). We chose these projects because they are industry scale projects offering sufficient research data.

Table 10: Case Study Projects

Project	#Files	#Methods	LOC	#Developers	#Revisions	History
OpenWrt	1052	6061	163114	137	38099	Since 2004
MD PnP	866	7872	73616	12	1605	Since 2013

Based on our analysis of these two projects, we made the following observations that show the feasibility of identifying meaningful modularity violations in a CPS based on maintenance data from the software side. In what follows we will use OpenWrt to present the findings.

8.3.1.1 Observation 1: the OpenWrt is more modularized than about 90% of the 129 (commercial and open source) traditional software systems we studied before

First, we want to understand how well OpenWrt is modularized. To do that, we reverse-engineered the code base of the projects. Figure 1-a shows an overview of the modular structure identified in OpenWrt, represented in the form of a DSM [Clark and Baldwin 2000]. This DSM is square matrix showing the structural dependencies among source files in OpenWrt. The rows and columns represent source files, and a black dot represent a structural dependency from the row to the column. To get an overall understanding of how modularized OpenWrt is based on the structural dependencies among source files, we calculated two metrics: 1) the Propagation Cost and 2) the Decoupling Level, based on the DSM.

Propagation Cost proposed by MacCormack et al. [2006] measures the density of the n-transitive closure of the DSM. The maximum value is 100%, meaning that every element is connected (directly or transitively) to another element in the system. The lower the value, the less coupled the system is. This

implies that the system is composed of independent modules. **The PC value for OpenWrt is only 1.4%, which is lower than about 90% of the 129 (commercial and open source) traditional software systems we studied before.**

Decoupling Level proposed by Ran et. al. [2016] measures how well a system is decomposed into small and manageable modules that can evolve independently from each other. The highest possible value is 1, meaning the system is perfectly modularized (which is not likely in practice). **The DL value for OpenWrt is 0.78. This is also higher than 90% of the 129 software projects we studied before.**

The implication is that the software components in this particular CPS are more modularized compared to traditional software systems. We conjecture that this is because the hardware components in a CPS increase the overall complexity of the system relative to pure software systems. Increasing the modularity of the software may be a way to cope with that complexity. Furthermore, by talking to IoT experts, we realized that the first priority of the software components is to stay concise to reach the quality goals of energy and cost efficiency in IoT systems.

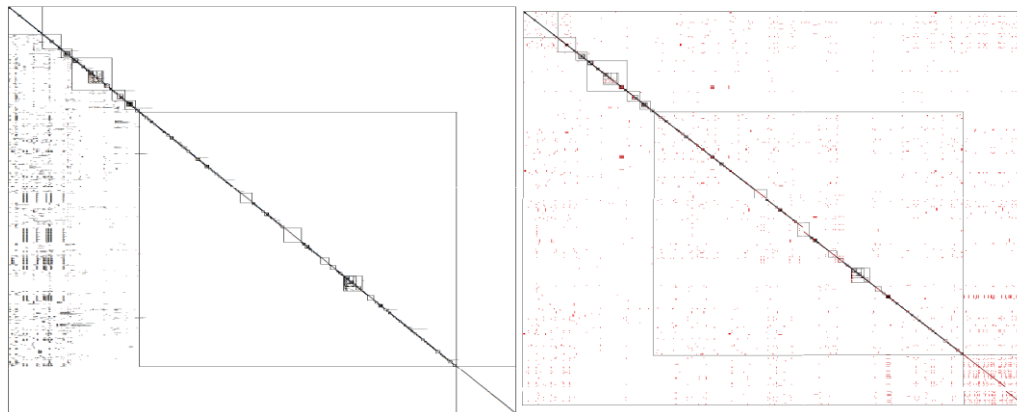


Figure 1-a

Figure 1-b

Figure 1:

1-a: the overview of the modular structure

1-b: the overview of changes made to the system

8.3.1.2 Observation 2: Software-Software modularity violations in OpenWrt include hardware related information in the naming conventions of the involved source files.

We applied the approach described in Section 8.2.2 [Xiao et. al. 2014; Mo et. al. 2015] to identify software-software level modularity violations. The goal was to determine whether software-software level modularity violations in CPS systems imply hardware related issues. If so, the hardware concepts are potential causal factors of modularity violations among software components in a CPS. As shown in Figure 1-b, we calculated the evolutionary coupling among source files in OpenWrt. The evolutionary coupling between two source files is the number of times the two files change together in the revision history. Each red dot in Figure 1-b indicates the evolutionary coupling between the file on the row and the file on the column. Actually, the weight of most of the evolutionary coupling in Figure 1-b is below 4, indicating software source files do not change together frequently. We computed the discrepancy between Figure

1-a and Figure 1-b, which resulted in Figure 2, containing 23 source files that exhibit evolutionary coupling above 4, but are structurally independent from each other. We consider these 23 source files as potential modularity violations, which indicates shared but latent assumptions among them.

Further inspection of the naming conventions of the source files indicate that they share hardware-related concepts. For example, from row 1 to row 10, the naming of the files all contain “mips”, which is a typical microprocessor architecture for CPS. In addition, row 20 to 23 shows that the file names all contain “firmware”, which is held in non-volatile memory devices, such as ROM or flash memory. Based on these observations, we conjecture that the hardware concepts are actually the causal factors of these software-software level modularity violations in OpenWrt.

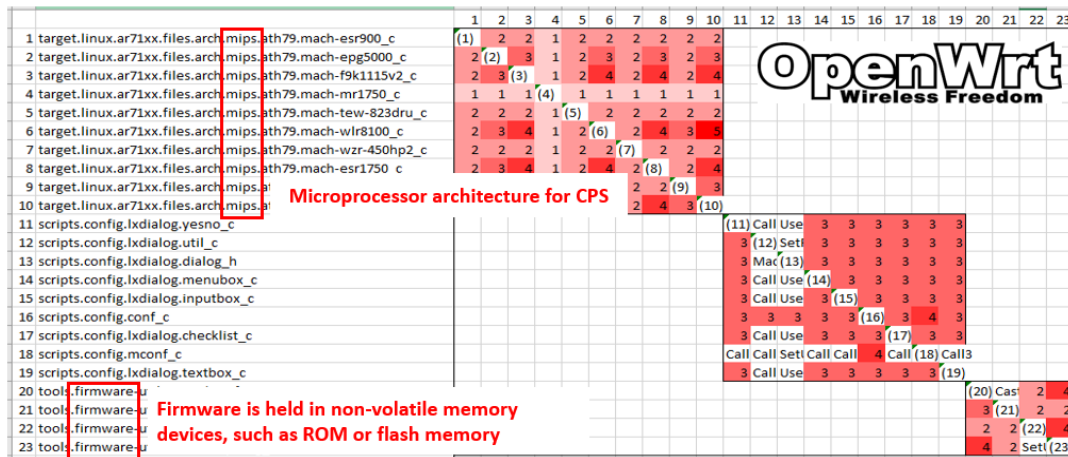


Figure 2: 23 files involved in software-software MV

8.3.1.3 Observation 3: Hardware-related concepts are the main contributing factors to Software-Hardware MV in OpenWrt.

Reasoning based on observation 2, we infer that hardware related concepts could also be the contributing factors for software-hardware MV. That is, software components that change frequently due to hardware related concepts. As an example, we find this comment when developers changed a software entity: “set *chip* type directly in ar8216_id_chip”, where chip is obviously a hardware term. We assume it is less likely the other way around: software concepts contribute to hardware changes, because it is, in most cases, more affordable to change software components.

We manually extracted 16 key words from the commit messages and hardware devices supported by OpenWrt. The details are shown below in Figure 3, containing keywords, like “radio”, “WiFi”, “zigbee”, etc. By matching these manually extracted keywords in developers’ change comments (which are usually used to explain why they made the change), we identified 71 source files in OpenWrt (of the 1052 total files) are potentially involved in software-hardware MV.

Figure 3: 16 manually extracted words

“radio”, “WiFi”, “zigbee”, “btle”, “mips”, “ramips”, “mtd”, “broadcom”, “routerboot”, “router”, “firmware”, “bluetooth”, “energy”, “power”, “soc”,

extracted

Figure 4 shows the change-proneness levels of the 71 files involved in modularity violations versus the average files in OpenWrt. The data show that these 71 files are twice likely to change compared to average files. Therefore, the hardware related concepts could be the main contributing factor to changes made to software modules.

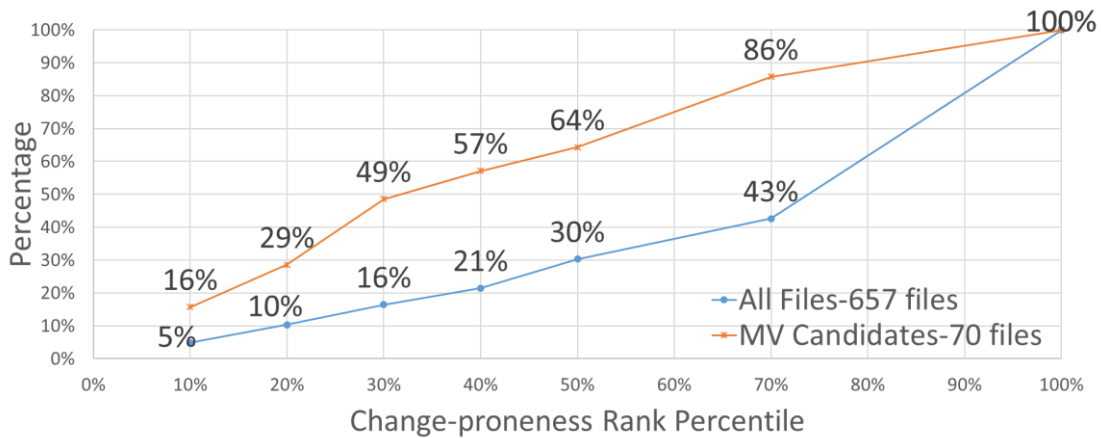


Figure 4: Hardware concepts are important change factors

We made similar observations in MD PnP. However, we found that the keywords identified for OpenWrt do not directly apply to MD PnP. The reason is that these two projects are in two completely different problem domains. OpenWrt is for supporting IoT. Therefore, the keywords are mostly related to network devices and sensors. However, MD PnP is in the medical domain. Therefore, the proper keyword sets to identify MV are completely different. This finding motivated us to propose the application of natural language processing mentioned previously. Doing so will enable an analysis approach applicable to cyber-physical systems in different domains.

8.4 PROPOSED WORK IN THE NEXT PHASE

Motivated by the observations and limitations identified in the incubator phase, we propose the following research activities for the second phase:

8.4.1 EXAMINE THE CRITERIA TO DECOMPOSE A CPS INTO MODULES

In the incubator phase, we treated source files as the granularity of a module. This is appropriate from the perspective of a low-level developer. However, a project owner views modules as cohesive functional components to deliver the product value and competitiveness. This results in a very different definition of a module. From that perspective, a module may contain many source files, and the project owner may not be concerned with modularity violations among source files within a given module.

The goal of this research task is to investigate the pertinent criteria to decompose a CPS into modules of different granularity levels. We propose investigate the following approaches for defining a module:

- *Modules for atomic working tasks*: each module can be mapped to a working task of developers. This was the strategy tested in the incubator phase. The study based on this strategy has provided meaningful insights for modularity.
- *Modules based on coupling*: each module is composed of a group of atomic modules that are structurally coupled with each other. This strategy helps project managers to understand the high-

level modules that map to sub-systems/sub-projects for different vendors to maximize the benefits of modularization.

- *Modules based on cohesion:* each module is composed of atomic modules that share semantic similarities and thus are likely to serve a cohesive purpose. This strategy serves to help understand the functional modules of a system.
- *Modules based on the system architecture:* These modules are identified from the top-down based on design documents or stakeholders' perceptions. This helps stakeholders to understand whether a sub-system that was designed to be modular actually is.

Based on the approach of interest, we can calculate and evaluate the decomposition of modules in a CPS. From there the analysis of modularity can be adapted to each approach.

8.4.2 BUILD A "DOMAIN CONCEPT LEARNER" TO IDENTIFY MODULARITY VIOLATIONS IN DIFFERENT DOMAINS

Based on the approaches to identify modules in a CPS, we can scale-up and scale-down the modularity violation analysis to address different levels of modular granularity. However, as mentioned earlier, in the incubator phase, we realized that the keyword heuristics developed for one project domain may not be applicable to another project domain. Therefore, this activity aims to develop a systematic approach that can be applied to different problem domains.

We plan to develop a "Domain Concept Learner" that leverages natural language processing techniques. The inputs to the concept learner are supporting wiki-pages or documents that record domain-related hardware components. For example, OpenWrt maintains a "Table of Hardware" (<https://wiki.openwrt.org/toh/start>) on the wiki-page. We can then build a Topic Model to summarize the key hardware concepts, keywords, and the associated probabilities of keywords referring to hardware concepts.

Using the resulting concept dictionary under different domain contexts and hardware environments, we can apply the approach to identify potential modularity violations to CPS projects in different domains.

8.4.3 BUILD DECISION FRAMEWORK AND DEMONSTRATOR

Last but not least, we plan to incorporate the methods developed in prior steps into a decision framework to increase their utility for decision makers. This decision framework will accommodate the detection and analysis of modularity violations at different modular granularities and in different problem domains. The ultimate goal is to provide decision making support to improve modularity where appropriate and achieve the associated benefits.

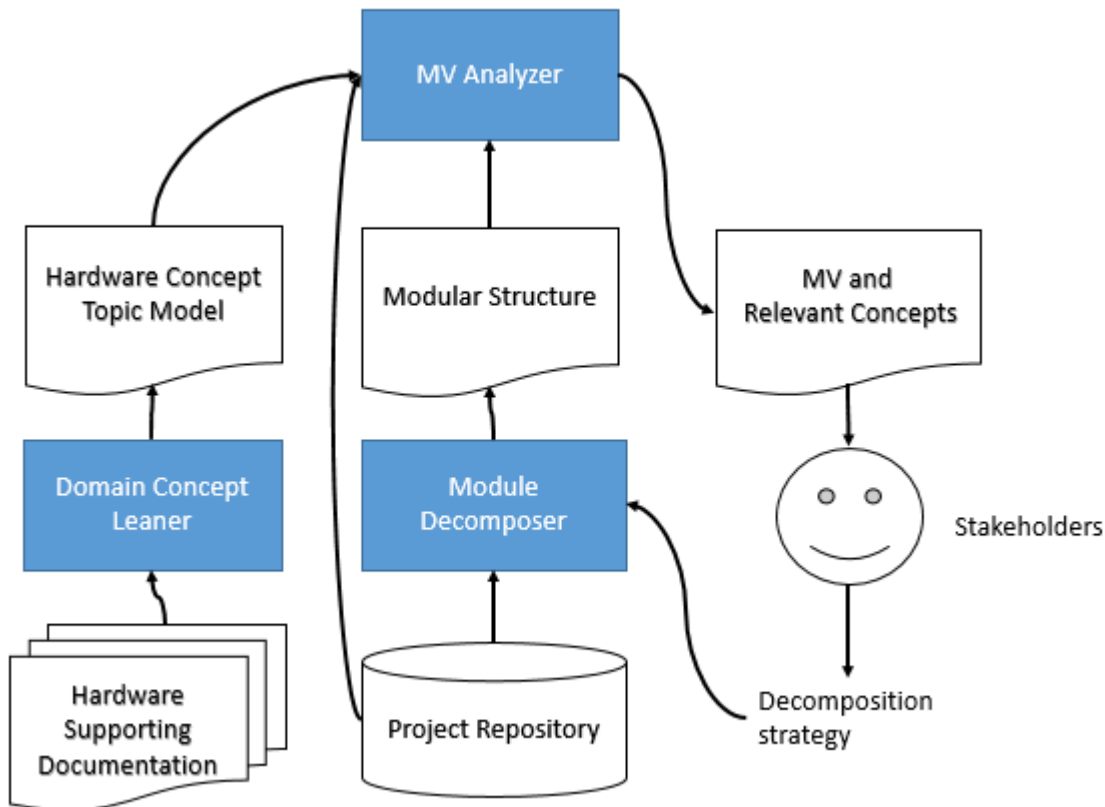


Figure 5: Decision Framework Overview

Figure 5 shows the overview of the decision framework for analyzing modularity violations, increasing awareness, and supporting decision making. The framework contains three processing components as the output of the above research activities.

- *Module Decomposer*: Takes the project repository and user’s decomposition strategy as input to calculate a modular decomposition based on user’s preferences.
- *Domain Concept Learner*: Takes the hardware supporting documentation, such as wiki pages or design documents as input to learn the topic model of the domain hardware concepts.
- *MV Analyzer*: Leverages the calculated hardware concepts and the modular decomposition as input to identify potential Modularity Violations involving hardware concepts. The stakeholder will see the potential modularity violations identified in their projects. This will enable them to be aware of potential modularity violations and understand the underlying hardware concepts that triggered changes to software modules. The ultimate goal is to help stakeholders to make re-structuring decisions to maximize the benefits of modularization.

8.5 RELEVANCE OF THE RESEARCH

Since the term first emerged in 2006, cyber-physical systems have achieved widespread application in diverse areas, such as civil infrastructure, healthcare, transportation, and national defense. In particular, the application of cyber-physical systems in the national defense domain is highly relevant to DoD (Department of Defense). It has been recognized that the design, modeling, maintenance of cyber-physical systems is more challenging than for traditional engineering systems, due to the intrinsic heterogeneity and nondeterministic nature of the interactions among cyber and physical components

[Lee 2015; González 2015; Jantunen et. al. 2016]. Existing engineering techniques either focus on the cyber side or the physical side, but not both. The proposed work tackles one aspect of this problem by identifying the latent connections between the cyber and physical aspects of the system. More specifically, it would identify the shared concepts that potentially propagate maintenance actions between the cyber and the physical sides. The incubator study results have shown the feasibility of this approach, and revealed that it is more likely that the direction of propagation is from the physical side to cyber side. Therefore, this research offers stakeholders in DoD a new approach to understand how the cyber and physical sides interact and impact each other.

In addition, DoD has increasingly emphasized modular and open approaches to system development. The goal of modularity is to improve interoperability, facilitate system evolution and technology insertion, and foster competition. Requirements such as the Modular Open Systems Approach (MOSA) have been imposed on acquisition efforts. However, it can be difficult for the stakeholders to assess whether the resulting architectures and systems are truly modular. In other words, can modules in a CPS actually be upgraded, replaced, and fixed in a plug-and-play manner without affecting one another? The proposed research helps stakeholders to answer this question based on the analysis of the modular structure with tunable granularity that fits specific stakeholders' concerns. Any identified latent connections between cyber and physical modules could prevent the upgrade, replacement, or maintenance of as-known "modules" in a plug-and-play manner. Our research considers these latent connections as modularity violations because the structure of a cyber-physical system is no longer truly modular. Modularity is not purely for the sake of modularity, but for achieving the benefits, such as interoperability and to foster competition. The latent connections among modules in cyber-physical systems can easily result in vendor-lock-in, reduce competition, and increase cost. Therefore, this research is highly relevant to the goal of the Modular Open Systems Approach emphasized by DoD.

8.6 EXPECTED CONTRIBUTION

The ultimate goal of this research is to help stakeholders of a cyber-physical system gain an understanding of the system's modular structure, and reveal the associated modularity violations within that system. This serves to support decision making, in terms of whether a re-design or re-structuring is needed. This research expects to offer the following concrete contributions:

- *A tunable granularity analysis of the modular structure of a given cyber-physical system.* Different levels of granularity address the concerns of stakeholders at different organizational levels. For example, low-level developers view the modular structure at the level of workable atomic tasks. Product owners view the modular structure at the cohesive functional level, where each module is composed of a set of related atomic elements. This study will result in a flexible and tunable strategy to decompose a complex system into modules based on different modularity strategies.
- *A systematic approach to identify potential modularity violations by analyzing cyber maintenance actions that are due to physical concepts.* This approach will be applicable to different problem domains leveraging an automated domain concept learner. This approach will help identify modularity violations that potentially lead to vendor-lock-in and high maintenance costs for stakeholders, without requiring the stakeholders to need prior in-depth knowledge of the system domain.
- *A software-based demonstrator combining the above modular structure analysis and modularity violation identification techniques to support decision making.* This demonstrator will facilitate decision making regarding whether to re-design or re-structure certain modules of a cyber-physical system.

8.7 RISKS AND PAYOFFS

There are two known risks to the execution of the proposed research:

1. *If there is a lack of high quality data, it will be impossible to develop the decomposition approaches or train the domain concept learner.* The intent of the incubator phase was to mitigate this risk. In the incubator, we have identified and evaluated data sets for two open source cyber-physical systems and found that they provide useful data for this study. We plan to further leverage these data sets to perform more in-depth research. These projects are in the domains of IoT and healthcare, which emphasize efficiency and safety, two desirable qualities in cyber-physical systems for the DoD. As the project proceeds we will continue to seek out additional data sets to exercise the resulting methods more broadly.
2. *If the domain concept learner fails to isolate the key terms from a domain, the detected modularity violations will be unreliable.* To evaluate and tune the domain concept learner, we will rely on assistance from subject matter experts. For MD PnP, the Co-PI of this project, Dr. Michael Pennock has prior experience modeling medical systems and has access to experts in the medical field. For OpenWRT, there are faculty members at Stevens that have led research projects in the IoT domain. They will be qualified to assess the quality of the domain concept learner outputs for OpenWRT. Finally, as MD PnP and OpenWRT are both open source projects, we will be able to reach out to the developers and project managers of the projects for their feedback.

If these risks can be overcome, the payoff will be a semi-automated tool that would enable DoD and other Government program managers to evaluate conformance to modularity requirements for procured systems, and consequently, reduce the risk of vendor lock-in.

8.8 BUDGET

The current estimated budget for Phase 2 is \$250K per year for two years. The primary cost is expected to be labor consisting of release time for the PI and Co-PI, student assistants to conduct the empirical aspects of the research, and research assistant/scientist support to develop the software demonstrator.

8.9 TIMELINE

Phase 1: the incubator feasibility study has been completed. In Phase 2, we plan to spend one quarter on each of the four modular decomposition approaches listed in Section 0. Once the decomposition approaches are finalized, we expect the development of the domain concept learner will take approximately 3 quarters. In parallel, we will begin development of the decision framework demonstrator starting from the 4th quarter of the second phase and lasting until the end of the project.

Research Phase	Task Description	17	18				19			
		Q3	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
1	Feasibility Incubation Case Study									

2	Atomic Level Modular Structure Analysis	■								
	Modular Analysis based on Coupling		■							
	Modular Analysis based on Cohesion			■						
	Modular Analysis at System Architecture Level				■					
	Domain Concept Learner					■	■	■		
	Build Decision Framework and Demonstrator					■	■	■	■	■

8.10 PROJECT EVALUATION AND MEASUREMENT

In order to evaluate the utility of the approach, we will apply the software demonstrator to a test cyber-physical system development project. By design, this test project will not be used to develop the methods or the software demonstrator. In other words, it will not be part of the “training” data set. We plan to identify a suitable project over the course of the research that would be sufficiently different from the training set to serve as a realistic test. Alternatively, we could work with the customer to identify a project of immediate interest that could be evaluated assuming that it does not require extensive engineering level modifications to the demonstrator due to differences in programming language, version control systems, etc. They will then be able to compare the results produced by the demonstrator to actual data or subject matter expert knowledge to make an independent assessment of the utility of the approach.

8.11 REFERENCES

- Baldwin, C. Y., & Clark, K. B. (2000). *Design rules: The power of modularity* (Vol. 1). MIT press.
- Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software*, 12(6), 42-50.
- González-Nalda, P., Etxeberria-Agiriano, I., & Calvo, I. (2015, June). Flexible, modular, standard, free and affordable model for CPS control applied to mobile robotics. In *Information Systems and Technologies (CISTI), 2015 10th Iberian Conference on* (pp. 1-6). IEEE.
- MacCormack, A., Rusnak, J., & Baldwin, C. Y. (2006). Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, 52(7), 1015-1030.
- Rajhans, A., Cheng, S. W., Schmerl, B., Garlan, D., Krogh, B. H., Agbi, C., & Bhave, A. (2009). An architectural approach to the design and analysis of cyber-physical systems. *Electronic Communications of the EASST*, 21.
- Sinha, K. (2014). *Structural complexity and its implications for design of cyber-physical systems* (Doctoral dissertation, Massachusetts Institute of Technology).
- Baheti, R., & Gill, H. (2011). Cyber-physical systems. *The impact of control technology*, 12, 161-166.

- Jantunen, E., Zurutuza, U., Ferreira, L. L., & Varga, P. (2016, April). Optimising maintenance: What are the expectations for cyber physical systems. In *Emerging Ideas and Trends in Engineering of Cyber-Physical Systems (EITEC), 2016 3rd International Workshop on* (pp. 53-58). IEEE.
- Derler, P., Lee, E. A., & Vincentelli, A. S. (2012). Modeling cyber-physical systems. *Proceedings of the IEEE, 100*(1), 13-28.
- Cristalli, C., Boria, S., Massa, D., Lattanzi, L., & Concettoni, E. (2016, October). A Cyber-Physical System approach for the design of a modular Smart Robotic Cell. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE* (pp. 4845-4850). IEEE.
- Lee, E. A. (2015). The past, present and future of cyber-physical systems: A focus on models. *Sensors, 15*(3), 4837-4869.
- Lee, E. A. (2008, May). Cyber physical systems: Design challenges. In *Object oriented real-time distributed computing (isorc), 2008 11th ieee international symposium on* (pp. 363-369). IEEE.
- Rajkumar, R. R., Lee, I., Sha, L., & Stankovic, J. (2010, June). Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference* (pp. 731-736). ACM.
- Wong, S., Cai, Y., Kim, M., & Dalton, M. (2011, May). Detecting software modularity violations. In *Proceedings of the 33rd International Conference on Software Engineering* (pp. 411-420). ACM.
- Schwanke, R., Xiao, L., & Cai, Y. (2013, May). Measuring architecture quality by structure plus history analysis. In *Proceedings of the 2013 International Conference on Software Engineering* (pp. 891-900). IEEE Press.
- Xiao, L., Cai, Y., & Kazman, R. (2014, November). Titan: A toolset that connects software architecture with quality analysis. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 763-766). ACM.
- Mo, R., Cai, Y., Kazman, R., & Xiao, L. (2015, May). Hotspot patterns: The formal definition and automatic detection of architecture smells. In *Software Architecture (WICSA), 2015 12th Working IEEE/IFIP Conference on* (pp. 51-60). IEEE.
- Malavolta, I., Muccini, H., & Sharaf, M. (2015, September). A preliminary study on architecting cyber-physical systems. In *Proceedings of the 2015 European Conference on Software Architecture Workshops* (p. 20). ACM.
- Xiao, L., Cai, Y., Kazman, R., Mo, R., & Feng, Q. (2016, May). Identifying and quantifying architectural debt. In *Proceedings of the 38th International Conference on Software Engineering* (pp. 488-498). ACM.
- Mo, R., Cai, Y., Kazman, R., Xiao, L., & Feng, Q. (2016, May). Decoupling level: a new metric for architectural maintenance complexity. In *Proceedings of the 38th International Conference on Software Engineering* (pp. 499-510). ACM.