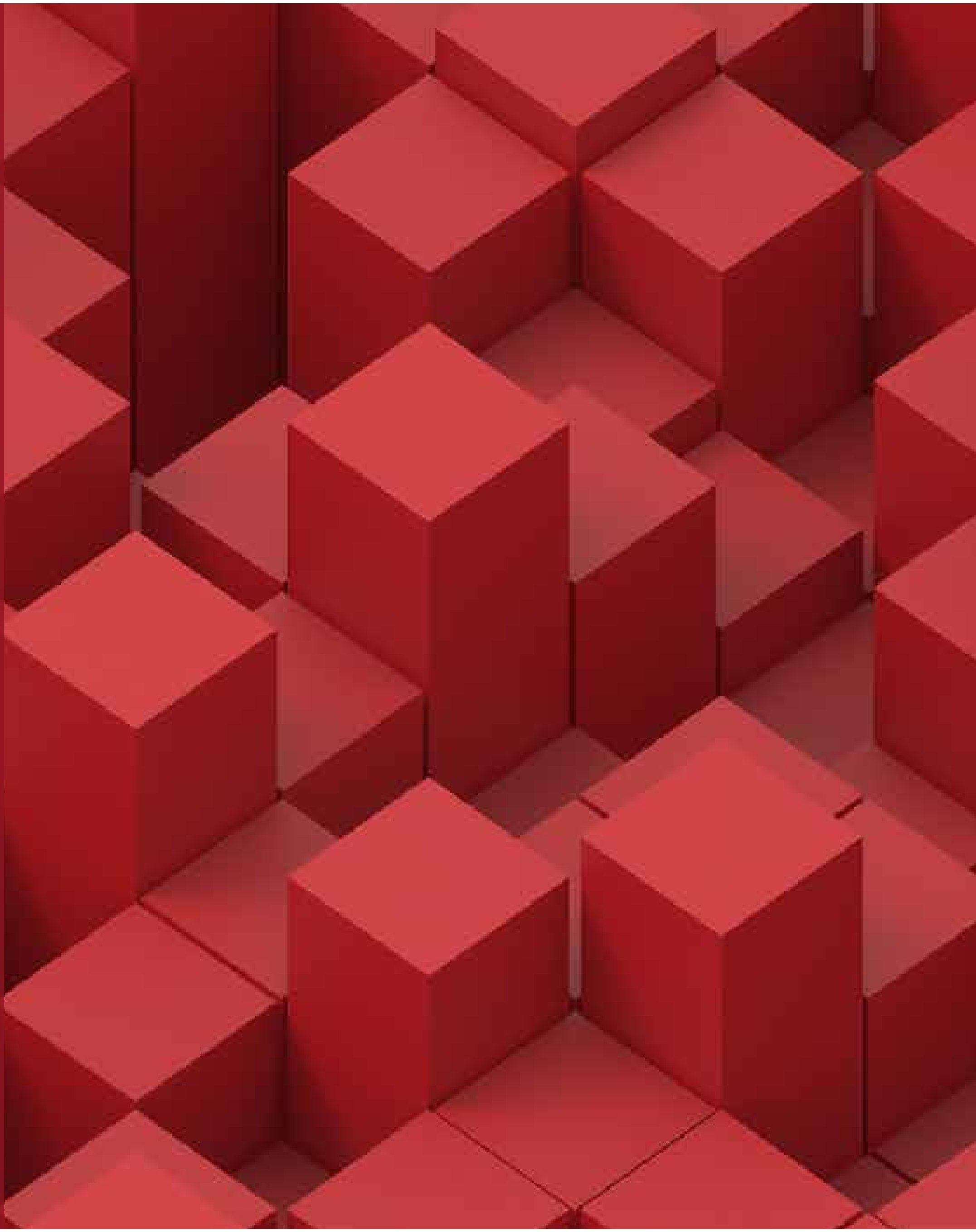
Carnegie Mellon University
Software Engineering Institute

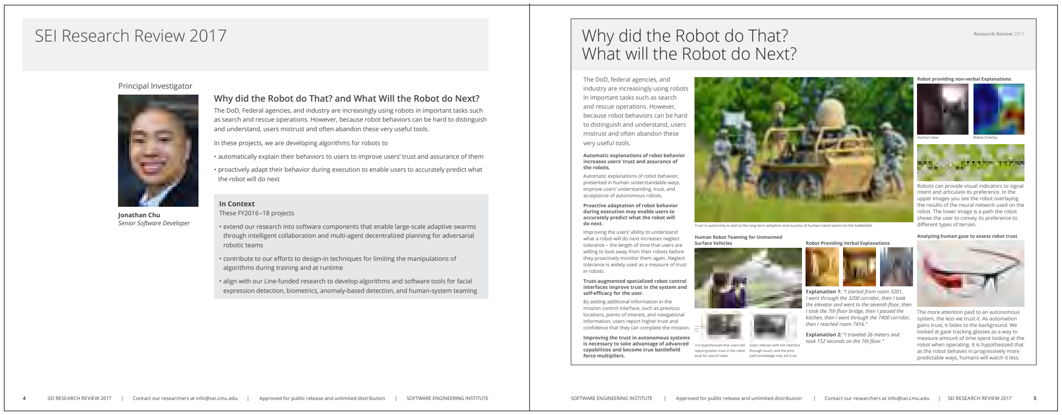
Research Review 2017

OCTOBER 17–18, 2017



Project Summaries and Posters

Approved for public release and unlimited distribution



SEI Research Review 2017

Principal Investigator

Jonathan Chu
Senior Software Developer

Why did the Robot do That? and What Will the Robot do Next?

The DoD, Federal agencies, and industry are increasingly using robots in important tasks such as search and rescue operations. However, because robot behaviors can be hard to distinguish and understand, users mistrust and often abandon these very useful tools.

In these projects, we are developing algorithms for robots to

- automatically explain their behaviors to users to improve users' trust and assurance of them
- proactively adapt their behavior during execution to enable users to accurately predict what the robot will do next

In Context
These FY2016–18 projects

- extend our research into software components that enable large-scale adaptive swarms through intelligent collaboration and multi-agent decentralized planning for adversarial robotic teams
- contribute to our efforts to design-in techniques for limiting the manipulations of algorithms during training and at runtime
- align with our Line-funded research to develop algorithms and software tools for facial expression detection, biometrics, anomaly-based detection, and human-system teaming

4 SEI RESEARCH REVIEW 2017 | Contact our researchers at info@sei.cmu.edu | Approved for public release and unlimited distribution | SOFTWARE ENGINEERING INSTITUTE

Why did the Robot do That?
What will the Robot do Next?

Research Review 2017

The DoD, federal agencies, and industry are increasingly using robots in important tasks such as search and rescue operations. However, because robot behaviors can be hard to distinguish and understand, users mistrust and often abandon these very useful tools.

Automatic explanations of robot behavior increases users' trust and assurance of the robots.

Automatic explanations of robot behavior, presented in human understandable ways, improve users' understanding, trust, and acceptance of autonomous robots.

Proactive adaptation of robot behavior during execution may enable users to accurately predict what the robot will do next.

Improving the users' ability to understand what a robot will do next increases neglect tolerance – the length of time that users are willing to look away from their robots before they proactively monitor them again. Neglect tolerance is widely used as a measure of trust in robots.

Trust-augmented specialized robot control interfaces improve trust in the system and self-efficacy for the user.

By adding additional information in the mission control interface, such as previous locations, points of interest, and navigational information, users report higher trust and confidence that they can complete the mission.

Improving the trust in autonomous systems is necessary to take advantage of advanced capabilities and become true battlefield force multipliers.

Trust in autonomy is vital to the long term adoption and success of human-robot teams on the battlefield.

Human Robot Teaming for Unmanned Surface Vehicles

It is hypothesized that users will report greater trust in the robot boat for search tasks.

Users interact with the interface through touch, and the prior path knowledge may aid trust.

Robot Providing Verbal Explanations

Explanation 1: *"I started from room 3201, I went through the 3200 corridor, then I took the elevator and went to the seventh floor, then I took the 7th floor bridge, then I passed the kitchen, then I went through the 7400 corridor, then I reached room 7416."*

Explanation 2: *"I traveled 26 meters and took 152 seconds on the 7th floor."*

Robot providing non-verbal Explanations

Robots can provide visual indicators to signal intent and articulate its preference. In the upper images you see the robot overlaying the results of the neural network used on the robot. The lower image is a path the robot shows the user to convey its preference to different types of terrain.

Analyzing human gaze to assess robot trust

The more attention paid to an autonomous system, the less we trust it. As automation gains trust, it fades to the background. We looked at gaze tracking glasses as a way to measure amount of time spent looking at the robot when operating. It is hypothesized that as the robot behaves in progressively more predictable ways, humans will watch it less.

SOFTWARE ENGINEERING INSTITUTE | Approved for public release and unlimited distribution | Contact our researchers at info@sei.cmu.edu | SEI RESEARCH REVIEW 2017 5

This booklet contains descriptions of SEI research projects and images of posters related to the research. In each of the sections, you will find a project description on a left-hand (even-numbered) page and a poster image facing it on a (odd-numbered) right-hand page.

Dr. Jeffrey Boleng
CTO (Acting)
Deputy CTO

Ensuring the Rapid Development of Effective, Reliable, Safe, and Secure Systems

The intensely competitive technology environment that supports U.S. military capabilities requires the U.S. Department of Defense (DoD) to continually and rapidly develop and deploy innovative, software-enabled components and systems that are affordable, resilient, and easy to modify. At the Carnegie Mellon University Software Engineering Institute (CMU SEI), we value our applied research and development (AR&D) in software and cybersecurity according to its impact on those software system qualities and behaviors.

We focus our AR&D on data-driven, formally verified, and automated development support across the system lifecycle. We produce algorithms, tools, techniques, and practices that address critical technology gaps. Further, we deliver capabilities through an execution model that combines our AR&D with customer engagement and our deep expertise in forming, applying, and transitioning solutions that improve the state of the practice in software engineering and cybersecurity.

In addition, as a federally funded research and development center (FFRDC), CMU SEI serves as a trusted and value-added broker of R&D by working with members of the software community in government, academia (in particular, CMU), and industry to customize, develop, and adapt software and cybersecurity technologies. With the access afforded by our DoD affiliation and a conflict-free status as an FFRDC, CMU SEI has a unique ability to undertake technical work—Line-funded research and sponsored engagements—ranging from fundamental research with widespread publication to support of sensitive government programs.

We invite you to explore the details in this report of our 2017 DoD Line-funded research portfolio.

Contents

Autonomy: Trustworthiness and Machine Perception

Principal Investigator

Jonathan Chu
Senior Software Developer

Why did the Robot do That? and What Will the Robot do Next?

The DoD, Federal agencies, and industry are increasingly using robots in important tasks such as search and rescue operations. However, because robot behaviors can be hard to distinguish and understand, users mistrust and often abandon these very useful tools.

In these projects, we are developing algorithms for robots to

- automatically explain their behaviors to users to improve users' trust and assurance of them
- proactively adapt their behavior during execution to enable users to accurately predict what the robot will do next

In Context

These FY2016–18 projects

- extend our research into software components that enable large-scale adaptive swarms through intelligent collaboration and multi-agent decentralized planning for adversarial robotic teams
- contribute to our efforts to design-in techniques for limiting the manipulations of algorithms during training and at runtime
- align with our Line-funded research to develop algorithms and software tools for facial expression detection, biometrics, anomaly-based detection, and human-system teaming

Why did the Robot do That? What will the Robot do Next?

The DoD, federal agencies, and industry are increasingly using robots in important tasks such as search and rescue operations. However, because robot behaviors can be hard to distinguish and understand, users mistrust and often abandon these very useful tools.

Automatic explanations of robot behavior increases users' trust and assurance of the robots.

Automatic explanations of robot behavior, presented in human understandable ways, improve users' understanding, trust, and acceptance of autonomous robots.

Proactive adaptation of robot behavior during execution may enable users to accurately predict what the robot will do next.

Improving the users' ability to understand what a robot will do next increases neglect tolerance – the length of time that users are willing to look away from their robots before they proactively monitor them again. Neglect tolerance is widely used as a measure of trust in robots.

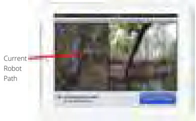
Trust-augmented specialized robot control interfaces improve trust in the system and self-efficacy for the user.

By adding additional information in the mission control interface, such as previous locations, points of interest, and navigational information, users report higher trust and confidence that they can complete the mission.

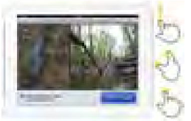
Improving the trust in autonomous systems is necessary to take advantage of advanced capabilities and become true battlefield force multipliers.

Trust in autonomy is vital to the long term adoption and success of human-robot teams on the battlefield.

Human Robot Teaming for Unmanned Surface Vehicles

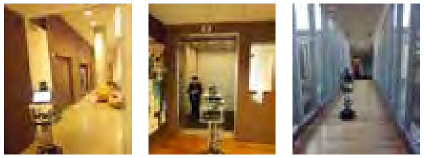


It is hypothesized that users will report greater trust in the robot boat for search tasks.



Users interact with the interface through touch, and the prior path knowledge may aid trust.

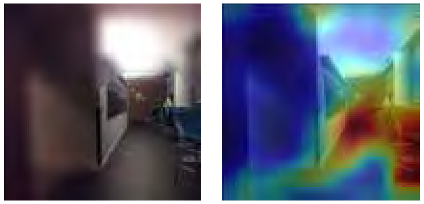
Robot Providing Verbal Explanations



Explanation 1: *"I started from room 3201, I went through the 3200 corridor, then I took the elevator and went to the seventh floor, then I took the 7th floor bridge, then I passed the kitchen, then I went through the 7400 corridor, then I reached room 7416."*

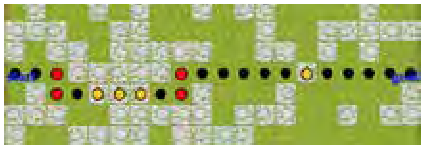
Explanation 2: *"I traveled 26 meters and took 152 seconds on the 7th floor."*

Robot providing non-verbal Explanations



Human view

Robot Overlay



Robots can provide visual indicators to signal intent and articulate its preference. In the upper images you see the robot overlaying the results of the neural network used on the robot. The lower image is a path the robot shows the user to convey its preference to different types of terrain.

Analyzing human gaze to assess robot trust

The more attention paid to an autonomous system, the less we trust it. As automation gains trust, it fades to the background. We looked at gaze tracking glasses as a way to measure amount of time spent looking at the robot when operating. It is hypothesized that as the robot behaves in progressively more predictable ways, humans will watch it less.

Principal Investigators

Dr. Scott McMillan
Senior Research Scientist

Dr. Franz Franchetti
Professor, Department of Electrical and Computer Engineering, Carnegie Mellon University

Automated Code Generation for High-Performance Graph Libraries

Graph analytics and other data-intensive applications are characterized by unpredictable memory access and low computation intensity. Called irregular algorithms, these applications are difficult to implement efficiently, but they are required by increasingly complex systems.

The goal of this work is to automate code generation of high-performance libraries of graph algorithms, tuned for at least two different hardware architectures—multi-core CPU and SIMD (Single Instruction, Multiple Data) GPU—to establish a foundation for the infrastructure of a wider range of systems. This work advances the state-of-the-art in code generation by introducing concepts needed to capture graph algorithm primitives, including a parameterization of the structure of sparse data to produce highly tuned codes for irregular algorithms.

In Context

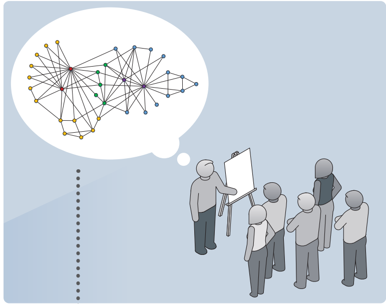
This FY2017 project

- builds on our prior Line-funded research in the in Patterns and Practices for Future Architectures and Graph Algorithms on Future Architectures
- contributes to an FY2018 Line-funded project in building a COTS benchmark baseline for graph analytics
- aligns with our other data-intensive projects such as the Predictive Analytics Workshop, through the use of graph algorithms as an enabling technology
- is related to technical work in the GraphBLAS task in the Big Learning Workloads project (see page 48)

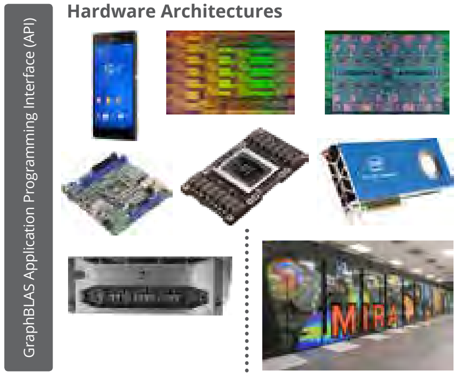
Automated Code Generation for High-Performance, Future-Compatible Graph Libraries

THE GRAPHBLAS APPLICATION PROGRAMMING INTERFACE (API) SPECIFICATION released this year allows graph analytics experts and hardware experts to more easily come together to develop high performance graph algorithms. In the coming year we will be developing automated code generation tools to help hardware experts tune the graph primitives defined in the API for each new hardware architecture that is developed.

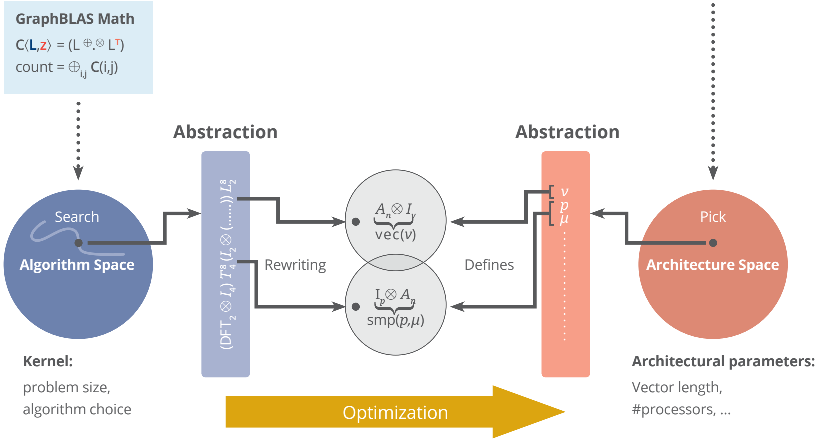
Algorithm Development



Hardware Architectures



By separating the concerns between algorithm development and hardware architectures, the GraphBLAS API (graphblas.org) frees graph experts to develop algorithms while allowing hardware experts to fine tune primitives for existing and new hardware architectures.



Achieving high-performance from today's complex hardware architectures traditionally requires large teams of developers with hardware expertise. This year, we are building on automated code generation technology called Spiral (spiral.net) to use the mathematical formalization of the GraphBLAS algorithms to automatically generate the high-performance code for targeted hardware platforms.

Goal: Write Once Run Everywhere

The GraphBLAS API Specification was released in May 2017. It defines a set of primitive operations that can be used to implement a wide variety of graph algorithms. It allows the separation of concerns where:

- Graph experts can more easily develop algorithms at a high level using the primitives defined by the API.
- Hardware experts can finely tune the primitives for their present and future hardware architectures

Goal: Automated Code Generation for High-Performance

It still takes a large team of developers with knowledge of the complex hardware architectures to finely tune the GraphBLAS primitives. With the rapid development of new hardware architectures, the burden on these teams to develop high-performance implementations for each architecture is high. Using formal specifications of hardware capabilities, CMU's Spiral code generation technology can already automatically generate high performance signal processing codes. We are currently augmenting Spiral to support the GraphBLAS primitives so it can generate high performance code for graph algorithms.

By combining the mathematical descriptions of the graph primitives specified by GraphBLAS and the formal specifications of hardware used by Spiral, our technology will be able to automatically generate high-performance graph library code for today's hardware platforms as well as future architectures still being designed.

Principal Investigator

Satya Venneti
Senior Member of the Technical Staff

Presenter

Oren Wright
Researcher

Micro-Expressions: More than Meets the Eye

The human face is a universal system of signals that reflect the moment-to-moment fluctuations in a person's emotional state. However, ordinary human facial expressions (macro-expressions) may not accurately portray genuine emotion. Micro-expressions are rapid, involuntary, and last for up to half a second. Hard to recognize due to their short duration, low intensity, and inherent inhibition, micro-expressions expose genuine emotions that people do not intend to show and are universal across cultures.

In this work, we are building an accurate, automatic micro-expression analysis prototype that outperforms humans in spotting and recognizing facial micro-expressions in near real time. The result of this work supports DoD forensics and intelligence mission capabilities. Recognizing micro-expressions is also a primary step for continuous emotion recognition from videos, which is a key ingredient for embedding machines with emotional intelligence and creating machines that can detect, understand, and respond to human emotions.

In Context

This FY2017 project

- builds on prior technical work to build algorithms that extract biometric traits (i.e., heart rate) in real-time from video of multiple non-stationary human subject faces
- contributes to our FY2018 Line-funded research into designing algorithms to recognize emotion from voice (important for speaker profiling for intelligence gathering)
- aligns with our Line-funded CMU SEI research to develop algorithms and software tools for robot explainability and predictability, anomaly-based detection, and human-system teaming

Micro-Expressions: More Than Meets the Eye

Using Software to Reveal True Emotions

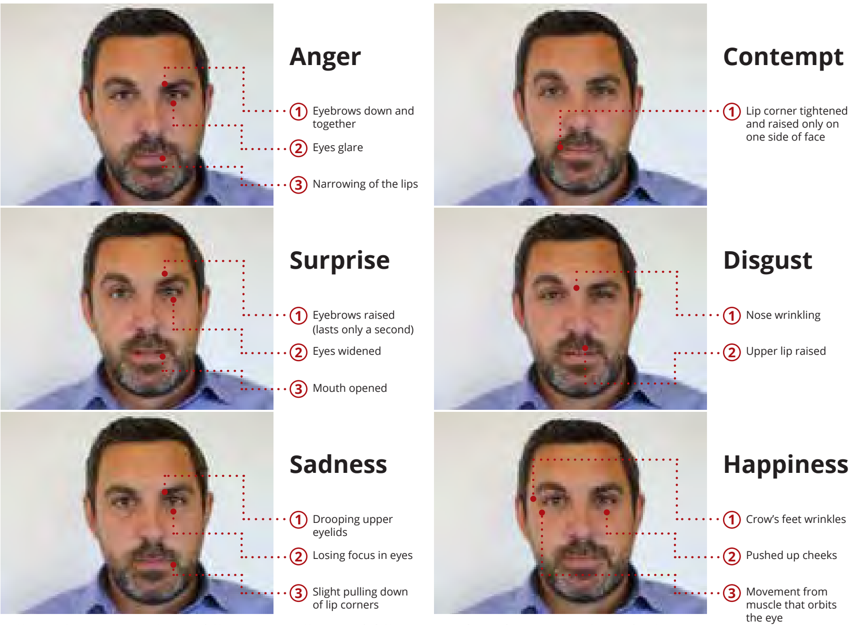
Micro-expressions—involuntary, fleeting facial movements that reveal true emotions—hold valuable information for scenarios ranging from security interviews and interrogations to media analysis. We are developing a prototype software tool to recognize micro-expressions, identifying the emotions they reveal.

Micro-expressions: tiny movements with a lot of information. Micro-expressions can occur on various regions of the face and last only a fraction of a second. These movements have been shown to be universal across cultures, and they are very difficult to suppress.

Defense and intelligence applications. Our work advances capabilities in human-machine teaming and machine emotional intelligence, and can be applied in a wide range of scenarios, including:

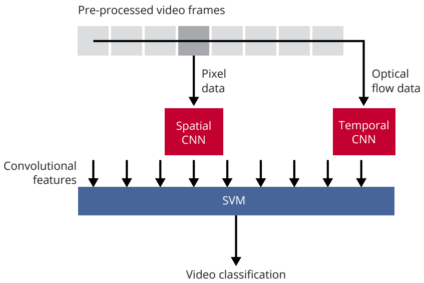
- security checkpoint encounters
- interrogations
- polygraph testing
- media analysis and exploitation
- detection of stress, PTSD

Current state of the art. Current tools for recognizing emotion (for example, Affectiva) can successfully identify emotions based on *macro-expressions* like broad smiles, exaggerated frowns, and obviously narrowed eyes and pursed lips. However, macro-expressions can be easily faked. Current approaches for recognizing *micro-expressions* use hand-crafted features and treat each video frame as a stand-alone image. This approach is brittle, is slow, and has limited accuracy.



Different features manifest at different time offsets and different speeds, making current frame-by-frame approaches to recognizing micro-expressions inaccurate.

An interesting test case for micro-expression recognition is poker tells: can our tool identify emotions highly skilled poker players intend to hide?



Our approach: We use machine learned features and incorporate optical flow to introduce temporal structure.

Results. We designed and built a micro-expression recognition system that improves upon the state of the art:

- We used machine-learned features that treat the whole face as a canvas, in contrast to traditional hand-crafted features and techniques that search pre-defined areas of the face for facial action units. Machine-learned features were generated with a pre-trained convolutional neural network.
- We combined optical flow data with frame-by-frame pixel information to better incorporate temporal structure into the recognition model.
- We used several pre-processing techniques, such as video interpolation via graph embedding, to improve accuracy or maintain accuracy while reducing runtime.

Next steps. A number of opportunities exist to extend our work in micro-expressions. Looking ahead, we are interested in combining micro-expression detection with recognition to increase the practicality of our solution, improving datasets to advance research in micro-expression recognition, and exploring solutions for long-running videos.

Future work includes recognizing emotion from voice.

Related work. Micro-expression recognition is part of a larger portfolio of SEI work in "machine emotional intelligence"—using physiological characteristics to enable machines to better understand humans. In 2016, we developed a tool to extract heart rate from video; our next project in this area is recognizing emotion from voice.

Taming Uncertainty in Software Cost Estimation

DoD acquisition regulations call for early (pre-Milestone A) estimates that stretch across the entire program lifecycle. These early cost estimates rely heavily on judgments about cost factors that may change over the lifecycle. The SEI Quantifying Uncertainty in Early Lifecycle Cost Estimation (QUELCE) method enables the calculation of cost impacts caused by changes.

Affordability: Lifecycle Costs

Principal Investigators

Robert Stoddard
Principal Researcher

Dr. Michael Konrad
Principal Researcher

Why Does Software Cost so Much? Towards a Causal Model

The DoD needs to identify factors causing high software costs in order to enact new software policy and enable more informed negotiated pricing of contracted software. In this work, we use new data mining techniques to evaluate datasets representing about 60 unique cost factors and more than 15 cost relationships. From that evaluation, we are building an actionable, full causal model of software cost factors that is immediately useful to DoD programs and contract negotiators.

In Context

This FY2017 project

- builds on our Line-funded research and customer engagements that formed and applied the Quantifying Uncertainty in Early Lifecycle Cost Estimation method
- extends technical work with the U.S. Air Force (should-cost analysis), the U.S. Navy (sustainment modeling), ongoing engagement with CMU faculty in causal learning, and collaboration with the University of Southern California Center for Systems and Software Engineering
- contributes to our FY2018 Line-funded research into using an integrated causal model for software cost prediction and control
- aligns with research projects exploring the applicability of machine learning, causal modeling, and systems thinking to technical debt and development lifecycle phases, such as testing and sustainment

Why does software cost so much?

Towards a Causal Model

How can we better control costs in software development and sustainment? This LENS is collaborating with the larger community of cost estimators in applying causal learning to program datasets to better understand which factors should be adjusted to reduce software lifecycle costs. The resulting causal model will be incorporated into a program dashboard to enable more effective program control.

Problem

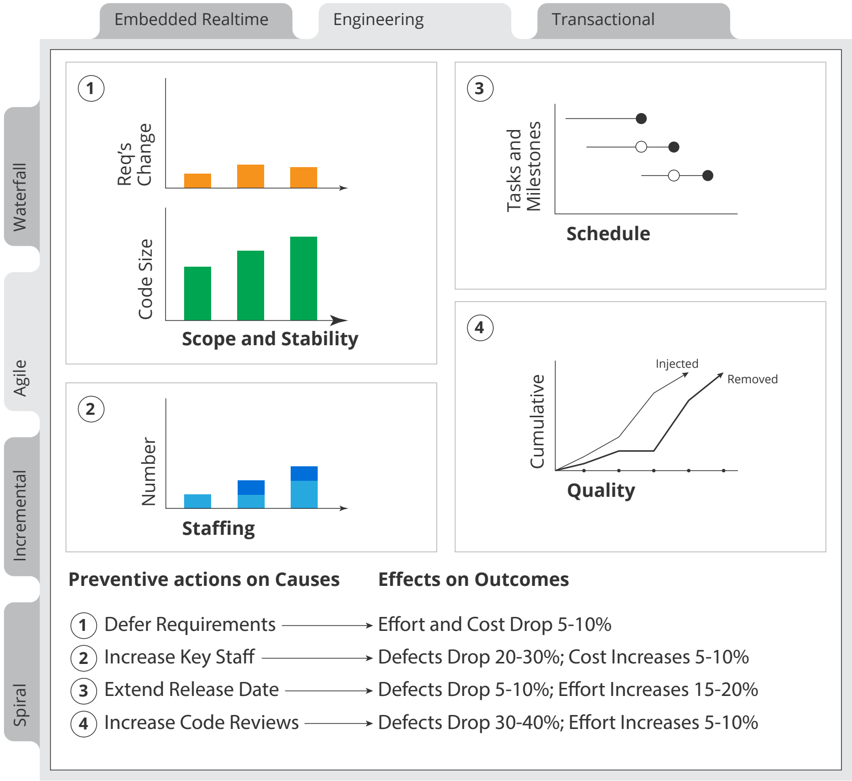
- DoD leadership continues to ask “Why does software cost so much?”
- DoD program offices need to know where to intervene to control software costs

Solution

An actionable, full causal model of software cost factors immediately useful to DoD programs and contract negotiators

Actionable intelligence

- Enhance program control of software cost throughout the development and sustainment lifecycles
- Inform “could/should cost” analysis and price negotiations
- Improve contract incentives for software-intensive programs
- Increase competition using effective criteria related to software cost



Preventive actions on Causes	Effects on Outcomes
1 Defer Requirements →	Effort and Cost Drop 5-10%
2 Increase Key Staff →	Defects Drop 20-30%; Cost Increases 5-10%
3 Extend Release Date →	Defects Drop 5-10%; Effort Increases 15-20%
4 Increase Code Reviews →	Defects Drop 30-40%; Effort Increases 5-10%

Program Dashboard extended with key relationships from causal modeling

Who would use the tool?

A program’s key stakeholders, as part of their regular progress reviews, will examine progress to date and evaluate proposed corrective actions, according to their respective roles:

Contract Manager
Implications for contract commitments?

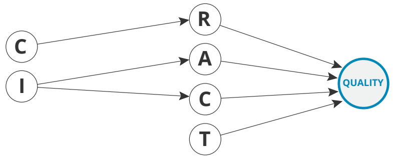
Acquisition Manager
Implications for end user and dependencies on/from other programs?

Program Manager
Change priorities for product content and reset expectations?

Chief Engineer
What changes to plans, resources, and tools are needed to implement action?

Technical Approach

Working with collaborators, we will identify and prepare datasets for causal learning to establish key cause-effect relationships among project factors and outcomes, segmented by software lifecycle and software super-domain. For example, for Quality, we might have this causal graph:



The resulting causal models will then be “stitched” using CMU algorithms to create a universal causal model, but estimated and calibrated for lifecycle and super-domain. These estimated models will be the basis for improved program management. (See figure at left.)

Collaborative Approach

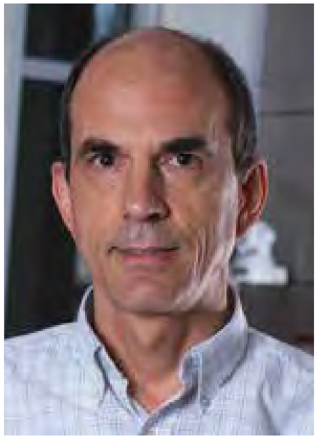
First, we train each collaborator so that they are capable of running causal analyses on their own proprietary datasets. Then, they derive causal models and only share with the SEI general information about the dataset and search and the resulting causal graph, sufficient for integrating into a universal model.

Summary

Causal learning has come of age from both a theoretical and tooling standpoint and provides better basis for program control than models based on correlation. Application to cost estimation requires large amounts of quality data. Now is the time to engage the larger community of cost estimators in deriving improved cost models that enable improved program control.

Principal Investigators

Dr. Ipek Ozkaya
Principal Researcher



Dr. Robert Nord
Principal Researcher

Technical Debt Analysis through Software Analytics

DoD and other government acquisition managers need capabilities to assess what kind of technical debt their developers and software contractors are creating through their decisions. Current solutions rely primarily on code analysis and fail to differentiate among design issues that lead to accumulating rework costs. The challenge at hand today is to provide the development teams and the government with software analytics capabilities that allow them to analyze the quality of the software and consequences of the design choices made continuously.

In this work, we are developing tools that integrate data from multiple, commonly available sources to pinpoint problematic design decisions and quantify their consequences in a repeatable and reliable way for uncovering technical debt. Improving identification of such issues and quantifying effect on accumulating rework provides data to help DoD control lifecycle costs, mitigate technical risk, and reduce cycle times.

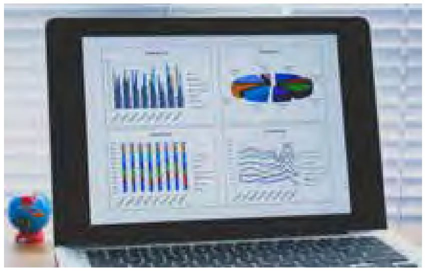
In Context

This FY2017–18 project

- extends prior Line-funded research on sustainability and technical debt: Guiding System Sustainment through Quantifiable Technical Debt Management and Finding Software Vulnerabilities Early by Correlating with Technical Debt
- aligns with Line-funded research and customer engagements that formed and applied the Quantifying Uncertainty in Early Lifecycle Cost Estimation method and research projects exploring the applicability of machine learning, causal modeling, and systems thinking to software cost estimation and development lifecycle phases such as testing and sustainment

Technical Debt Analysis through Software Analytics

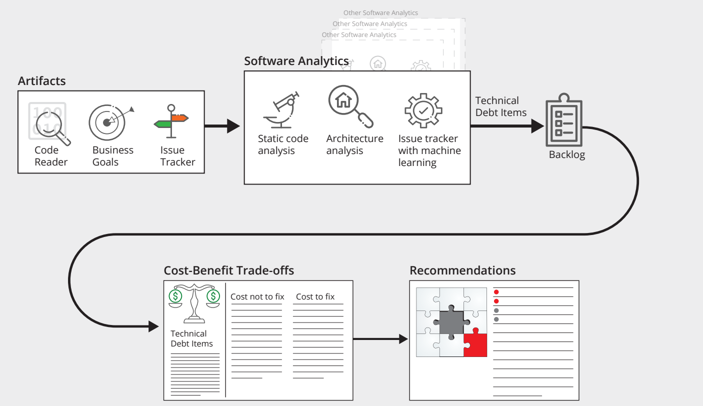
Technical debt conceptualizes the tradeoff between the short-term benefits of rapid delivery and the long-term value of developing a software system that is easy to evolve, modify, repair, and sustain. In this work, we are developing tools that integrate data from multiple, commonly available sources to pinpoint problematic design decisions and quantify their consequences in a repeatable and reliable way for uncovering technical debt.



Our research approach includes:

- Automatically extract potential TD issues from issue trackers using data mining techniques
- Enrich these clusters by incorporating information from code analyses and software repositories.
- Rank each TD cluster in terms of accumulating rework to date (e.g., total change and bug churn, number of related issues).

Technical Debt Analysis Workflow



Categorization and classification:

- Created a classifier that utilizes n-gram feature engineering and gradient boosting
- Our data shows that developers do identify long-lasting design issues as technical debt
- We are in the process of improving the classifier to help identify technical debt.

Model	Precision	Recall	F-Measure	Test Count	Training Count
1	0.60	0.42	0.50	510	157
2	0.71	0.32	0.44	411	256
3	0.77	0.42	0.54	311	356
4	0.74	0.70	0.72	161	506

> **Results:** Our models are improving, especially in recall (more true debt is identified).

TD clusters

Design problems, frequently the result of optimizing for delivery speed, are a critical part of long-term software costs. Detecting such design issues with tool support is a high priority for technical debt analysis. We developed an approach where existing static analysis rules are mapped to design issues. The goal of the analysis is to help teams focus on rework causing issues earlier in the lifecycle.

		P1	P2	P5	P6	P7	All
	Rule	#	#	#	#	#	#
Logging	Standard outputs should not be used directly to log anything	334			7		343
Data consistency	@Override annotation should be used on any method overriding (since Java 5) or implementing (since Java 6) another one	1	172	1285	340	11	1799
Security & performance	Exception handlers should preserve the original exceptions	3	7	235	161	922	451
Maintainability	Source files should not have any duplicated blocks	133	36	16	23		260
	Methods should not be empty		65	463	74	67	615
	Control flow statements “if”, for, while, switch and try should not be nested too deeply	794	53	8	10	2	911
	Unused local variables should be removed	1	91	189	48	106	401
	Dead stores should be removed	5	491	512	372	291	1437
	Methods should not be too complex		79	208	157	1	498
	Sections of code should not be “commented out”		91	2	167		453
	Unused “private” fields should be removed		68	42	27	12	154
	Collapsible “if” statements should be merged	159	11	1	3		178

> **Results:** Analysis across 8 projects reveals that our approach highlights hidden issues.

Government acquisition managers need capabilities to assess what kind of technical debt is created throughout the software lifecycle. The SEI team has been a pioneer in advancing the research agenda in this regard. Our ongoing work is focused on creating analytical tools towards an integrated software analytics approach.

Principal Investigator

William Novak
Senior Member of the Technical Staff, Client Technical Solutions

A Game-Theoretic Approach to Optimizing Behaviors in Acquisition

The acquisition community is increasingly shifting to use the “Government as the Integrator” approach that depends on unrelated contractors working together closely, even when there may be little incentive for them to cooperate with companies that are potential competitors.

This misalignment of incentives both between contractors, and with the government, is a primary cause of poor acquisition outcomes, including wasted effort, lost time, and suboptimal results. To address this, our research characterizes and computationally quantifies the equilibrium structure of the acquisition “game” (i.e., the expected outcomes, given various utilities) using agent-based modelling and simulation to frame the misaligned incentive situation. To validate our research results, we will calibrate our model through interviews with program stakeholders and plan to pilot a set of incentive mechanisms that our simulation analyses have shown will improve outcomes through greater contractor cooperation.

In Context

This FY2017 project

- builds on CMU SEI research in using modeling and simulation techniques to develop innovative approaches to improve DoD software acquisition outcomes, such as the Quantifying Uncertainty in Early Lifecycle Cost Estimation method
- relies on the findings of several CMU SEI independent assessments of acquisition programs and research using system dynamics models to identify archetypes of common problems across programs

Getting Contractors to Cooperate

A Game-Theoretic Approach to Optimizing Acquisition Behaviors

Misaligned incentives between contractors and the PMO produce ineffective cooperation, causing wasted effort, lost time, and poor results. Using game theory to frame these situations, and agent-based modelling to quantify them, we can design incentive mechanisms to promote cooperation and improve results.

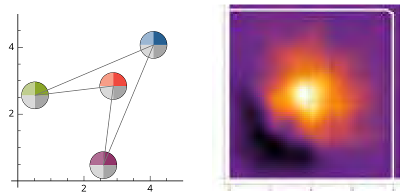


Figure 1: Animating effects of varying moral hazard incentives shows how to drive behaviors toward goals.

Theory: Principals, Agents, & Moral Hazard

Acquisition is a "Principal-Agent" problem where an expert "agent" (i.e., contractor) works for the "principal" (i.e., government PMO). Since the agent has better information than the principal, and their interests conflict, the exchange often goes awry. The government can't always verify contractor claims, so contractors may be deceptive, and quality vendors may not get the price they deserve. A contractor may even take on excessive risk because the government bears the costs—that's called "moral hazard" *(see Figure 1).*

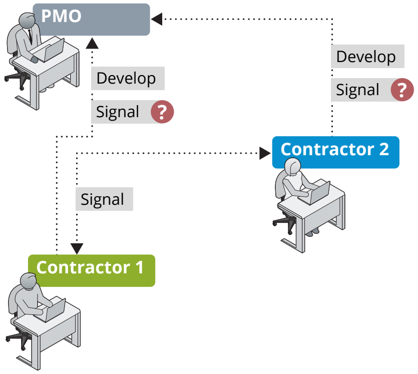


Figure 2: Contractors' incentives may make them uncooperative, while claiming to help the program.

Poor Contractor Cooperation with GATI

Despite "Government As The Integrator" (GATI) becoming more widespread, PMOs are still learning how to do GATI. Without a Lead Systems Integrator (LSI) leading a consortium, independent contractors have little incentive to share data and support other contractors, causing delays, overruns, and poor performance—all while denying such actions to the PMO to avoid penalty *(see Figure 2).*

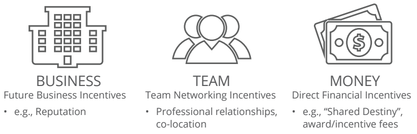

BUSINESS
Future Business Incentives
• e.g., Reputation
TEAM
Team Networking Incentives
• Professional relationships, co-location
MONEY
Direct Financial Incentives
• e.g., "Shared Destiny", award/incentive fees

Figure 3: Three types of incentive mechanisms can appeal to three levels of the contractor organization.

Incentive Mechanism Solutions

Custom incentive mechanisms can align contractor incentives, so contractors serving their interests also serve program interests. Multiple incentives used together maximizes results, reaching multiple organizational levels *(see Figure 3).*

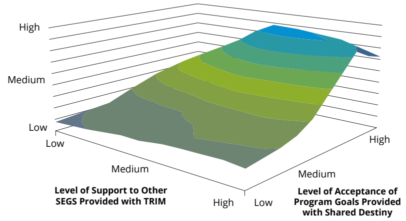


Figure 4: The effect of pair-wise combinations of incentive types on program performance

Modelling & Simulating Program Behavior

By analyzing GATI contractor incentives using game theory, we can characterize the likely contractor and PMO moves and counter-moves. Modelling can then quantify the outcomes of the game, to see which incentives best promote cooperation *(see Figure 4).*

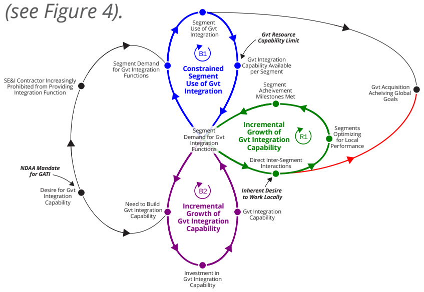

Segment Use of Gvt Integration
Gvt Resource Capability Limit
B1
Constrained Segment Use of Gvt Integration
Segment Demand for Gvt Integration Functions
Gvt Integration Capability Available per Segment
SE&I Contractor Increasingly Prohibited from Providing Integration Function
Segment Acheivement Milestones Met
Gvt Acquisition Acheiving Global Goals
Incremental Growth of Gvt Integration Capability
R1
Segments Optimizing for Local Performance
Segment Demand for Gvt Integration Functions
Direct Inter-Segment Interactions
NDAA Mandate for GATI
Desire for Gvt Integration Capability
B2
Inherent Desire to Work Locally
Need to Build Gvt Integration Capability
Incremental Growth of Gvt Integration Capability
Gvt Integration Capability
Investment in Gvt Integration Capability

Figure 5: Growing demand for government integration is frustrated by limited resources, driving segment-level integration to meet local goals, undermining program success.

A Real-World Program with GATI Issues

As one program moved from LSI to GATI, the PMO wasn't ready to do system integration (SI). The SI contractor lost its duties, but the government was slow to grow its own SI. The segments became impatient with the poor PMO SI support and turned to their fellow segments to do the SI work. Without central SI the segments made tactical decisions that didn't consider the global good of the program *(see Figure 5).*

Empirical Validation & Piloting Incentives

Remaining FY17 Work: Interview USAF program staff to gather empirical data.

Future Work: Pilot the most promising mechanisms in a real-world acquisition program and measure the results.

Preliminary High-Level Research Results

Incentive fees (e.g., TRIM) focus contractors program work, but a mix of types of incentives is needed to be effective across different kinds of contractor organizations.

Incentivizing contractors to meet cost/schedule goals can subvert other incentives promoting cooperation, thus sacrificing program goals.

Incentives are "weapons" in an ongoing "war" that must evolve and be replaced. Evolutionary Game Theory models competition in a population of acquisition programs to evolve and adapt incentives over time to keep them effective.

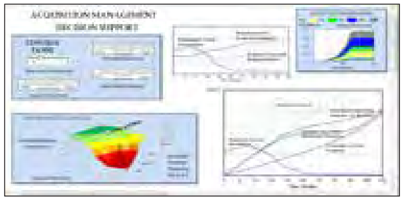


Figure 6: A notional concept of what a modeling/simulation workbench could look like

Future Engagement Model for Programs

This approach can solve many incentive problems that plague acquisition performance. We envision a virtual Acquisition Modelling Laboratory (vAML) service, based on game-theory and modeling/simulation, that helps DoD acquisition programs mitigate such problems and improve program outcomes *(see Figure 6).*

Saving Malware Analysts Time

The SEI is developing an automated tool to dramatically reduce the time human analysts need to gather data for malware comparisons. The researchers are extending existing SEI automated analysis capability built in the ROSE open source compiler infrastructure to provide the required data. The ROSE infrastructure was developed at Lawrence Livermore National Laboratory.

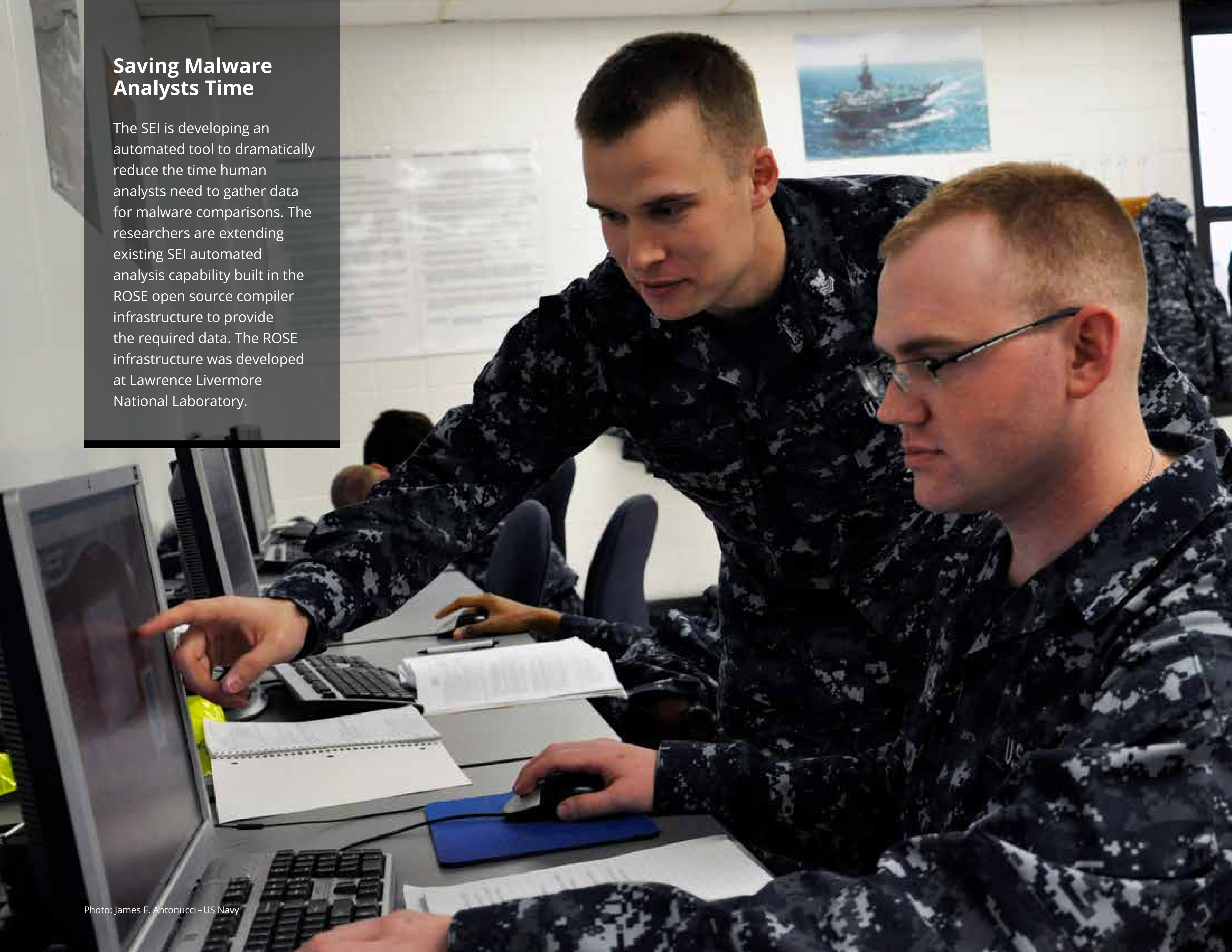


Photo: James F. Antonucci – US Navy

Data Analytics for Decisions

Principal Investigator

Dr. Lori Flynn
Software Security Researcher

Rapid Expansion of Classification Models to Prioritize Static Analysis Alerts for C

As automated static analysis tools identify more kinds of code flaws, the number of reported flaws (alerts) is increasing. Validation and repair of flaws discovered by static analysis requires manual effort from auditors and coders, a limited resource in every organization.

In this work, we created a method to automatically classify and prioritize alerts that minimizes manual effort to address the large volume of alerts.

In Context

This FY2017 project

- extends our prior work in Prioritizing Alerts from Static Analysis with Classification Models
- contributes to our FY2018 Line-funded research project to develop a prototype accurate alert handling system

Rapid Expansion of Classification Models

For prioritizing static analysis alerts for C

Problem: Security-related code flaws detected by static analysis require too much manual effort to triage; plus **it takes too long to audit enough alerts to develop classifiers to automate the triage.**

Solution: Rapid expansion of number of classification models by using "pre-audited" code, plus collaborator-audited code.

Approach:

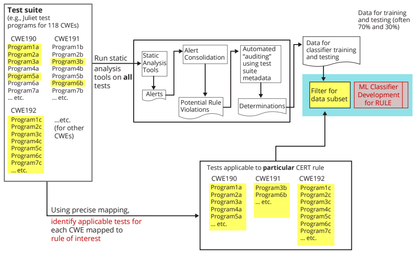
1. Modify SCALe research tool to map alerts to CWE
2. Systematically map CERT rules to named flaws in subsets of pre-audited code (published as true or false for flaw Automated analysis of pre-audited (not by SEI) codebases to gather sufficient code & alert feature info for classifiers)
3. Test classifiers on alerts from real-world code: DoD data

Process:

1. Generate data for Juliet: Proprietary and open-source static analysis tools and metrics tools
2. Generate data for STONESOUP: similar/same tools
3. Generate scripts for classifier development
4. Build classifiers: directly for CWEs, partitioned test suite for CERT rules
5. Test classifiers

Using CWE Test Suites for CERT Rule Classifiers

One time, develop data for classifiers. Per rule or CWE classifier, filter data.



Novel method developed that successfully and quickly partitioned sets of thousands of tests.

Examine together:

- Precise mapping
- Test suite metadata (structured filenames)
- Rarely examine small bit of code (variable type)

CERT rule	CWE	Count files that match
ARR38-C	CWE-119	0
ARR38-C	CWE-121	6,258
ARR38-C	CWE-122	2,624
ARR38-C	CWE-123	0
ARR38-C	CWE-125	0
ARR38-C	CWE-805	2,624
INT30-C	CWE-190	1,548
INT30-C	CWE-191	1,548
INT30-C	CWE-680	984
INT32-C	CWE-119	0
INT32-C	CWE-125	0
INT32-C	CWE-129	0
INT32-C	CWE-131	0
INT32-C	CWE-190	3,875
INT32-C	CWE-191	3,875
INT32-C	CWE-20	0
INT32-C	CWE-606	0
INT32-C	CWE-680	984

Rows with same color are for different CWEs mapped to same CERT rule

CWE test programs useful to test CERT rules

STONESOUP: **2,608** tests

Juliet: **80,158** tests

- Test set partitioning incomplete (32% left (Static analysis tools might not alert, still!)

Some types of CERT rule violations not tested, in partitioned test suites.

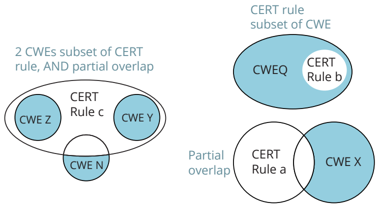
- Possible coverage in other suites

Precise mappings: Defines *what kind* of non-null relationship, and if overlapping, *how*. Enhanced-precision added to "imprecise" mappings.

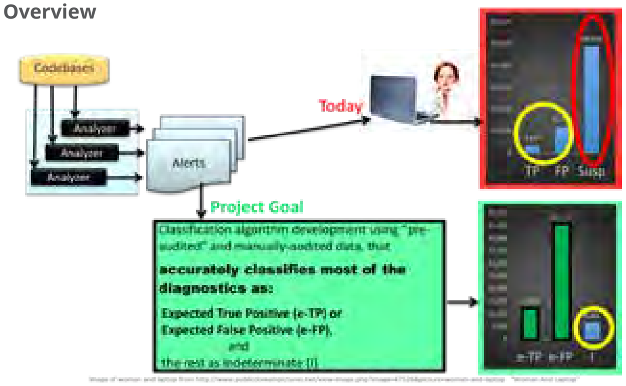
Imprecise mappings ("*some* relationship") → Precise mappings (set notation, often more)

Mappings	
Precise	248
Imprecise TODO	364
Total	612

Now: all CERT C rules mappings to CWE precise



Overview



Classification algorithm development using "pre-audited" and manually-audited data, that

accurately classifies most of the diagnostics as:

Expected True Positive (e-TP) or
Expected False Positive (e-FP),
and
the rest as Indeterminate (I)

Image of woman and laptop from http://www.publicdomainpictures.net/view-image.php?image=[illegible]&picture=woman-and-laptop "Woman And Laptop"

Achievements:

- Preliminary classifier development and testing results (in progress):
  - *Such high accuracies may be artifact of test metadata, currently investigating cause. Expect reduced performance against native files.*
- Xgboost: classifier tested on **56 CWEs** (97.2% avg. accuracy)
- Lasso: classifier tested on **31 CWEs** (98.7% avg. accuracy)
- Xgboost: classifier tested on **44 CERT rules** (95% have **at least 95% accuracy**, and lowest accuracy was **83%**)
- Widely useful general method for using test suites across taxonomies
  - New mappings published on CERT and MITRE websites
- Large archive of "pre-audited" alerts, useful for both CWEs and CERT rules
- Improved tooling that can be transitioned to DoD organizations
- Code infrastructure for classifier development (extensible!)
- Classifier development and testing results (in progress)
- Research paper submission to ICSE 2018 workshop (in progress)
- IEEE SecDev 2017 Tutorial "Hands-on Tutorial: Alert Auditing with Lexicon & Rules"
- 2 SEI blogposts on classifier development
- **Novel** speculative mapping method, for mapping checkers from tools with no public mappings to both CWEs and CERT rules.
  - **16,305** speculatively-mapped-to-CWEs alerts, from 3 tools run on Juliet.

Juliet initial analysis:

Number of **"Bad"** Functions	103,376
Number of **"Good"** Functions	231,476

This is a lot of new data for creating classifiers!

Alert Type	Equivalence Classes: (EC counts a fused alert once)	Number of Alerts Fused (from Different Tools)
HCTP	16,664	2,111
HCFP	32,684	2,699

We automated alert-to-alert matching (alerts fused: same line & CWE), combined with test suite metadata.

Above metrics after only used 3 tools on Juliet.

This project developed a large archive of "pre-audited" alerts useful for building accurate CWE and CERT rule classifiers. It developed reusable code and a method for using test suites across taxonomies.

Principal Investigator

Tracy Cassidy
Insider Threat Researcher

Technical Detection of Intended Violence against Self or Others

Increasing rates of suicide and high profile workplace violence incidents drive the need to identify employees who are on the pathway to intended violence against self and/or others within the workplace.

In support of the DoD mission and mandated insider threat program efforts, this project is determining the extent to which it is possible to technically detect indicators of employees who may be on a path to harm themselves and/or others within the workplace via insider threat detection tools. Project findings can also be used to drive technical requirements that advance the state-of-the-art in automated insider threat data collections and analysis tools. These advances will increase the efficacy and efficiency of DoD Insider Threat Programs and ultimately improve DoD mission assurance.

In Context

This FY2017 project

- builds on our prior Line-funded work in threat analysis and modeling, such as a capability to evaluate insider threat risk and a basis for handling workplace violence and IT sabotage efficiently
- relies on our prior Line-funded research to form a model to mitigate insider threats

Technical Detection of Intended Violence Against Self or Others

INSIDER THREAT PROGRAMS

do not know how to detect heightened risk of workplace violence early enough to prevent harm to individuals and mission.

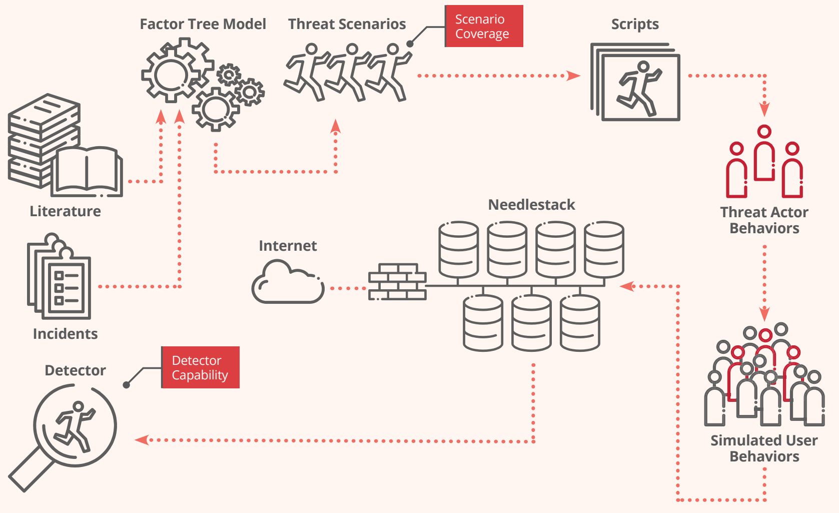
Solution

Develop indicator ontology that maps across the incident lifecycle to online observables and assess technical detection capabilities in a virtual environment

Project Artifacts

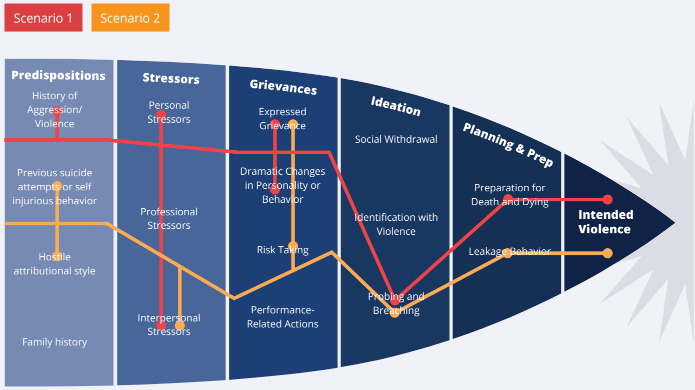
- Scenario Generation Framework
- Coverage and Detection Measures
- Detection Test (Needlestack) Instance
- Report on Balanced Insider Threat Defense

Research Approach

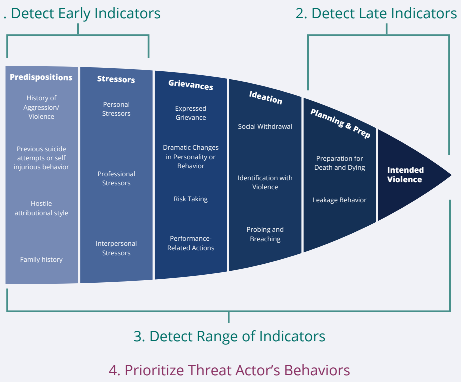


Measures

Scenario Coverage



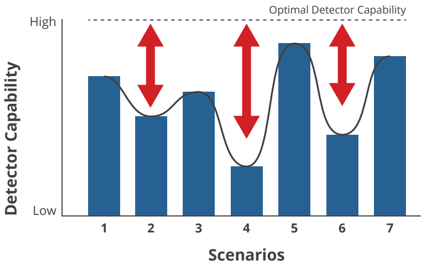
Detector Capability Assessment Criteria


1. Detect Early Indicators
2. Detect Late Indicators
Predispositions
History of Aggression/ Violence
Previous suicide attempts or self injurious behavior
Hostile attributional style
Family history
Stressors
Personal Stressors
Professional Stressors
Interpersonal Stressors
Grievances
Expressed Grievance
Dramatic Changes in Personality or Behavior
Risk Taking
Performance-Related Actions
Ideation
Social Withdrawal
Identification with Violence
Probing and Breaching
Planning & Prep
Preparation for Death and Dying
Leakage Behavior
Intended Violence
3. Detect Range of Indicators
4. Prioritize Threat Actor's Behaviors

Expected Results

- Human Resources (HR) and other detection tools will be configurable for kinetic threats, though they will not have off-the-shelf capabilities
- Detectors will perform better at detecting late stage than early stage indicators
- False alarms for insider kinetic threats may actually be hits for insider cyber threats

Results Format



Future Work

Calendar Year 2017

- Finish developing scripts for the scenarios
- Run the detector capability assessment trial using Needlestack
- Write final research report on method, measures, insider threat tool testing usage, and findings

Post 2017

- Analyze the applicability of project results to the DoD Insider Threat Program
- Develop technical recommendations for reducing intended harm to self and others in the DoD.

Principal Investigator

Kevin Pitstick
Member of the Technical Staff – Engineer

Foundations for Summarizing and Learning Latent Structure in Video

Video data from a variety of DoD platforms is proliferating in the modern battlespace, and expanded dissemination of this video makes it widely available. However, the detection of artifacts of interest in streaming surveillance video is a manually intensive process. As the volume of video data continues to increase, automated video summarization that highlights artifacts of interest is needed.

In this work, we are developing automated and semantically meaningful video summarization and sense making to improve situational awareness, reduce the amount of manual processing necessary, and increase the volume of video data that can be analyzed in near real-time.

In Context

This FY2017 project

- builds on our prior Line-funded research including Structural Multitask Transfer Learning for Improved Situational Awareness and Generalizing Supervised Latent Dirichlet Allocation (a strategy to provide supervision hints to an unsupervised algorithm)
- contributes to our FY2018 Line-funded research to improve the summarizing and searching of video
- draws from sponsored engagements for DoD programs and agencies
- aligns with our work to collaborate with CMU researchers and technologies for tactical analytics, managing information flow in small units, and rapid analysis of streaming data

Foundations for Summarizing and Learning Latent Structure in Video

Problem

The growing volume of streaming and archived surveillance video in the DoD is outpacing the ability of analysts to manually monitor and view it. There is a lack of automated tools available to assist analysts in monitoring real-time video or analyzing archived video.

Solution

Inspired by the past research of CMU Machine Learning professor Dr. Eric Xing, we investigated a new unsupervised video summarization pipeline that functions on extracted clips of objects in motion, rather than whole frames. The goal of the summary is to identify the key "object motion clips" occurring in the video.

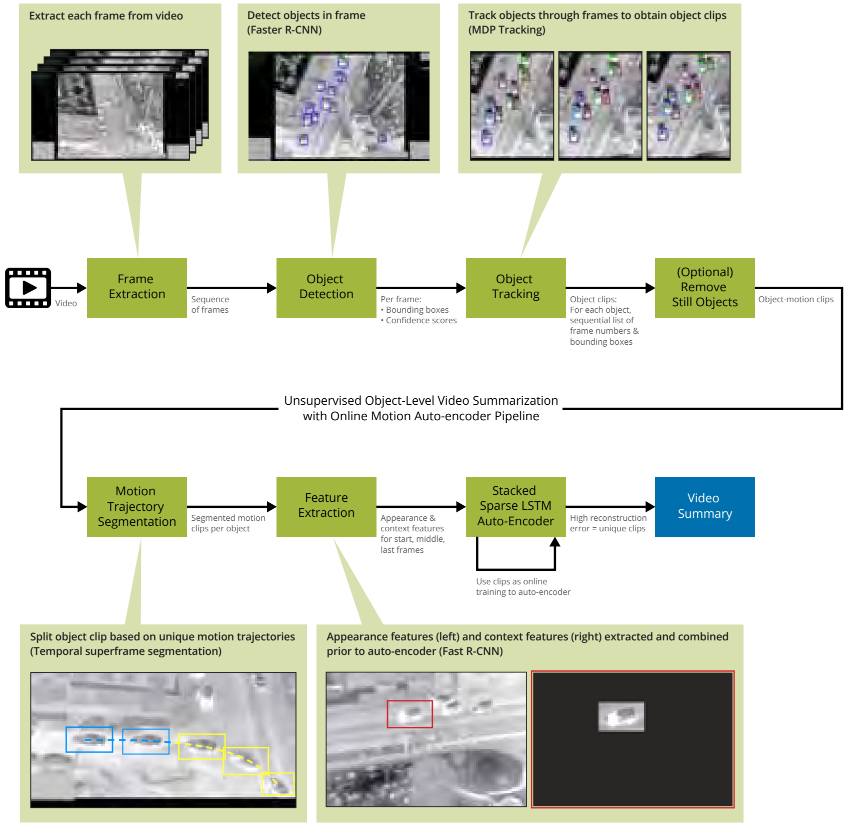
Approach

Unsupervised object-level video summarization with online motion auto-encoder

Offline Training: Initialize auto-encoder by training on random bounding boxes from representative video in order to learn a basic reconstruction capability

Online Processing & Training: For each video being processed:

1. Detect object bounds in each frame
2. Track objects through frames
3. (Optional) Remove clips of still objects
   - Useful for stationary camera
4. Segment object clips based on unique motion trajectories
5. Extract appearance and context features from the start, middle, and last frame of each clip
6. Feed the clip features through a three-layer, sparse, long short-term memory (LSTM) auto-encoder
   - Clips with high reconstruction error are collected as the "summary"
   - As clips are processed, use them as online training for the auto-encoder



Extract each frame from video

Detect objects in frame (Faster R-CNN)

Track objects through frames to obtain object clips (MDP Tracking)

Video → Frame Extraction → Sequence of frames → Object Detection → Per frame:
- Bounding boxes
- Confidence scores

→ Object Tracking → Object clips: For each object, sequential list of frame numbers & bounding boxes → (Optional) Remove Still Objects → Object-motion clips

Unsupervised Object-Level Video Summarization with Online Motion Auto-encoder Pipeline

Motion Trajectory Segmentation → Segmented motion clips per object → Feature Extraction → Appearance & context features for start, middle, last frames → Stacked Sparse LSTM Auto-Encoder → High reconstruction error = unique clips → Video Summary

Use clips as online training to auto-encoder

Split object clip based on unique motion trajectories (Temporal superframe segmentation)

Appearance features (left) and context features (right) extracted and combined prior to auto-encoder (Fast R-CNN)

	Sparse Coding	Stacked Sparse Auto-encoder	Stacked Sparse Auto-encoder	Stacked Sparse LSTM Auto-encoder (OURS)
AUC score	0.4252	0.4354	0.5680	**0.5908**
AP score	0.1542	0.1705	0.2638	**0.2850**
F-measure	0.1284	0.1662	0.2795	**0.2901**

Table 1: Object-level summarization results between competing approaches on Orangeville dataset

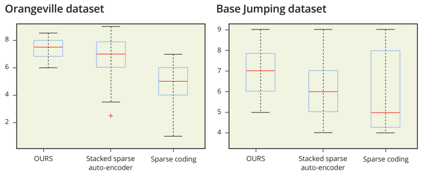


Figure 1: User study evaluation scores between competing approaches on Orangeville and Base Jumping datasets

Results

Quantitative: Our results matched or exceeded competing algorithms on benchmark datasets (see Table 1)

Qualitative: In a user study, our approach received higher and more consistent evaluation scores (see Figure 1)

Artifacts

Software

- Prototype utilizing the unsupervised, online, object-level video summarization pipeline
- Video Markup Tool for annotating spatial-temporal object clips within video

Paper

- Submission to IEEE Transactions on Cybernetics

Datasets

- "Orangeville" benchmark for object-level summarization – dataset & annotations
- Annotations & model for vehicle detection in FBI IR surveillance data

Future Work

FY18 Project: Summarizing and Searching Video

- Finish investigation of current pipeline to summarization of full-motion video (FMV) datasets
- Unsupervised activity clustering, utilizing object-motion clips as basis
- AFRL collaboration to explore applying analysis techniques to existing DoD problems
  - e.g., Nothing Significant to Report

In summary, we investigated object-level video summarization as an approach to identify the key segments in video with the final goal of applying these techniques to analyze real DoD surveillance data.

Principal Investigator

Dr. Sam Procter
Architecture Researcher

Guided Architecture Trade Space Exploration for Safety-Critical Software Systems

Safety-critical systems are composed of embedded software components selected from thousands of vendors. As the embedded software trade space grows, developers cannot test all candidate designs exhaustively to decide which ones optimize performance, or even comply with system requirements. Too often, the legacy of uninformed choices is avoidable rework and higher sustainment cost.

In this work, we have created a new tool prototype that automatically explores a system's trade space: its possible combinations of system software, hardware, and configuration options. This exploration is interactive and user-guided, and it lets system designers uncover subtle relationships between various quality attributes. This work builds on previous SEI technologies for modeling system architecture and enables a new paradigm—design by shopping—of system design.

In Context
This FY2017 project

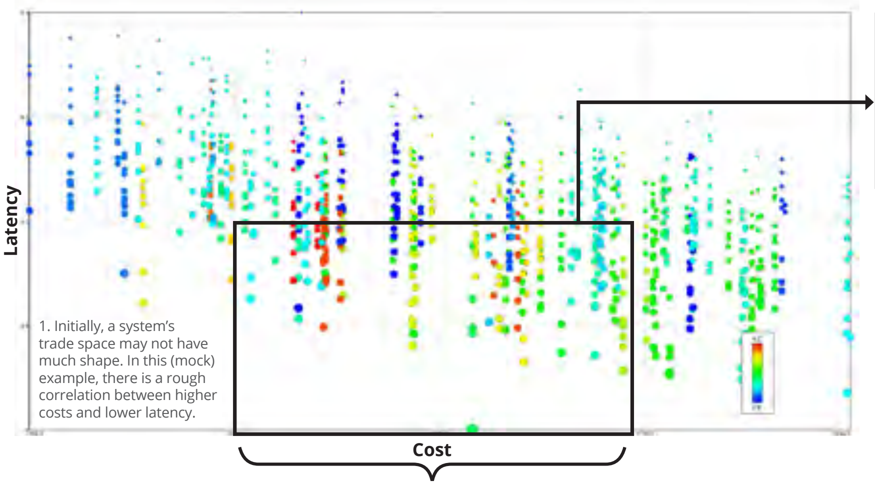
- extends our research to form the Architecture-Led Incremental System Assurance (ALISA) method, reduce avoidable complexity in embedded software, and identify software attribute tradeoffs
- contributes to our FY2018 Line-funded research project into integrating safety and security engineering

Guided Architecture Trade Space Exploration of Safety Critical Software Systems

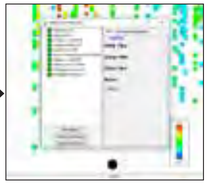
Modern systems are so heavily interconnected that the impacts of component and configuration choices can be difficult to know at design time. In this work, we partially automate the exploration of a system's architectural trade space to enable system designers to rapidly evaluate system design options. The result is a graphical, user-guided suite of tools that enables a 'design-by-shopping' style of system design.

We build on SEI's existing work in high-fidelity system architecture modeling.
In previous work, the SEI created the *Architecture Analysis and Design Language* (AADL), which enables the creation of very detailed models of a system's architecture. The SEI also maintains the *Open Source Architecture Tool Environment* (OSATE), which contains a number of analyses for AADL models, ranging from weight and latency calculations to sophisticated projections of system error-behavior.

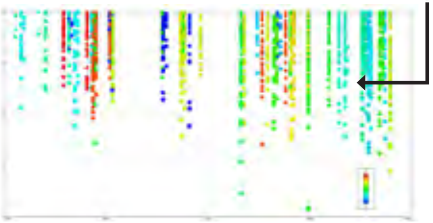
A system's architectural trade space is the set of all possible candidate designs plotted in n-dimensional space where each dimension is a different quality attribute.
Penn State's ARL Trade Space Visualizer (ATSV) enables designers to visually explore multidimensional and potentially infinite data sets. It uses an evolutionary algorithm to learn which input characteristics correspond to which outputs; this lets designers refine their searches and generate additional candidate architectures that meet particular criteria.



1. Initially, a system's trade space may not have much shape. In this (mock) example, there is a rough correlation between higher costs and lower latency.



3. Once a satisfactory architecture has been identified, the designer can click the point to reveal the component and configuration options chosen, as well as exact quality attribute values for each dimension.

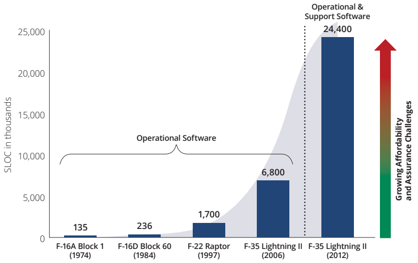


2. Designers can refine their searches, and ATSV will generate more candidate architectures in the reduced search area. This is done automatically, using an evolutionary algorithm to learn which input values correspond to which outputs.

This work has three primary tasks:

1. Extend existing architecture modeling language (SEI's AADL) to encode component choices and their interactions.
2. Extend existing architecture modeling tooling (SEI's OSATE) to automatically analyze the resulting system for cost, weight, performance, etc.
3. Enable trade space visualizer (Penn State's ATSV) to automatically select valid components and configurations, and visually display analysis results.

Why do we need something new?
System development costs are rising at an unsustainable rate. Much of this rise is driven by software, which presents a number of unique challenges in terms of massive customizability/configurability and subtle interactions with other components. SEI's work in system architecture modeling addresses this need head-on by modeling software, hardware, and the bindings between the two.



Software as % of total system cost

1997: 45% → 2010: 66% → 2024: 88%

Graphic: Hagan/Sorenson, "Delivering Military Software Affordably," *Defense AT&L*, Mar-Apr 2013

How does 'design-by-shopping' work?
Much like traditional shopping, design-by-shopping begins with a broad search of candidates that potentially meet some need. Individual candidates are evaluated according to attributes like cost, performance, weight, power, etc. If none of the candidates are satisfactory, more can be generated – or the trade space can be refined to focus the search. If one of the candidates is satisfactory, though, then the designer can view its exact configuration and proceed to more detailed modeling and analysis.

This work is an enabling technology for future architecture modeling research at the SEI.
Any quantitative analysis can be used as a dimension of the trade space visualization. Sophisticated new analyses for AADL are being created in OSATE to analyze complex system properties like safety and security. Once built, it's relatively straightforward to automate these analyses and connect them to ATSV.

This project combines a number of technologies to enable a new paradigm of system design. As it relies heavily on quantitative analyses of system architectures, in the future we will look into novel quantification strategies for traditionally qualitative attributes like safety and security.

New Solutions for Verifying Critical Systems

The DoD faces an overwhelming challenge when developing complex systems: how to verify that the software will do the right thing at the right time—which is especially important when the work is safety or mission critical. Working with the Air Force Research Laboratory at Wright-Patterson Air Force Base, SEI researchers continue to make significant advances in software model checking and timing verification that can be applied earlier in the development lifecycle to reduce the cost of assurance.

V&V for Security from the Center to the Edge

Principal Investigator

Dr. Grace Lewis
Principal Researcher

Authentication and Authorization for Internet of Things (IoT) Devices in Edge Environments

Integrating Internet of Things (IoT) devices into DoD tactical systems expands the attack surface in environments that are resource-constrained and adversarial. Existing IoT security approaches are insufficient because they typically do not consider these environments.

In this work, we are analyzing, extending, and influencing the IETF ACE (Authentication and Authentication for Constrained Environments) working group proposal for authentication and authorization of IoT devices such that it

- addresses high priority threats of tactical environments, such as node impersonation and capture
- considers operations in DIL (disconnected, intermittent, limited) environments

In Context

This FY2017 project

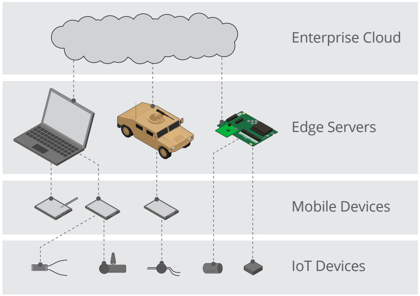
- builds on our prior work in the mobile communication and computing needs of edge users (e.g., trusted identities, delay-tolerant data sharing, and group-context-aware data and resource sharing)
- contributes to our FY2018 Line-funded research into assurance for software-defined IoT
- draws from our collaboration with CMU researchers and sponsored engagements to reduce risk through architecture analysis

Two Perspectives on IoT Security

"IoT Security Standards" and "Software-Defined Networking for IoT Security"

FY17: Authentication and Authorization for IoT Devices in Edge Environments

Evaluation, adaptation, and implementation of an IETF proposal for authentication and authorization in constrained environments (ACE) to enable future integration of ACE-compliant IoT devices into DoD systems



Tactical Edge System

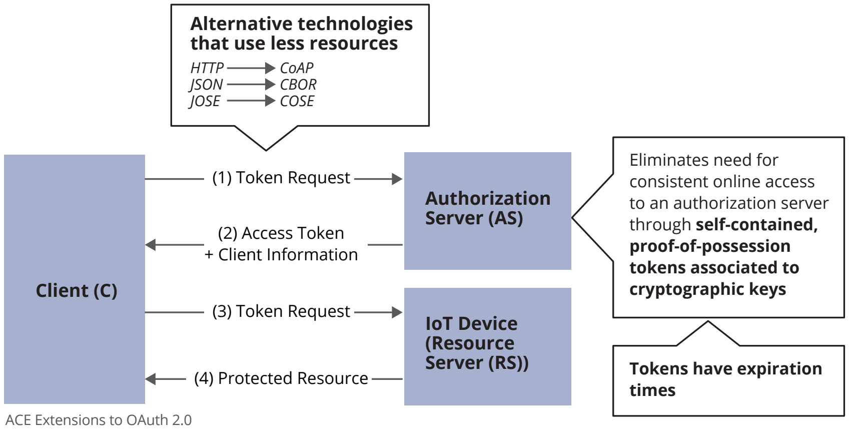
Goal:

Develop a solution for authorization and authentication of IoT devices that

- addresses high-priority threats of tactical edge environments,
- operates in DIL environments, and
- considers resource constraints of IoT devices

ACE (Authentication and Authorization in Constrained Environments)

- IETF proposal in Working Group Status—next step is Proposed Standard
- Extends OAuth 2.0 to IoT devices
- Addresses some of the challenges of tactical environments



Alternative technologies that use less resources

HTTP → *CoAP*
JSON → *CBOR*
JOSE → *COSE*

Client (C)
(1) Token Request
(2) Access Token + Client Information
(3) Token Request
(4) Protected Resource
Authorization Server (AS)
IoT Device (Resource Server (RS))

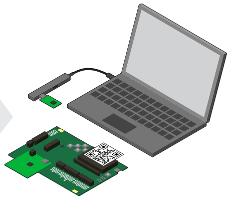
Eliminates need for consistent online access to an authorization server through **self-contained, proof-of-possession tokens associated to cryptographic keys**

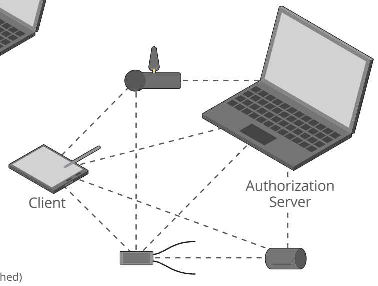
Tokens have expiration times

ACE Extensions to OAuth 2.0

Threat modeling identified the following gaps in ACE.

? Gaps	Solution
Bootstrapping of credentials is considered out-of-scope	Pairing mechanism for IoT devices that involves the use of QR codes as an out-of-band channel for exchanging initial encryption keys between IoT devices and the Authorization Server (AS)
Assumption of short periods of disconnection	Integration with delay-tolerant mechanisms and opportunistic routing for clients and IoT devices to reach AS
On-demand token revocation	Periodic introspection between IoT devices and the AS, and clients and the AS

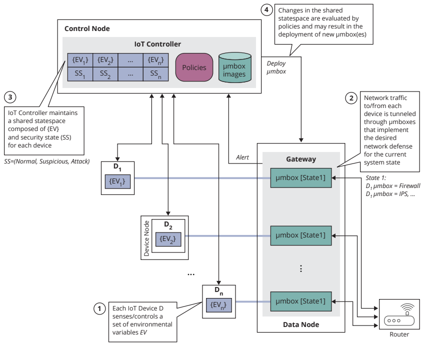




Periodic Introspection

```
if (introspection interval reached)
 for each token t in token list
         r = send_introspection_message(t)
         if (r == "Invalid")
           delete_token(t)
 end for
 reset introspection interval
end if
```

FY18: High-Assurance Software-Defined IoT Security



1. Each IoT Device D senses/controls a set of environmental variables *EV*
2. Network traffic to/from each device is tunneled through µmboxes that implement the desired network defense for the current system state
3. IoT Controller maintains a shared statespace composed of {EV} and security state (SS) for each device
4. Changes in the shared statespace are evaluated by policies and may result in the deployment of new µmbox(es)

Dynamic deployment of network defenses based on composite state analysis of all controlled IoT devices

Principal Investigator

Dr. Dionisio de Niz
Principal Researcher

Certifiable Distributed Runtime Assurance

Leveraging rising software complexity and use of machine learning (ML) techniques, the DoD is increasingly using complex non-deterministic systems. Runtime Assurance (RA) is a promising technique for ensuring safe behavior of those systems because they cannot be verified satisfactorily prior to deployment. RA relies on the use of an enforcer to monitor the target system and correct (i.e., preserve the safety of) its behavior at runtime.

However, current RA techniques are restricted to single domains (i.e., types of analyses); rely on unverified and potentially inconsistent enforcers that can be circumvented easily; and require system-wide re-verification when an enforcer is changed, added, or removed.

In this work, we are addressing those challenges in the context of distributed real-time systems (DRTS) by creating tools and techniques to

- express enforceable policies in multiple domains, including logical and timing correctness
- verify correctness of an enforcer implementation against its policy
- combine multiple enforcers and resolve any inconsistencies between their behavior
- verify that enforcers across multiple nodes of DRTS implement a global safety policy
- deploy enforcers so that they cannot be circumvented by a well-defined attacker (e.g., one that has control of one/more monitored components)

We are validating our results on DoD-relevant examples.

In Context

This FY2017–18 research

- extends sponsored engagement work and prior Line-funded research in formal methods to verify distributed algorithms, distributed adaptive real-time (DART) systems (e.g., autonomous multi-unmanned air system missions), and software with timers and clocks
- aligns with our work to develop the open-source model checker SeaHorn and to generate property-directed test cases

Certifiable Distributed Runtime Assurance

Challenge

Assure Safety of Distributed Cyber-Physical Systems

- Unpredictable Algorithms (Machine Learning)
- Coordinating multiple vehicles (distributed) to achieve mission

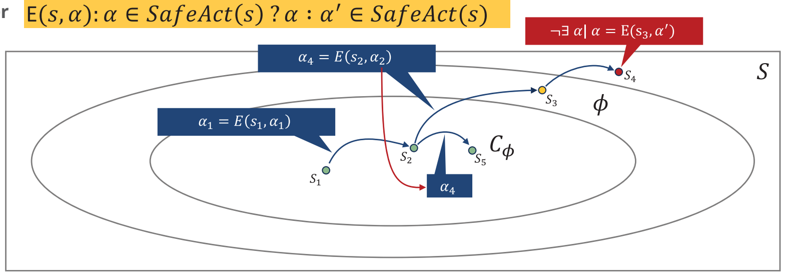
Solution

- Add simpler (verifiable) runtime enforcer to make algorithms predictable
- Formally specify, verify, and compose multiple enforcers
- Enforcer intercepts/replaces unsafe action at right time

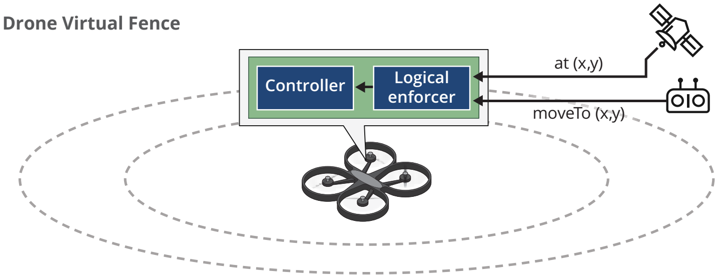
Formalization (Time-Aware Logic)
State of system: variable values

- State variables: V_S
  e.g., (x,y) position
- Action variables: V_Σ
  e.g., move-to(x,y) action
- System state: $s{:}V_S \mapsto D \in S$
- Actions: $\alpha{:}V_\Sigma \mapsto D$
- Behavior: periodic state transition
  $R_P\,(\alpha) \subseteq S{\times}S$
  $R_P\,(\alpha{,}s) = \{s'|(s{,}s'\,) \in R_P\,(\alpha)\}$
- Safe state: $\phi \subseteq S$
- Enforceable state:
  $\phi \supseteq C_\phi = \{s \,\big|\, \exists\alpha \in \Sigma{:}R_P\,(\alpha{,}s) \in C_\phi\,\}$
- Safe Actuation
  $SafeAct(s) = \{\alpha|R_P\,(\alpha{,}s) \in C_\phi\}$

Enforcer $E(s,\alpha){:}\, \alpha \in SafeAct(s)\ ?\ \alpha : \alpha' \in SafeAct(s)$


$\neg\exists\, \alpha|\ \alpha = E(s_3,\alpha')$
$\alpha_4 = E(s_2,\alpha_2)$
S
s_4
s_3
ϕ
$\alpha_1 = E(s_1,\alpha_1)$
s_2
s_5
C_ϕ
s_1
α_4

Enforcing Drone Virtual Fence



Timing enforcement

- Unverified software may never finish!
- => No action produced to be enforced!

Temporal enforcer

- Protect other tasks from bogus never-ending (or large) executions
- Produce default safe actuation if task takes too long

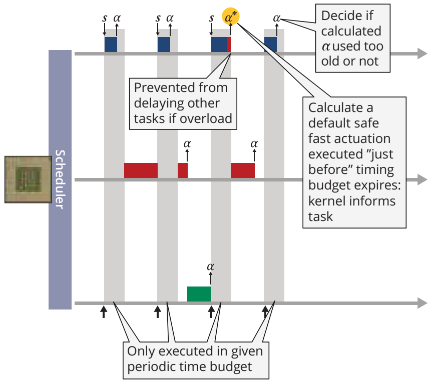
How

- Each task gets a CPU budget
  stop task if budget exceeded
- If task about to exceed budget
  execute safe action

Timing guarantees

- Never allow task to exceed budget
- Always execute actuation

Safe actuation on timing enforcement

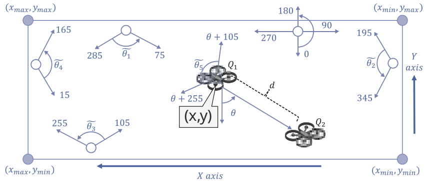


Decide if calculated α used too old or not

Prevented from delaying other tasks if overload

Calculate a default safe fast actuation executed ”just before” timing budget expires: kernel informs task

Only executed in given periodic time budget



Composing enforcers

- System with multiple enforcers: virtual fence+ collision avoidance
- Safe actions from different enforcers may conflict
  Drone at fence limit + other drone approaching
- Enforcer composition to resolve conflicts
  (1) priority based (2) utility maximization

Formalization

Enforcer: $\boldsymbol{E}{:}(\boldsymbol{P}, \boldsymbol{C_\phi}, \boldsymbol{\mu}, \boldsymbol{U})$

$\mu(s) \subseteq SafeAct(s)$; U: utility

No conflict:

$E_1\,(P_1, C_{\phi_1}, \mu_1, U_1\,), E_2(P_2, C_{\phi_2}, \mu_2, U_2\,)$
$\mu_{1,2} : \mu_1 \cap \mu_2$

Conflict: Priority resolution

$\mu_{1,2} : \mu_1 \cap \mu_1 \neq \emptyset\ ?\ \mu_1 \cap \mu_2 : \mu_1$

Conflict: Utility maximization

$\mu_{1,2} : \mu_1 \cap \mu_2 \neq \emptyset\ ?\ argmax_{\alpha\in\mu_1 \cap \mu_2}\sum U_i\,(s{,}\alpha'\,)$:
$argmax_{\alpha\in\mu_1}\sum U_i\,(s{,}\alpha')$

Enforcers allows verification of complex CPS: Autonomous Vehicles

- Limit misbehavior
- With Verifiable Enforcers
- Result: Verified whole system

Principal Investigator

Dr. Rick Kazman
Visiting Scientist

Dynamic Design Analysis

Increasingly, software systems are composed at runtime. Yet, the impact of runtime composition on design quality is unknown. Static analysis, a state-of-the-practice approach, has demonstrated that dependency-caused design hotspots make security vulnerabilities more likely, but it does not detect the effect of dynamic dependencies.

In this work, we are creating tooling to identify dynamic dependencies as well as other information that is not available via static, parsing-based approaches, to both determine and augment the information that is missing in static approaches. One transition opportunity for DoD is envisioned to be extensions to the open standard-based 18F project on Compliance Masonry for automate production of Authority to Operation (ATO) documentation.

In Context

This FY2017 project

- builds on our prior Line-funded research to establish model--based engineering practice and reflects needs identified through several sponsored engagements
- contributes to FY2018 Line-funded research into rapid software composition by assessing untrusted components
- aligns with research and development to form the architecture-led virtual integration practice, a virtual upgrade validation method, and a reliability improvement and validation framework that draw from CMU SEI technical leadership of the Architecture Analysis and Design Language (AADL) standard for embedded software system modeling and analysis

Dynamic Design Analysis

Increasingly, software systems are composed at runtime. Yet, the impact of runtime composition on design quality is unknown. Static analysis, a state-of-the-practice approach, has demonstrated that dependency-caused design hotspots make security vulnerabilities more likely, but it does not detect the effect of dynamic dependencies.

In this work, we are creating tooling to identify dynamic dependencies as well as other information that is not available via static, parsing-based approaches, to both determine and augment the information that is missing in static approaches.

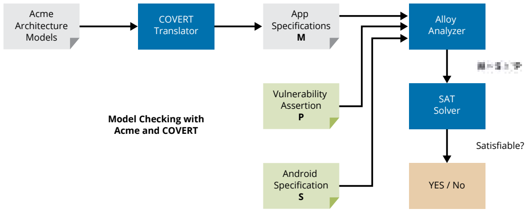
Technical challenges:

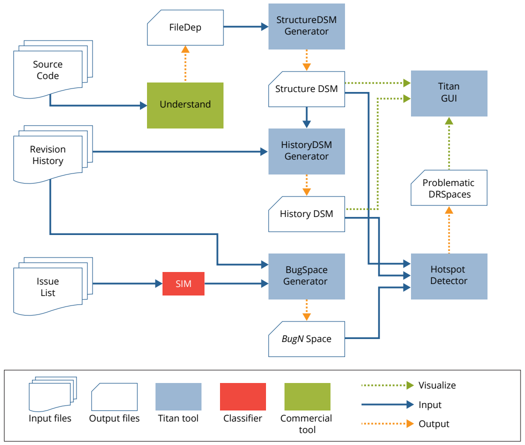
1. Detecting dynamic dependencies (DDs)
2. Determining whether DDs create new kinds of architectural flaws
3. Determining the consequences of DD-induced design flaws
4. Proposing refactorings to remedy flaws

Two approaches employed:

Approach 1: Acme/Alloy-based Analysis

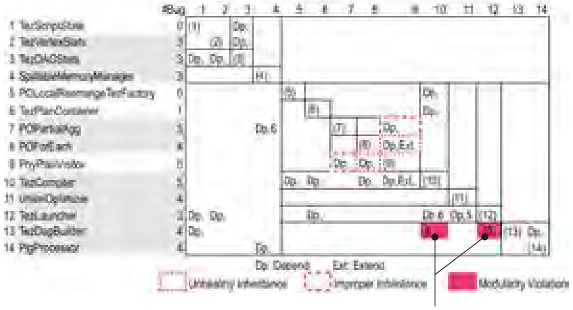
- defined a formal architectural model of the Android framework as an architectural style in Acme
- used Soot to capture an abstract model of each app and translated these models to a single architecture in Acme
- defined Acme invariants to check architecture properties (e.g., apps cannot communicate implicit intents to apps that have a lower trust level)
- translated these models to Alloy specifications
  - checked that privilege escalation is not possible (or provide a counter-example where it is possible)





Approach 2: Titan-based analysis

- reverse-engineering of architecture facts using Understand
- built static representations of architecture dependencies as DSMs
- added historic dependencies from the revision history
- captured potential dynamic relationships as issues from the issue list
- defined a new architecture view: the issue space
- showed that when a system has files revised in many different issues—what we call a hotspot--these “shared” files are connected
- these design dependencies are frequently only seen at run-time and they almost always have design flaws leading to bugs, security flaws, and maintenance problems
- thus they should be analyzed and refactored



Identified Design Flaws

Conclusions and Future Work

Design flaws are introduced unknowingly by the daily activities of developers—adding features and fixing bugs. If left untreated they degrade the system over time, making it harder to understand, maintain, extend, and fix.

By considering dynamic information in conjunction with static information we can precisely locate such design flaws, and hence determine the root causes of bugs more quickly.

This information is not available solely via static analysis; dynamic dependencies must be considered. But such information is difficult to obtain. We have explored two methods for doing so.

These design flaws are the roots of technical debt.

In our future work, we are examining the relationship between such design flaws and security bugs.

SEI Research Review 2017

Principal Investigator

Dr. Peter Feiler
SEI Fellow and Principal Research Scientist

Presenter

Dr. Sam Procter
Architecture Researcher

Automated Assurance of Security Policy Enforcement

As DoD mission and safety-critical systems become increasingly connected, exposure due to security infractions is likewise increasing. This project developed techniques to detect vulnerabilities early in the lifecycle in architecture models by producing tools to

- detect security policy violations early
- assure that the system implementation enforces the policies and that no security risks are introduced by the runtime architecture
- automate the execution of security assurance plans

Tools produced in this project have been released under an open-source license and are available on the SEI Github code repository (https://github.com/cmu-sei/AASPE).

In Context

This FY2016–17 project

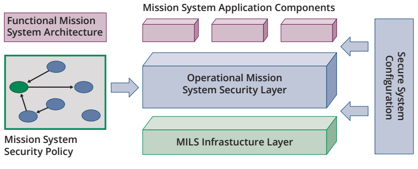
- builds on our prior Line-funded research to establish model-based engineering practice and reflects needs identified through several sponsored engagements
- contributes to an FY2018 line-funded research into integrating safety and security engineering
- aligns with research and development to form the architecture-led virtual integration practice, a virtual upgrade validation method, and a reliability improvement and validation framework that draw from CMU SEI technical leadership of the Architecture Analysis and Design Language (AADL) standard for embedded software system modeling and analysis

Automated Assurance of Security Policy Enforcement

As safety-critical system have become more connected, “closed” system assumptions are no longer valid and security threats affect safe system operation.

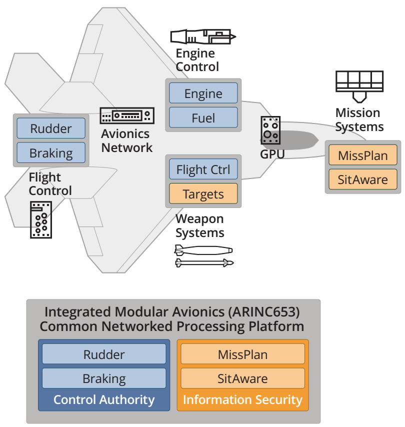
Virtual system integration and analysis of embedded software systems has been embraced by the safety-critical system community to address exponential growth in system development cost due to increased interaction complexity and mismatched assumptions in embedded software systems.

In this project, we demonstrate how the virtual system integration approach can be extended to address security concerns at the architecture level to complement code level security analysis.

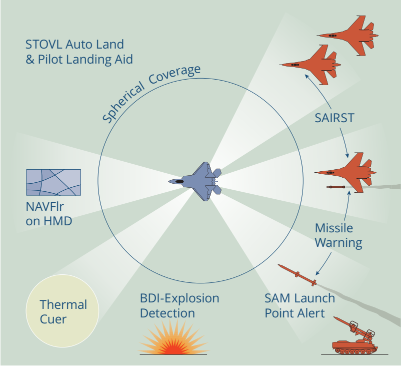

Functional Mission System Architecture
Mission System Application Components
Operational Mission System Security Layer
MILS Infrastucture Layer
Secure System Configuration
Mission System Security Policy

We Apply Multiple Independent Levels of Security (MILS) Framework to Software System Integrity

Security Challenges as Safety-critical Systems Become Connected



From closed to connected systems



Security Attacks on External Channels and System Internal Vulnerabilities

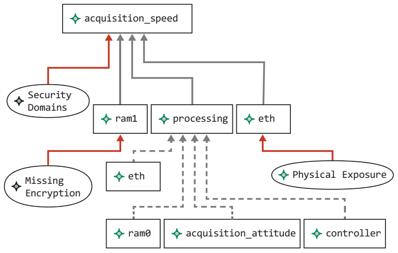
Safety-critical avionics systems use partitioning to achieve fault isolation

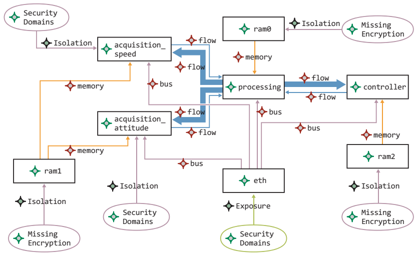
Our focus is on security policy specification and its enforcement

We utilize the SAE International Architecture Analysis & Design Language (AADL) industry standard to model and analyze embedded software systems. It includes an annex for fault modeling and analysis.

We analyze security policy specifications for consistency and gaps in flow constraints and isolation requirements.

We analyze the software system architecture for potential enforcement vulnerabilities due to incorrect deployment of security mechanisms.

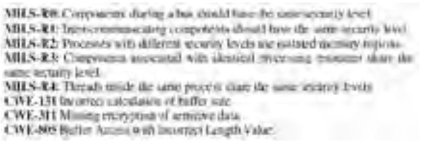




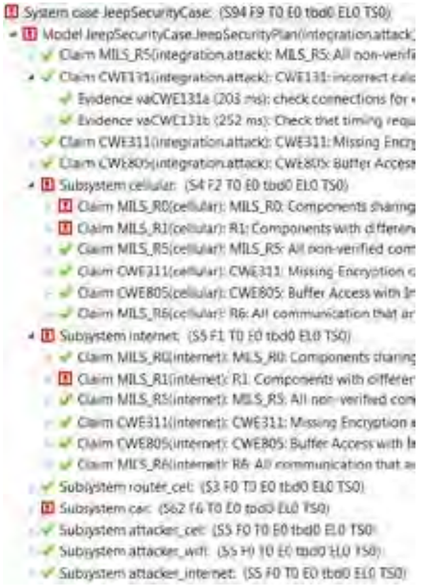
Automated Assurance through Continuous Analysis of Potential Architecture Level Security Policy and Enforcement Vulnerabilities

We leverage the Architecture-Led Incremental System Assurance (ALISA) capability in the Open Source AADL Tool Environment (OSATE).

Executable verification plans identify how potential architecture level security vulnerabilities are addressed through model-based analysis.



MILS-R0: Components sharing a bus should have the same security level
MILS-R1: Intercommunicating components should have the same security level
MILS-R2: Processes with different security levels are isolated memory regions
MILS-R3: Components associated with identical processing resources share the same security level
MILS-R4: Threads inside the same process share the same security levels
CWE-131 Incorrect calculation of buffer size
CWE-311 Missing encryption of sensitive data
CWE-805 Buffer Access with Incorrect Length Value



```
! System case JeepSecurityCase: (S94 F9 T0 E0 tbd0 EL0 TS0)
  ! Model JeepSecurityCase.JeepSecurityPlan(integration.attack_
      ✓ Claim MILS_R5(integration.attack): MILS_R5: All non-verifi
      ✓ Claim CWE131(integration.attack): CWE131: incorrect calc
          ✓ Evidence vaCWE131a (203 ms): check connections for
          ✓ Evidence vaCWE131b (252 ms): Check that timing requ
      ✓ Claim CWE311(integration.attack): CWE311: Missing Encry
      ✓ Claim CWE805(integration.attack): CWE805: Buffer Acces
      ! Subsystem cellular: (S4 F2 T0 E0 tbd0 EL0 TS0)
          ! Claim MILS_R0(cellular): MILS_R0: Components sharing
          ! Claim MILS_R1(cellular): R1: Components with differen
          ✓ Claim MILS_R5(cellular): MILS_R5: All non-verified com
          ✓ Claim CWE311(cellular): CWE311: Missing Encryption o
          ✓ Claim CWE805(cellular): CWE805: Buffer Access with In
          ✓ Claim MILS_R6(cellular): R6: All communication that ar
      ! Subsystem internet: (S5 F1 T0 E0 tbd0 EL0 TS0)
          ✓ Claim MILS_R0(internet): MILS_R0: Components sharing
          ! Claim MILS_R1(internet): R1: Components with differer
          ✓ Claim MILS_R5(internet): MILS_R5: All non-verified con
          ✓ Claim CWE311(internet): CWE311: Missing Encryption a
          ✓ Claim CWE805(internet): CWE805: Buffer Access with In
          ✓ Claim MILS_R6(internet): R6: All communication that ar
      ✓ Subsystem router_cel: (S3 F0 T0 E0 tbd0 EL0 TS0)
      ! Subsystem car: (S62 F6 T0 E0 tbd0 EL0 TS0)
      ✓ Subsystem attacker_cel: (S5 F0 T0 E0 tbd0 EL0 TS0)
      ✓ Subsystem attacker_wifi: (S5 F0 T0 E0 tbd0 EL0 TS0)
      ✓ Subsystem attacker_internet: (S5 F0 T0 E0 tbd0 EL0 TS0)
```

Principal Investigators

Dr. Eliezer Kanal
Technical Manager

Michael Coblenz
Graduate Student, CMU Computer Science Department

Obsidian: A Safer Blockchain Programming Language

The Defense Advanced Research Projects Agency (DARPA) and other agencies are expressing significant interest in blockchain technology because it promises inherent transparency, resiliency, forgery-resistance, and nonrepudiation that can be used to protect sensitive infrastructure. At the same time, numerous high-profile incidents of coding errors in blockchain applications causing major damage to organizations have raised serious concerns about adoption of this technology.

CMU SEI and the CMU Computer Science Department are creating a novel programming language—Obsidian—specifically tailored to secure blockchain software development, significantly reducing the risk of such coding errors.

In Context

This FY2017 project

- applies the science of cybersecurity to address software assurance and survivability throughout the development and acquisition lifecycles
- builds on our technical work and collaboration with CMU in automated static (source) code analysis open-source tools, prioritizing alerts with classification models, and embedded system safety analysis
- aligns with our other Line-funded research into automated code repair and vulnerability analysis, secure coding standards, and malware analysis

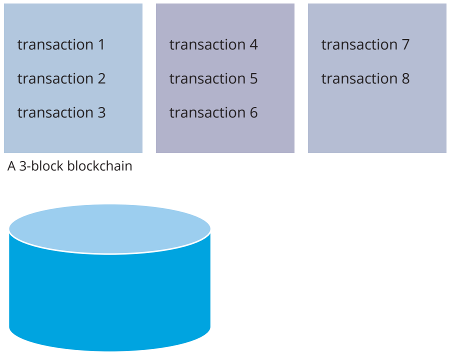
Obsidian

A Safer Blockchain Programming Language

What is a blockchain?

- Blockchain programming environments provide *shared, global* state on *untrusted, distributed computing nodes.*
- Global state consists of *smart contracts,* which include both data structures and *transactions* that manipulate them.
- Transactions can *deploy* smart contracts to the blockchain (initializing their state), or *invoke* code implemented in specific deployed smart contracts.
- "Code is law": a principle that suggests that a contract's code specifies an agreement between parties. This principle implies that contracts are *immutable*; bugs in them cannot be fixed after deployment.

One blockchain network node



A 3-block blockchain

Key-value store with state of all contracts

The nodes execute a *consensus protocol* by which they agree on which transactions have been processed and in what order. After consensus is reached, all nodes agree on the transactions and their ordering.

Blockchain facts	Consequences	Design approach
Smart contracts can hold **resources**.	Bugs can transfer resources to the wrong party or lose them entirely. A bug was recently exploited to steal over US$40M from a contract.	Obsidian models resources with **linear types**, which statically restrict lifecycles, so that **resources** cannot be accidentally lost.
Modifying a contract's code could change an agreement that parties have settled on.	Smart contracts are **immutable** once deployed, but this means that bugs are not fixable.	Support developer-authored **specifications** and use **verification** tools to prove that the code satisfies the specifications. Use a strong, static type system to detect as many bugs as possible at compile time.
Proposed application domains, such as finance and medical records, cannot tolerate serious bugs.	**Correct software** is paramount.	
Many proposed applications are inherently *stateful*.	Blockchain applications commonly implement **state machines**. Behavior and available transactions depend on the current state.	Obsidian is a **typestate-oriented language**, representing *state* in *types* and statically preventing some invalid invocations. For example, a Bond that has been bought is in the Bought state and cannot be bought again.

Annotated example code

```
interface account { transaction pay (money m); }
resource contract money {…}; ← Instances of resource contracts,
                               unlike other contracts, always have
                               exactly one owning reference.
contract Bond {
   account seller;
   Bond(account s) {
      seller = s;
      -> Offered(); ← The constructor transitions to
                      Offered state at the end.
   }
   state Offered {
      transaction buy(money m, account b) {
         seller.pay(m); ← After this line, the transaction no longer owns m.
         -> Sold({buyer = b}); ← Transition to Sold state with buyer
                                 field set to b.
      }
   }
   state Sold {
      account buyer;
      transaction makePayment(money m) {
         buyer.pay(m);
      }
   }
}
contract ErroneousClient {
   transaction badTransaction() {
     Bond.Offered b = new Bond(…);
     b.buy(…); ← Type of b after this line is Bond.Sold due to buy() invocation.
     b.buy(…); ← Compile error: b is of type Bond.Sold, which has no
                 buy() transaction.
  }
}
```

Status and Upcoming Work

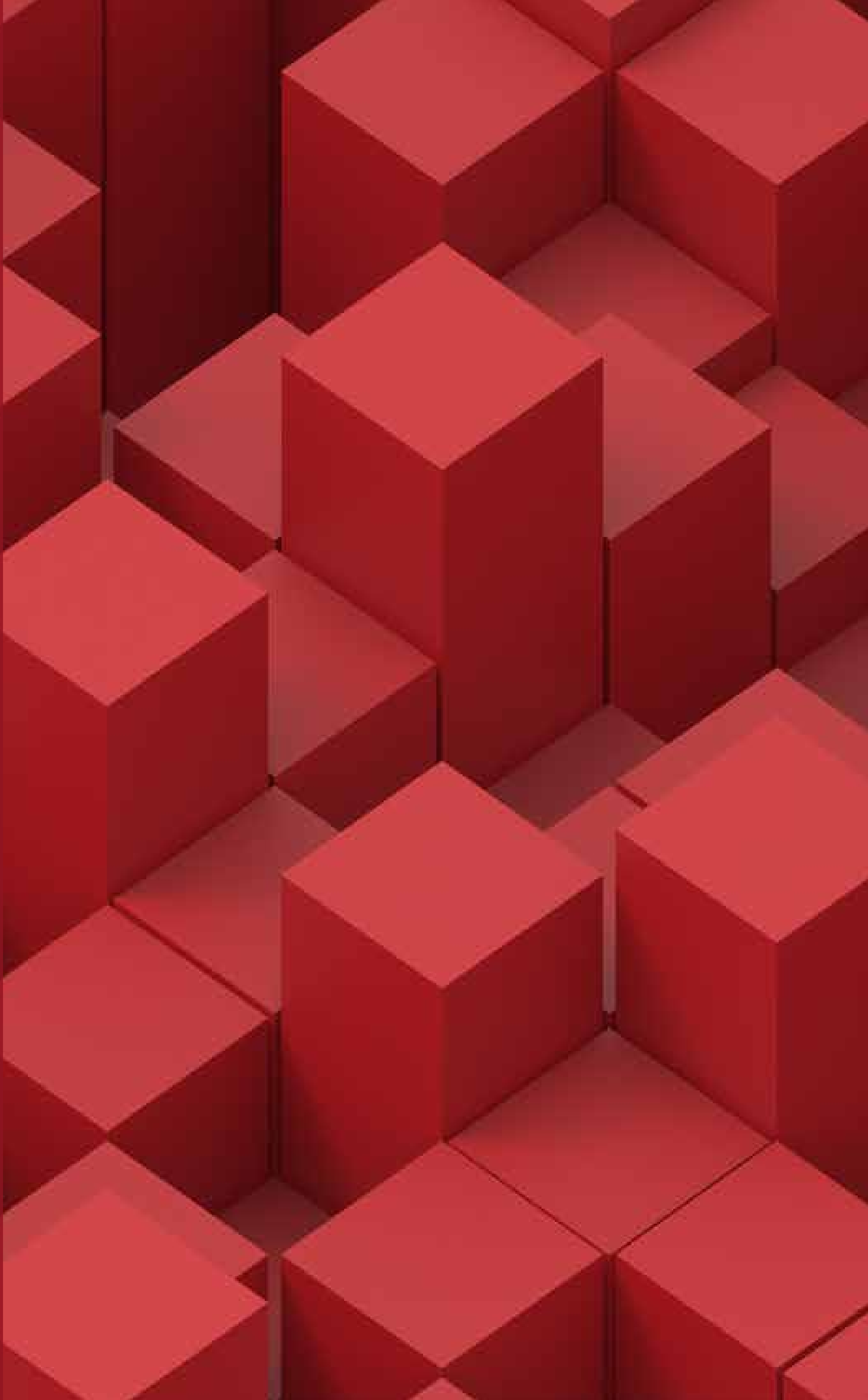
- An initial compiler translates Obsidian code to Java (for execution on a mock blockchain) and to Dafny (for verification).
- The compiler includes a typechecker based on a draft typesystem.
- We plan to port Obsidian to Hyperledger Fabric and complete the compiler.
- Our evaluation will ask users to write programs in Obsidian and either Hyperledger Composer or Solidity (existing blockchain development tools) and compare task completion times and rates and kinds of bugs.
- We are iteratively evaluating portions of the language design in the context of Java so that we can isolate those parts of the design for study.

Providing Computation and Data at the Tactical Edge

The SEI developed KD-Cloudlet, a software solution that enables the quick deployment of tactical cloudlets—forward-deployed, discoverable, virtual-machine-based cloudlets that can be hosted on vehicles or other platforms and provide secure computation offload and data staging capabilities for soldiers in the field. KD-Cloudlet is available on GitHub.

Photo: Spc. Phillip McTaggart

Software Capabilities for the Warfighter



Principal Investigators

Josh Hammerstein
Research Team Lead,
Cyber Workforce Development

Jeffrey Mattson
Deputy Director, Cyber
Workforce Development

Cyber Affordance Visualization in Augmented Reality

There are many opportunities for front-line soldiers to use cyber tactics to help them achieve their missions. For example, a soldier on a reconnaissance mission who enters a potentially hostile or dangerous space, such as a storefront in enemy territory, might be able to gain access to an open wireless access point in the area or exploit vulnerabilities in the building's alarm-communication system. Soldiers can expand their arsenal through greater awareness of specific lethal and non-lethal cyber tactics available to them.

This work is intended to help front-line soldiers visualize cyber-warfare opportunities in their natural surroundings by using augmented reality to display virtual content to the soldier, in the form of 3D graphics aligned and overlaid on the real world. The information that a soldier sees identifies possible actions with regard to objects in the field of vision, such as downloading electronic information from a computer, exploiting a network vulnerability in a router, or disabling an alarm system. The goal of the project is to integrate cyber effects into tactical decision-making for soldiers.

In Context

This FY2017 research

- builds on our prior technical work to develop and deliver force-multiplying solutions such as human-computer decision-making and automated cyber readiness evaluation
- aligns with our sponsored technical work regarding platforms and content to train cyber mission force teams
- is related to our Line-funded research to develop algorithms to improve video analysis

Cyber Affordance Visualization in Augmented Reality

Cyberspace is unique in that it is intertwined with the other physical (air, land, sea, and space) warfighting domains. By providing soldiers with a greater awareness of specific lethal and non-lethal cyber tactics available to them, we can expand their arsenal and provide more options for ensuring mission success.

CAVIAR works through a head-worn device that displays virtual content to the soldier in the form of 3D graphics that are aligned and overlaid on the real world. The information identifies specific actions that the soldier could take related to objects in the soldier's field of vision such as download electronic information from a computer, exploit a network vulnerability in a router, disable an alarm system, etc.

Internet of Things Devices

Cyber Personas

Wireless Access Points

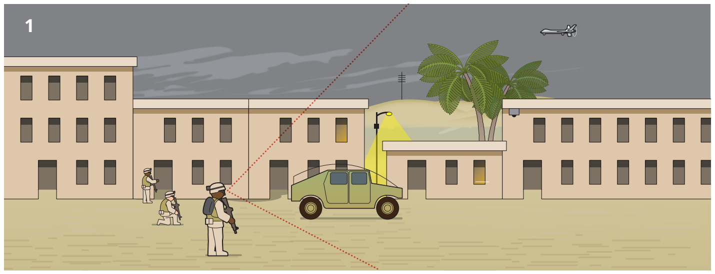
Consumer Electronics

RF Devices

SCADA/Industrial Control Systems

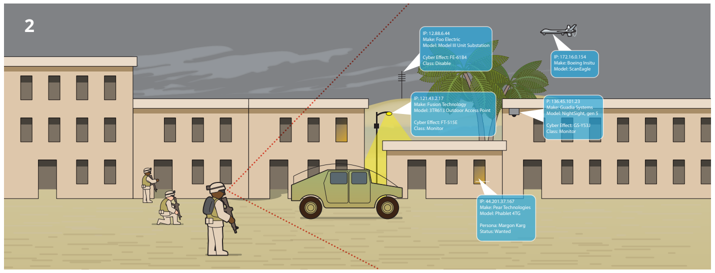
Affordances are opportunities presented by physical objects at the interface between individuals and the world. They are potential cyber actions that soldiers could take in their immediate geographical surroundings.

Natural View



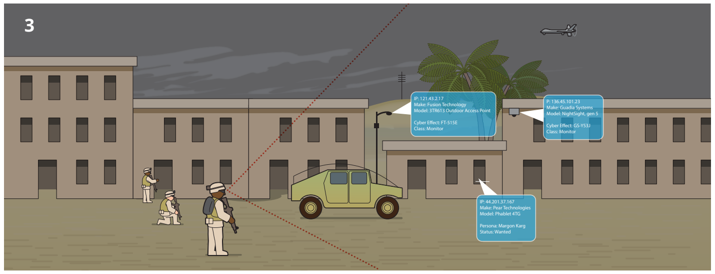
Soldiers navigate through an urban area in search of a person of interest. However, they are unaware of the cyber terrain in their vicinity.

View with Cyber Affordances



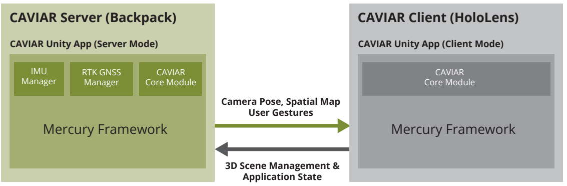
Cyber affordances make the soldiers aware of the cyber terrain in their immediate surroundings and of the potential cyber effects that can be deployed. The cyber affordances show that the person of interest is located in one of the buildings and that there is a power substation nearby.

Call For Cyber Effects



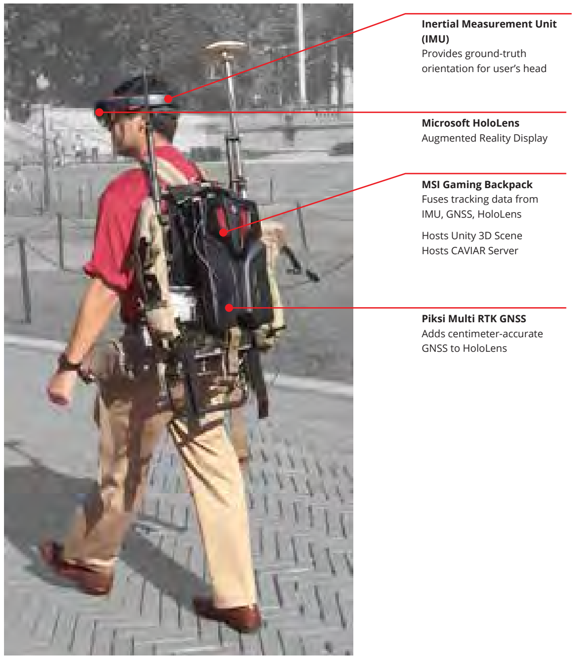
The soliders call in a cyber effect against the power substation to disable lighting in the area so that they can enter the building under the cover of darkness.

Software Architecture



The CAVIAR software architecture consists of both a client and server. The purpose of the server is to fuse tracking data from GNSS, the IMU and other data sources and to prepare scenes for the surrounding area. The client loads the scenes prepared by the server and displays it on the CAVIAR headset.

Hardware



Inertial Measurement Unit (IMU)
Provides ground-truth orientation for user's head

Microsoft HoloLens
Augmented Reality Display

MSI Gaming Backpack
Fuses tracking data from IMU, GNSS, HoloLens

Hosts Unity 3D Scene
Hosts CAVIAR Server

Piksi Multi RTK GNSS
Adds centimeter-accurate GNSS to HoloLens

Outdoor test of CAVIAR backpack and visor

Principal Investigator

Edwin Morris
Deputy Technical Director,
Critical System Capabilities

Events, Relationships, and Script Learning for Situational Awareness

Intelligence analysts working on new tasks have difficulty extracting meaningful information and recognizing patterns from high volumes of open and closed source textual data. Their efforts suffer from the absence of automated methods that reliably recognize actors, activities, and objects comprising an event or sequence of events (i.e., scripts). Further, the state-of-the-practice lacks a means to transfer knowledge learned from previously identified situations to new situations.

In the project, we address those deficiencies to provide first steps towards a system that recognizes events, sequences of events, instances of scripts and that supports transfer of this information to new situations.

In Context

This FY2017 project

- builds on prior Line-funded research including Structural Multitask Transfer Learning for Improved Situational Awareness
- contributes to FY2018 Line-funded research to search and summarize video
- draws from sponsored technical work engagements to reduce technical risk for DoD programs and agencies
- relies on our collaboration with CMU researchers and technologies for managing information flow in small units and rapid analysis of streaming data

Events, Relationships, and Script Learning

A huge volume of textual data is produced every day by traditional and social media. It is not possible to keep track of the many events that are mentioned, nor to recognize known patterns of events. This project uses and develops natural language processing (machine learning) techniques to automatically extract events from text. In future work we hope to relate these events to known patterns called scripts.

For this work, we pursued 3 strategies for extracting events from textual data:

1. Event extraction from sentences
2. Macro-event extraction from documents
3. Minimally supervised extraction of events from Twitter

Trigger Identification:

Bi-directional LSTM + CRF

- Concatenate character embeddings with pre-trained word embeddings to get vectors representing each word
- Run a bi-LSTM over the sequence of word vectors to obtain the two hidden states
- Use a CRF to find the sequence with the highest probability
- Comparison of state-of-the-art JointEventEntity strategy with our Bi-directional LTSM + CRF strategy

Results	P	R	F1
State-of-the-art	61.4	71.0	65.9
Our strategy	66.7	66.0	**66.4**

Trigger Replicated:

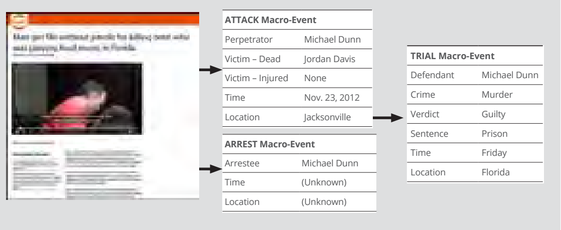
- Train SVMs to label triggers as belonging to one of 33 subtypes (e.g., attack, sentence, convict, etc.)
- However, with perfect trigger identification, F1 for trigger classification is 80.0.

Results	P	R	F1
JointEventEntity (replicated)	61.5	71.2	66.0
SVM	81.6	54.5	65.3

> Bottom line: Neither technique should be trusted to perform accurate trigger identification

Macro-Event Extraction

Document-level analysis – extract only the most important events in a text



ATTACK Macro-Event	
Perpetrator	Michael Dunn
Victim – Dead	Jordan Davis
Victim – Injured	None
Time	Nov. 23, 2012
Location	Jacksonville

ARREST Macro-Event	
Arrestee	Michael Dunn
Time	(Unknown)
Location	(Unknown)

TRIAL Macro-Event	
Defendant	Michael Dunn
Crime	Murder
Verdict	Guilty
Sentence	Prison
Time	Friday
Location	Florida

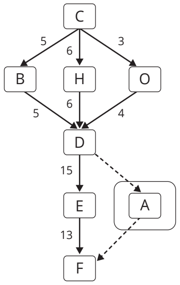
We are developing two novel ML-based algorithms for solving this problem

- A structured prediction model based on Learning to Search
- A deep neural network based on machine comprehension with no reliance on target domain training data

Preliminary results on the attack and elections domains show significantly improved performance against baseline methods

Currently gathering annotated data via Mechanical Turk

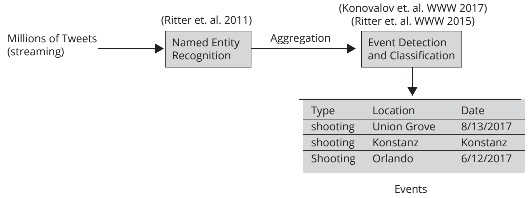
Next Step: Recognize and create scripts



- Better representation of the world
- Finer-grain event representation (actor, location, etc.)
- Probability distributions over the possible arguments
- A refined similarity metric that can reflect the new representation
- More robust script manipulation
- Better script addition (avoiding adding rare instances, etc.)
- Splitting / pruning existing scripts

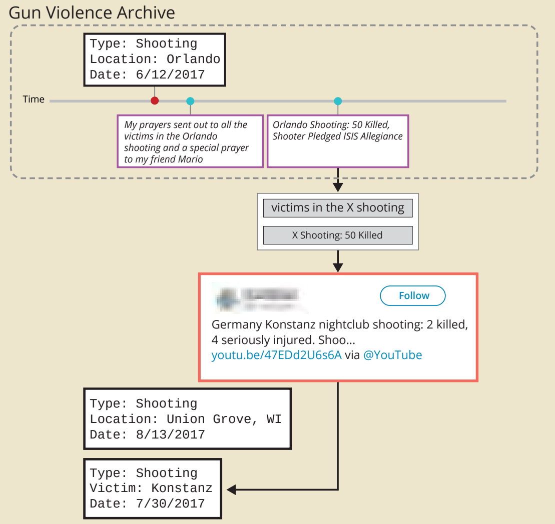
Use of macro-event knowledge for inferring constraints

Learning to Extract Events Minimal Supervision


(Ritter et. al. 2011)
(Konovalov et. al. WWW 2017)
(Ritter et. al. WWW 2015)
Millions of Tweets (streaming)
Named Entity Recognition
Aggregation
Event Detection and Classification

Type	Location	Date
shooting	Union Grove	8/13/2017
shooting	Konstanz	Konstanz
Shooting	Orlando	6/12/2017

Events



```
Type: Shooting
Location: Orlando
Date: 6/12/2017
```

My prayers sent out to all the victims in the Orlando shooting and a special prayer to my friend Mario

Orlando Shooting: 50 Killed, Shooter Pledged ISIS Allegiance

Germany Konstanz nightclub shooting: 2 killed, 4 seriously injured. Shoo... youtu.be/47EDd2U6s6A via @YouTube

```
Type: Shooting
Location: Union Grove, WI
Date: 8/13/2017
```

```
Type: Shooting
Victim: Konstanz
Date: 7/30/2017
```

Principal Investigator

Dr. Will Klieber
Software Security Engineer

Inference of Memory Bounds

Experience from CMU SEI CERT Division and DoD source code analysis labs shows that most software contains numerous vulnerabilities, often arising from common coding errors. Automated code repair cuts the cost of addressing software vulnerabilities—making more effective use of existing human programming resources, reducing a system's attack surface, and improving system resilience.

This work is developing an algorithm to automatically infer the bounds of memory regions intended to be accessible via specific pointer variables and repair certain memory-related defects in software. The results of the work will help the DoD improve

- software assurance of existing source code at a fraction of the cost of a manual inspection and repair
- evaluation of high-risk legacy binary software in its sustainment centers, which cannot be evaluated or fixed with existing tools
- analysis of malware code that would substantially improve the efficiency of DoD and National Security Agency (NSA) malware analysts

In Context

This FY2017 project

- extends our prior Line-funded research in automated repair of code for integer overflow and incorrect use of security application programming interfaces (APIs)
- contributes to FY2018 Line-funded research to ensure memory safety and automate executable program transformation
- aligns with our related technical work includes automated malware analysis advancements based on the Pharos static binary analysis framework, vulnerability discovery, and code diversification to avoid detection of vulnerabilities by adversaries

Inference of Memory Bounds

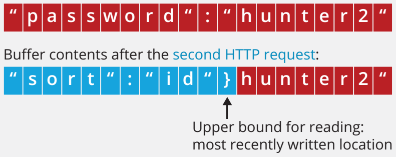
Invalid memory accesses are one of the most prevalent and most serious software vulnerabilities. This project aims to detect and repair not only out-of-bounds WRITEs, but also out-of-bounds READs, which are a relatively newer problem that can leak highly sensitive information.

A prime example of out-of-bounds READs is the OpenSSL HeartBleed vulnerability, which could be used to compromised the SSL private keys of two thirds of all websites. This type of vulnerability is unaffected by mitigations such as ASLR and DEP.

In general, for a re-usable buffer with stale data, READs should be bounded to the valid portion of the buffer. This type of problem affects even memory-safe languages such as Java. For example, the Jetty web server leaked passwords and any other data contained in a previous HTTP request.

Example: Re-used buffer with state data

Buffer contents after the first HTTP request:



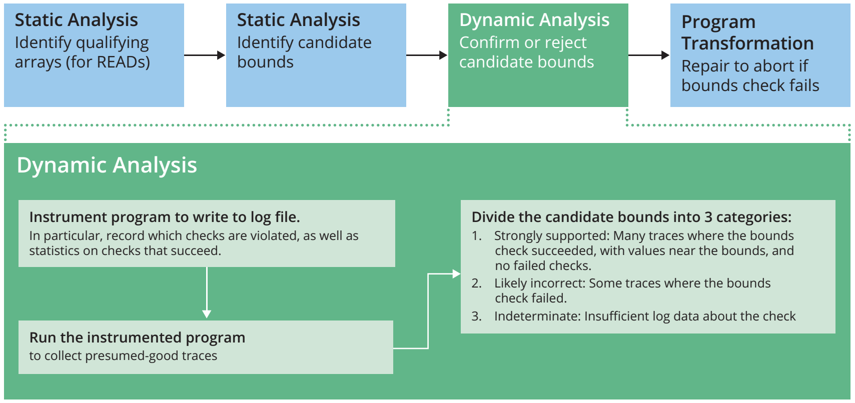
“ p a s s w o r d “ : “ h u n t e r 2 “

Buffer contents after the second HTTP request:

“ s o r t “ : “ i d “ } h u n t e r 2 “

Upper bound for reading:
most recently written location

This project is also useful for a second problem: decompilation of binaries. The relations between reconstructed fields is usually is left for the human analyst to manually investigate. We will try to reconstruct information of the form “[*n*, *m*] is bounds of pointer *p*”.



Static Analysis
Identify qualifying arrays (for READs)

Static Analysis
Identify candidate bounds

Dynamic Analysis
Confirm or reject candidate bounds

Program Transformation
Repair to abort if bounds check fails

Dynamic Analysis

Instrument program to write to log file.
In particular, record which checks are violated, as well as statistics on checks that succeed.

Run the instrumented program
to collect presumed-good traces

Divide the candidate bounds into 3 categories:

1. Strongly supported: Many traces where the bounds check succeeded, with values near the bounds, and no failed checks.
2. Likely incorrect: Some traces where the bounds check failed.
3. Indeterminate: Insufficient log data about the check

Strategies to propose candidate bounds:

1. (For reads) The most recently written position in the buffer.
2. Bounds of region allocated by malloc.
3. Pointer arithmetic with constant offset (e.g., field of a struct)—mainly for use in decompilation.
4. Analysis of memory accesses within loops and limits of the loop.
   - Exact if the number of iterations is known at start of loop.
   - Only a candidate bound if it is possible to break out of the loop early.
5. Invariants for structs (by typename or by allocation site).
   - Suppose that we discover that, in most of the program, one field of a struct supplies the bounds of another field of the struct.
   - Then we guess that this is an invariant and violations of it are errors.
6. If in most callsites of a function foo(int n, char *p, ...), the bounds on p is the closed interval [p, p+n-1], then propose that in the other callsites, the same bounds should apply.

How do we determine which arrays should be subject to the analysis for READs outside the valide portion of an arrary?

- We consider an array to be a *qualifying array* if every write to the array is at either index 0 or at the successor of the last written position.

How do we identify the valid portion of the array?

- Heuristic: It is from the start of the array up to and including the last written element of the array.

How often do qualifying arrays occur in real-world programs?

- Imprecision in static analysis might cause false negatives.
- To establish ground truth, we do a separate dynamic analysis (next column).

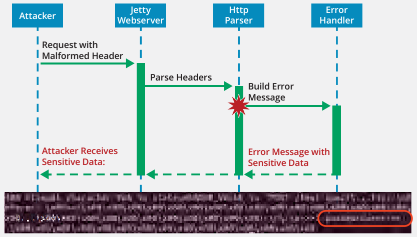
Stand-alone dynamic analysis for out-of-bounds READS:

We have written a Java agent to:

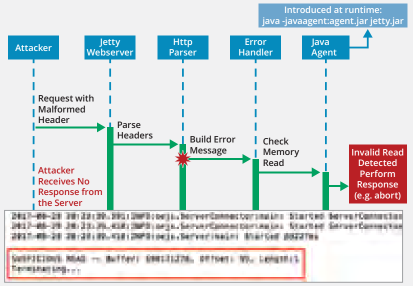
- Record the allocation site and the last written position (LWP) of each allocated ByteBuffer.
- Check whether each write to the ByteBuffer is consistent with definition of qualifying array.
- If all the writes have been qualifying, we flag any reads beyond LWP.

- Note that this dynamic analysis is different than the dynamic validation of statically-inferred candidate bounds.
- With this tool, we dynamically patch Jetty to prevent leakage of sensitive information, as shown below.

Unpatched Jetty


Attacker
Jetty Webserver
Http Parser
Error Handler
Request with Malformed Header
Parse Headers
Build Error Message
Attacker Receives Sensitive Data:
Error Message with Sensitive Data

Jetty with runtime bounds enforcement


Introduced at runtime:
java -javaagent:agent.jar jetty.jar
Attacker
Jetty Webserver
Http Parser
Error Handler
Java Agent
Request with Malformed Header
Parse Headers
Build Error Message
Check Memory Read
Attacker Receives No Response from the Server
Invalid Read Detected Perform Response (e.g. abort)

Principal Investigator

Dr. Scott McMillan
Senior Research Scientist

Measuring Performance of Big Learning Workloads

Machine learning (ML) is becoming a primary mechanism for extracting information from data. Indeed, each year, academic and research organizations publish more than 1,000 technical papers on ML, few (if any) of which provide enough detail to reproduce the results. The state of reporting on ML research slows adoption by DoD of advances needed to effectively use the surging volume of Big Data from network logs, internet activities, and sensory advancements.

In this work, we are building a performance measurement workbench with tools to provide sound, reproducible ways to measure and report performance of large-scale ML platforms designed to operate on Big Data workloads.

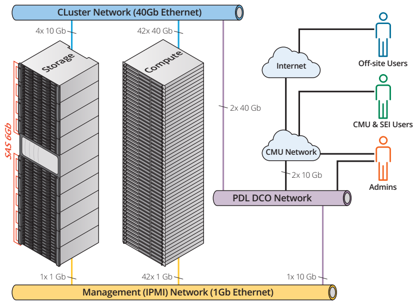
In Context

This FY2016–17 project

- builds on our prior Line-funded research in Graph Algorithms on Future Architectures and our related technical work in Automated Code Generation for Future-Compatible High-Performance Graph Libraries (see page 6)
- aligns with our other data-intensive projects such as the Predictive Analytics Workshop, through the use of graph algorithms as an enabling technology

Measuring Performance of Big Learning Workloads

Big Learning platforms—large scale hardware and software systems designed to perform large-scale machine learning (ML) workloads on big data—are extremely complex and lack consistent and sufficient reporting of performance metrics. These difficulties can slow DoD adoption of new advances in machine learning. A key obstacle to overcome is in the collection and analysis of these metrics.



Component	Per Node	Totals (42 nodes)
CPU, Xeon E5, 2 GHz	16 cores, 32 threads	672 cores, 1344 threads
GPU, Titan X	3072 cores, 12 GB RAM	129K cores
RAM, DDR4	64 GB	2688 GB
NVMe storage	400 GB	16.8 TB
HDD storage	8 TB	336 TB
Network	40 GB	
Persistent storage		432 TB

The Big Learning cluster consists of 42 compute nodes each with CPU and GPU processing units, complex storage system and fast networking. It supports research in the development of parallel ML computing frameworks as well as development and testing of large-scale metrics collection systems.

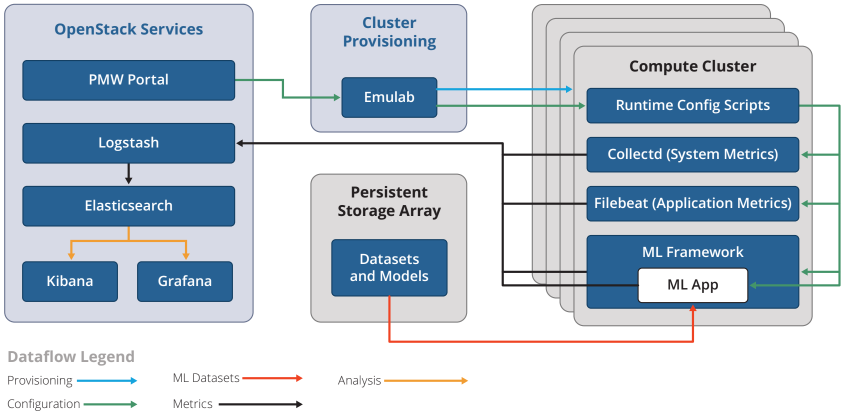
Big Learning Cluster.
The Big Learning cluster, located at the Carnegie Mellon University Parallel Data Lab, was designed to support research in evaluating existing Big Learning platforms and developing new platforms for a wide variety of large-scale ML applications. It is a distributed cluster with CPU and GPU processors, a complex storage hierarchy, a high-bandwidth/low latency network for communication, and a large persistent store.

Performance Measurement Workbench.
We developed the Performance Measurement Workbench (PMW) to collect metrics about the performance of hardware components (CPU, memory, disk, network, etc.) and the performance of the ML algorithms (accuracy, convergence, iteration times, etc.) that run on the Big Learning cluster.

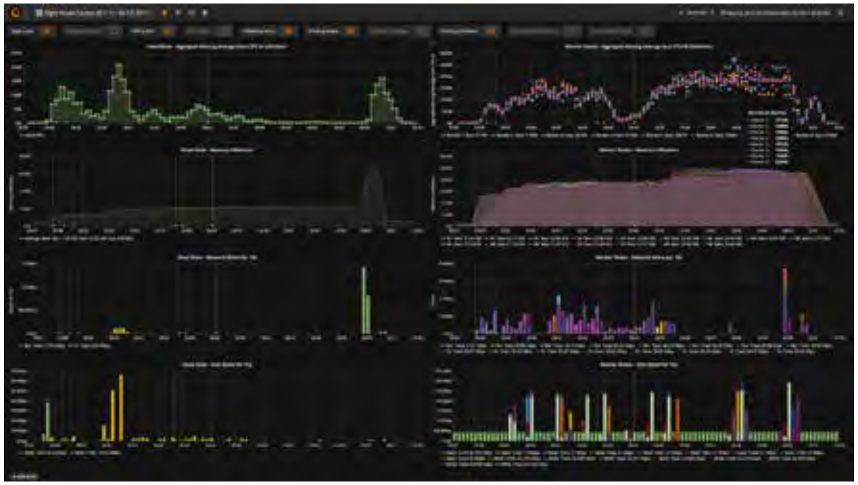
Goal: Ease of Use.
PMW provides a simple, web-based portal for researchers and users to configure and submit jobs, collect and store metrics, and analyze data both during computation and in post mortem.

Goal: Reproducibility.
PMW not only collects the performance metrics for each job, but it can also collect and store the configuration of the operating system, the ML platform, and the algorithm being run. With this information reproducible experiments are achievable.

With the Performance Measurement Workbench—combining a few open-source software packages (especially Elastic Stack)—we have demonstrated how consistent and complete measurement metrics for complex Big Learning systems can be collected. PMW has the added benefit of supporting collection of configuration aimed at reproducibility of results.


OpenStack Services
PMW Portal
Logstash
Elasticsearch
Kibana
Grafana
Cluster Provisioning
Emulab
Persistent Storage Array
Datasets and Models
Compute Cluster
Runtime Config Scripts
Collectd (System Metrics)
Filebeat (Application Metrics)
ML Framework
ML App
Dataflow Legend
Provisioning
ML Datasets
Analysis
Configuration
Metrics

Performance Measurement Workbench system architecture. PMW provides a simple, web-based portal for submitting jobs, operating system images with collection tools preconfigured, and persistent database query and visualization services using the open-source Elastic Stack.



PMW's dashboard display using Grafana integrates with Elastic Stack to achieve complex visualizations. This example shows system metrics for a Spark MLlib job that uses one "head" node (displayed on the left) and eight worker nodes to perform a logistic regression algorithm (displayed on the right).

Examining Trust in Autonomous Systems

Trust is a systemic challenge to adoption of autonomous systems in the U.S. Department of Defense (DoD). In exploring trust, CMU and SEI researchers draw from artificial intelligence, machine learning, software and systems engineering, robotics, and human-machine interactions. They focus on developing formal methods for verification, human-system and system-system teamwork, and issues of complexity and scalability.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

DM17-0719

About Us

The Software Engineering Institute (SEI) is a not-for-profit Federally Funded Research and Development Center (FFRDC) at Carnegie Mellon University, specifically established by the U.S. Department of Defense (DoD) to focus on software and cybersecurity. As an FFRDC, the SEI fills voids where in-house and private sector research and development centers are unable to meet DoD core technology needs.

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412.268.5800 | 888.201.4479
Web: sei.cmu.edu | cert.org
Email: info@sei.cmu.edu