Carnegie Mellon University
Software Engineering Institute

RESEARCH REVIEW

2019 FUTURE REACH

FUTURE

The future of software is evolving faster than ever. As the most critical component of all systems, we rely on software to secure our sensitive data, drive our vehicles safely, control our critical infrastructures, and define the capabilities that keep the U.S. Department of Defense (DoD) ahead of our adversaries.

The Carnegie Mellon University Software Engineering Institute (CMU SEI) vision is a DoD able to gain and sustain advantage over adversaries through software. To translate that vision into reality, CMU SEI integrates research in software, and cyber to provide solutions for DoD capabilities through the acquisition, integration, and delivery of software. We recognize the DoD faces enduring challenges as the need for software innovation and cybersecurity evolves and intensifies. The 2018 National Defense Strategy (NDS), 2018 National Security Strategy, and a 2018 report on software from the Defense Innovation Board make clear that the DoD needs its software-enabled systems to

- bring capabilities that make new missions possible or improve the likelihood of success of existing ones
- be timely so that the cadence of acquisition, delivery, and fielding is responsive to and anticipatory of the operational tempo of DoD warfighters and that the DoD is able to field these new software-enabled systems and their upgrades faster than our adversaries
- be trustworthy in construction and implementation and resilient in the face of operational uncertainties including known and yet unseen adversary capabilities
- be affordable such that the cost of acquisition and operations, despite increased capability, is reduced and predictable and provides a cost advantage over our adversaries

In this booklet and during our research review, you will learn how we build on our past experience to reach into the future with new capabilities that lead the software engineering, cyber, and AI communities.

The theme for this year's Research Review is Future Reach. Rather than focus exclusively on a recap of our FY19 research, we seek to place this work in the context of the future needs of the DoD. This approach reflects our commitment to prioritize research that has a significant impact on the software used in future DoD acquisition and operations. Our research and collaboration with cutting-edge researchers at CMU assures that we are applying the most effective results to support DoD mission success.

Since its inception, the SEI has evolved to anticipate and address the most critical software problems facing the DoD. We execute our research across multiple disciplines to bring a wide spectrum of technical know-how to our projects. We strive to create an exciting, innovative, and collaborative environment that anticipates a world we will work together to create. We are actively seeking collaborators and partners in our work going forward. I encourage you to reach out to our researchers to discuss your critical software needs and experiences so we can work together to reach the future. We stand ready to work with you.

TOM LONGSTAFF
Chief Technology Officer
Carnegie Mellon University Software Engineering Institute

Foreground

DR. TOM LONGSTAFF, Chief Technology Officer, Carnegie Mellon University Software Engineering Institute

Background

MIKE DUDA, Senior Multimedia Designer, Carnegie Mellon University Software Engineering Institute

REACH

Contents

Impacting Today's State of the Practice in Software Engineering and Cybersecurity

SESSION 1

Rapid Construction of Accurate Automatic Alert Handling System

Problem

Static analysis alerts for security-related code flaws require too much manual effort to triage efficiently. **Organizations are reluctant to fully adopt automated alert classifier technology because of barriers, including high cost, lack of expertise, and shortage of labeled data.**

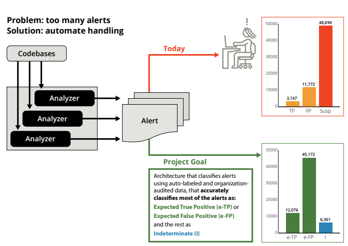
Solution

Develop an extensible architecture that supports classification and advanced prioritization, and builds on a novel test-suite-data method we developed.

- We developed a model and code intended to enable organizations to quickly start using classifiers and advanced prioritization by making API calls from their alert auditing tools.
- We implemented a prototype of the model.
- We developed adaptive heuristics for classifiers to adapt as they learn from test suite and natural program data.

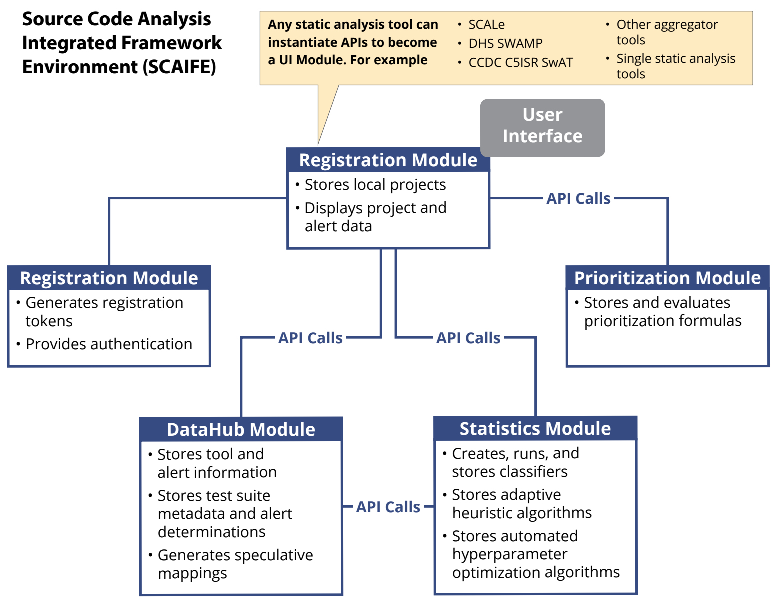
Approach

- Design the architecture.
- Develop an API definition.
- Implement a prototype system.
- Develop adaptive heuristics and test them with datasets that combine test-suite and real-world (e.g., DoD) data.
- Test the architecture and prototype with collaborators.



To **overcome cost** and **data barriers**, we prototyped a **modular architecture that enables** the rapid adoption of **automated classifiers for static analysis alerts.**

SCAIFE Architecture



FY19 Artifacts

Code and Test Results

- Beta SCAIFE prototype VM (v1, v2) released to collaborators (August & September 2019)
- API definitions (0.0.2-0.0.5) YAML publication (GitHub + SCAIFE VM)
- SCALe v3 and v4: tool released with new features for collaborators to generate data
- SCALe DevOps improvements for research transitionability
- SCALe v.r.4.*: released to collaborators with features for SCAIFE integration (August & September 2019)
- Code developed for prototype
- Adaptive heuristics

Publications

- SCAIFE API Definition and Prototype
  - Manual: How to Review & Test the Beta SCAIFE (v1, v2) VM (August & September 2019)
  - SEI blog post: An Application Programming Interface for Classifying and Prioritizing Static Analysis Alerts (July 2019)
  - SEI whitepaper: SCAIFE API Definition Beta Version 0.0.2 for Developers (June 2019)
  - SEI technical report: Integration of Automated Static Analysis Alert Classification and Prioritization with Auditing Tools (May 2019)
  - SEI blog post: SCALe v3: Automated Classification and Advanced Prioritization of Static Analysis Alerts (December 2018)
  - SwACon paper: Introduction to Source Code Analysis Laboratory (SCALe) (November 2018)
  - SEI webinar: How can I use new features in the CERT SCALe tool to improve how my team audits static analysis alerts? (November 2018)
- Classifier Development Research
  - Presentation: Automating Static Analysis Alert Handling with Machine Learning: 2016-2018 (October 2018)
  - Four in-progress papers addressing precise mapping, architecture for rapid alert classification, test suites for classifier training data, and API development

Project members developed (1) an architecture, (2) an API definition, and (3) a prototype system for static analysis alert classification and advanced alert prioritization.

Principal Investigator

DR. LORI FLYNN
Software Security Engineer
Carnegie Mellon University
Software Engineering Institute

On Right

EBONIE MCNEIL, Software Security Analyst, Carnegie Mellon University Software Engineering Institute

Rapid Construction of Accurate Automatic Alert Handling System

Static analysis (SA) alerts about code flaws require costly human effort to adjudicate (i.e., determine if they are true or false) and repair them. SA tools produce many false positives and often generate many more alerts than can be adjudicated manually. These tools generate roughly 40 alerts per 1,000 lines of code [Heckman 2008]. Each alert requires approximately 117 seconds to audit [Pugh 2010], and many of these alerts are false positives [Beller 2016, Delaitre 2015].

Previous research developed accurate SA alert classifiers (e.g., 85% accuracy in a 2008 report [Ruthruff 2008], 91% accuracy in our prior results [Flynn 2016], and various rates in many other studies [Heckman 2011]). However, DoD organizations do not use them because they lack a reference architecture that allows them to be rapidly and automatically created [Flynn 2017].

In this project, we are developing a reference architecture and prototype that enables rapid deployment of a method intended to automatically, accurately, and adaptively classify and prioritize alerts. Our research on reference architectures will enable DoD organizations to become more efficient by (1) reducing their adjudication workload by at least 60% and (2) focusing their manual validation efforts on truly dangerous code flaws in the remaining 40% of alerts.

IN CONTEXT: THIS FY2018–19 PROJECT

- builds on the techniques and tools we developed in prior DoD line-funded research to prioritize vulnerabilities from static analysis using classification models (and their rapid expansion) to prioritize static analysis alerts for C
- aligns with the CMU SEI technical objective to make software trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties, including known and yet unseen adversary capabilities

Integrating Safety and Security Engineering for Mission-Critical Systems (ISSE-MCS)

Critical systems must be both safe from inadvertent harm and secure from malicious actors. However, safety and security practices have historically evolved in isolation. Safety-critical systems, such as aircraft and medical devices, were long considered standalone systems without security concerns. Security communities, on the other hand, have focused on information security and cybersecurity. Mechanisms such as partitioning, redundancy, and encryption are often deployed solely from a safety or security perspective, resulting in over-provisioning and conflicts between mechanisms. Despite the recognition that this disconnect is harmful [Friedberg 2017], there is limited understanding of the interactions between safety and security.

To combat this lack of understanding, we are developing an integrated safety and security engineering approach based on system theory and supported by an AADL-based workbench. This approach

- unifies safety and security analysis through a formalized taxonomy that is used to drive system verification via fault-injection and simulation
- provides a design framework to combine safety and security mechanisms into a more robust and resilient system architecture through continuous analytic verification
- ensures traceability by linking machine-readable requirements to the tests that verify them and the system elements that implement them

In the Joint Multi-Role Rotorcraft (JMR) program, contractor teams are piloting Architecture-Centric Virtual Integration Practice (ACVIP) as a key technology on a mission-critical system architecture. Our ongoing partnership with JMR provides an excellent transition pathway for our research results and influences the Army's Future Vertical Lift (FVL) program.

IN CONTEXT: THIS FY2018–20 PROJECT

- extends the safety analysis capability present in OSATE with conditional fault source and propagation capabilities and the requirement specification capability in ALISA, the incremental assurance component of OSATE, to support specification of reusable requirements and verification plans
- aligns with the SEI's technical objective to make software trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties, including known and yet unseen adversary capabilities

Principal Investigator

DR. SAM PROCTER
Senior Architecture Researcher
Carnegie Mellon University Software Engineering Institute

Integrating Safety and Security Engineering for Mission-Critical Systems

Overview

We're in the second year of a three-year project that aims to make systems safer and more secure. This project consists of four efforts, all of which utilize the Architecture Analysis and Design Language (AADL), an SEI-created, internationally standardized language for designing critical systems.

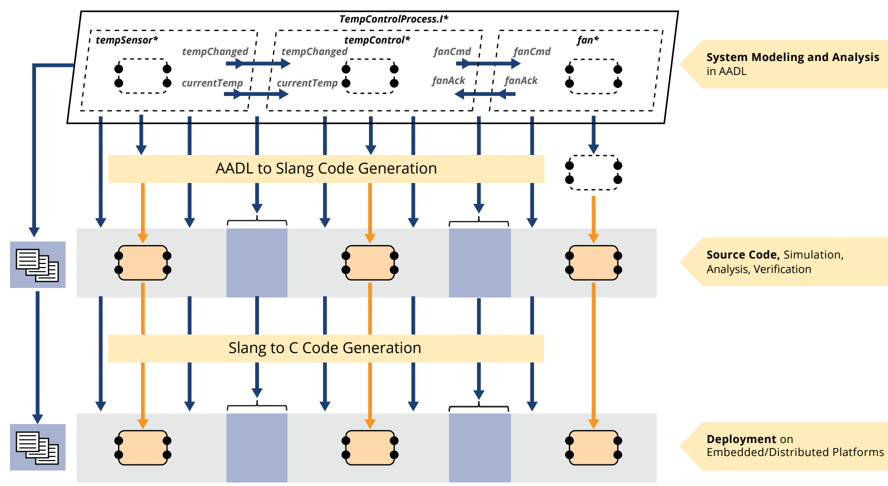
Security Annex & Patterns

There are many ways to design a system, and subtle changes can have large impacts on security and safety. One effort is looking at creating patterns—in AADL—that let our tooling automatically check for known security issues and offer suggestions for improvement.

ASAP

Performing a hazard analysis is a common way of examining a system for safety or security issues. This effort integrates a number of sources of system information—requirements, error behavior, Slang & HAMR, and more—into a set of dynamic reports. The Architecture-Supported Audit Processor (ASAP) will allow system analysts to query interesting portions of a system's architecture interactively, rather than read only what an analysis format specifies.

We're making it easier to **specify, design,** and **assure** critical systems that are safer and more secure.



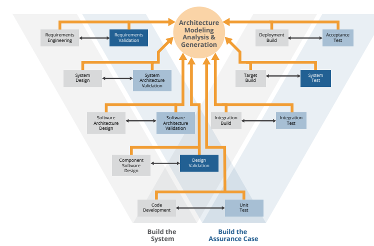
Slang and HAMR Integrate Verification with Code Generation. (Figure adapted from John Hatcliff).

Slang & HAMR

Slang, a safety-critical subset of Scala, and HAMR (High-Assurance Modeling and Rapid engineering for embedded systems) are in development by Kansas State University. These technologies support both system verification—of things like reachability properties and contract violations—and code generation to languages like C.

ALISA2

The Architecture-Led Incremental System Assurance (ALISA) project created a suite of languages and tools that let system designers specify requirements and verification activities in a machine-readable format that can be directly linked to AADL specifications. In this effort, we're updating ALISA to support the integration of other tools so system designers can use one unified interface.



Safety and Security Across the System Development Lifecycle

Recovering Meaningful Variable Names in Decompiled Code

Introduction

Conventional wisdom tells us that when a compiler transforms a program from source code to an executable file, some information is lost and cannot be recovered. For example, variable names are not included in a compiled executable, and we often assume they are lost. Although state-of-the-art decompilers can recover the presence of variables, they make no attempt to recover their original names. Instead, they name the variables "v1," "v2," and so on. Renaming the variables is unfortunate because, as several studies have shown, programmers carefully select variable names to make the program easier to understand.

In this project, we showed that the conventional wisdom that variable names cannot be recovered is wrong. Specifically, we showed that variable names can largely be predicted based on the context of code in which they are used and accessed. We trained a neural network to predict variable names on a large corpus of C source code that we collected from GitHub.

Corpus

To generate our corpus, we scraped GitHub for projects written in C. We then automatically built 164,632 binaries from these project and extracted 1,259,935 functions. For each function, we generated a corpus entry that consisted of the original source code with placeholder variables, as shown in the code figure to the right.

Each corpus entry also included a mapping from a placeholder variable to the original identifier in the source code and the decompiler's identifier.

We can make **exact** predictions for **74.3%** of variable names in decompiled executable code by training a neural network on a large corpus of C source code from GitHub.

```
void *file_mmap(int v1|fd|fd, int v2|size|size)
{
void *ptr|ret|buf;
ptr|ret|buf = mmap (0, v2|size|size, 1, 2, v1|fd|fd, 0);
if (ptr|ret|buf == (void *) -1)
{ perror ("mmap"); exit(1); }

return ptr|ret|buf;
}
```

Key
Decompiled Original Recovered

Results

Experiment	Accuracy
Overall	74.3
Function in Training	85.5
Function not in Training	35.3

When evaluating a solution based on machine learning such as ours, it is important to consider the construction of the training and testing sets. Each binary was randomly assigned to either the training or testing set. As in real reverse-engineering scenarios, library functions may be present in multiple binaries and may therefore be present in both the training and testing sets. To better understand the effect of the presence of library functions on our system, we partitioned our testing set into the set of functions that were also in the training set and those that were not in the training set. As shown in the table above, DIRE achieves 85.5% accuracy on functions it has been trained on, compared to 74.3% overall. For functions that it has not encountered in training, it yields 35.3% accuracy.

For more information, see: Jeremy Lacomis, Pengcheng Yin, Edward J. Schwartz, Miltiadis Allamanis, Claire Le Goues, Graham Neubig, and Bogdan Vasilescu. DIRE: A Neural Approach to Decompiled Identifier Renaming, Procedings of the 2019 IEEE/ACM International Conference on Automated Software Engineering.

Principal Investigator

CORY COHEN

Senior Member of the Technical Staff

Carnegie Mellon University
Software Engineering Institute

Recovering Meaningful Variable Names in Decompiled Code

Highly skilled Department of Defense (DoD) malware and vulnerability analysts currently spend significant amounts of time manually reading executable code to understand how it behaves when run. Understanding executable code is significantly more difficult than reading source code, since the compilation process removes much of the information in the original source code (e.g., control flow structure, type information, and variable names). Decompilers, which convert the program back into high-level code, can recover some of this information and some of the most important tools for analyzing executables when corresponding source code is not available.

Although modern decompilers can recover a great deal of information, such as control flow structure and types, they do not recover meaningful variable names. This is unfortunate because semantically meaningful variable names are known to increase code understandability. In this project, we propose the Decompiled Identifier Renaming Engine (DIRE), a novel probabilistic technique for variable name recovery that uses both lexical and structural information. We also present a technique for generating corpora suitable for training and evaluating models of decompiled code renaming, which we use to create a corpus of 164,632 unique x86-64 binaries generated from C projects mined from Github. Our results show that on this corpus DIRE can predict variable names identical to the names in the original source code up to 74.3% of the time.

IN CONTEXT: THIS FY2018–20 PROJECT

- extends DoD line-funded research and tool development for vulnerability and binary code analysis
- contributes to development and transition of Pharos binary code analysis framework
- aligns with the CMU SEI technical objective to make software trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties including known and yet unseen adversary capabilities

In collaboration with

ED SCHWARTZ

Research Scientist

Carnegie Mellon University
Software Engineering Institute

Causal Models for Software Cost Prediction & Control (SCOPE)

Cost estimation inaccuracy continues to be cited as a dominant factor in DoD cost overruns. Research has shown causal models are superior to traditional statistical models because, by identifying truly causal factors, proactive control of project and task outcomes is possible.

Until recently, we did not have a way to obtain or validate causal models from primarily observational data, a challenge shared across nearly all systems and software engineering research, where randomized control trials are nearly impossible. Yet, in search of good practice, systems and software engineering researchers and practitioners espouse various theories about how best to conduct system and software development and sustainment. The SCOPE project will apply causal modeling algorithms and tools [Spirtes 2010] to a large volume of project data to identify, measure, and test causality.

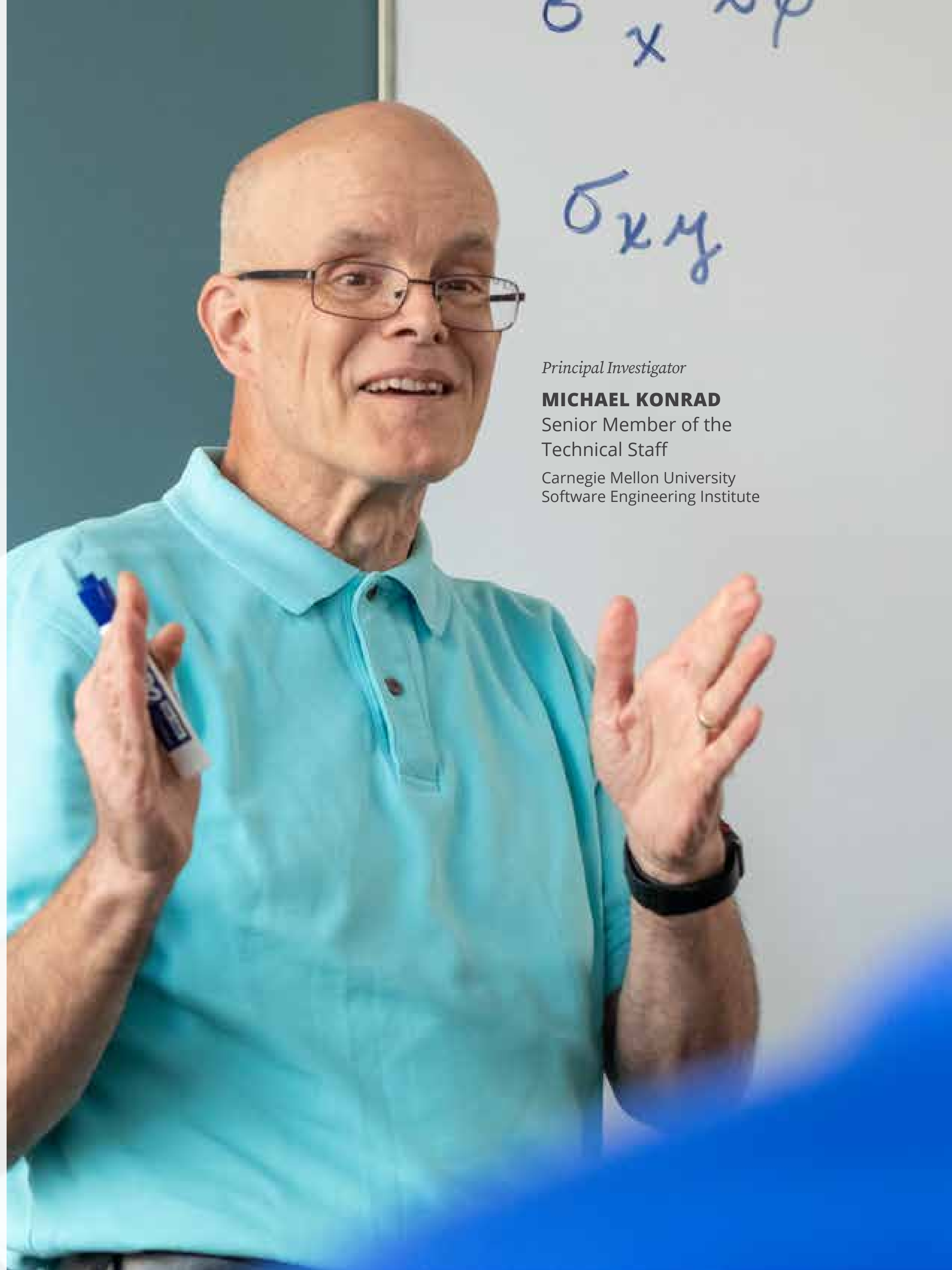
In this work, we expect to develop causal models, including structural equation models (SEMs) that provide a basis for

- calculating the effort, schedule, and quality results of software projects under different scenarios (e.g., traditional vs. agile)
- estimating the results of interventions applied to a project in response to a change in requirements (e.g., a change in mission) or to help bring it back on track toward achieving cost, schedule, and technical requirements

Thus, an immediate benefit of this work is the identification of causal factors that provide a basis for controlling program costs. A longer-term benefit is the use of causal models in negotiating software contracts, designing policy and incentives, and informing could/should cost and affordability efforts.

IN CONTEXT: THIS FY2018–20 PROJECT

- contributes to a longer-term research road map to build causal models for the software developer, software development team, organization, and acquirer
- aligns with the CMU SEI technical objective to make software affordable such that the cost of acquisition and operations, despite increased capability, is reduced and predictable



Principal Investigator

MICHAEL KONRAD
Senior Member of the Technical Staff

Carnegie Mellon University
Software Engineering Institute

Causal Models for Software Cost Prediction & Control (SCOPE)

Recent Results from Ongoing Studies

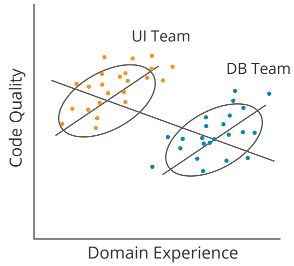
We are collaborating with other researchers to apply causal learning to learn how to control costs in software development and sustainment.

DoD Problem

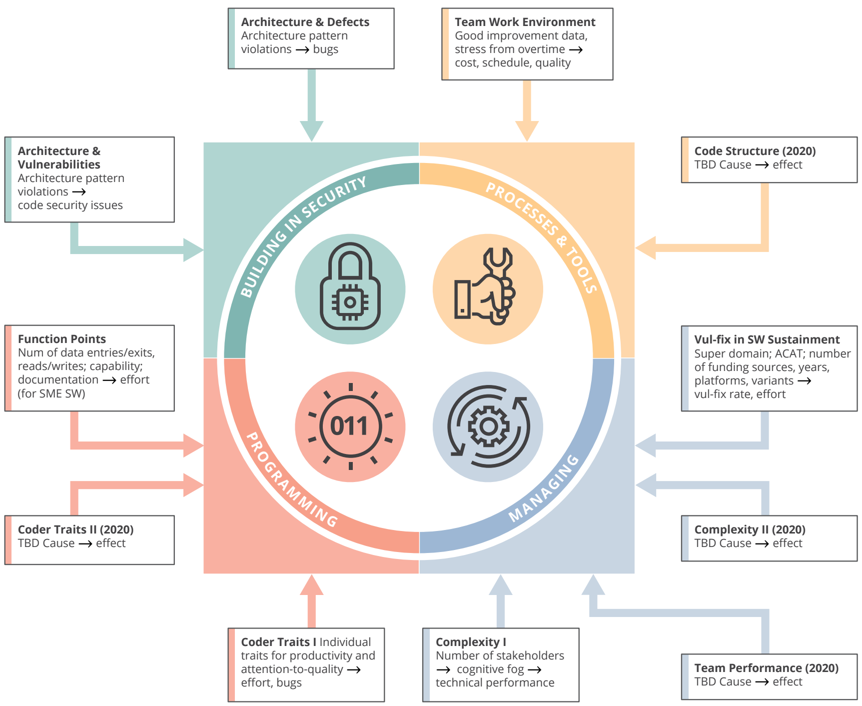
- DoD leadership needs to understand why software costs so much.
- DoD program offices need to know where to intervene to control software costs.

Why Causal Learning?

To reduce costs, the causes of an outcome (good or bad) need to be considered. Correlations are insufficient due to Simpson's Paradox. For example, in the figure below, if you did not segment your data by team (User Interface [UI] and Database [DB]), you might conclude that increasing domain experience reduces code quality (downward line); however, within each team, it's clear that the opposite is true (two upward lines). Causal learning identifies when factors such as team membership explain away (or mediate) correlations, and it works for much more complicated datasets too.



Reduce costs through **causal learning**.

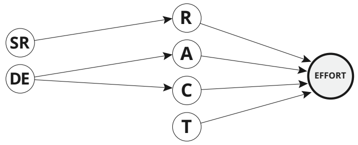


Our Solution

Working with collaborators, we will jointly apply causal learning to their datasets to establish key cause-effect relationships among project factors and outcomes.

Our collaborators include the University of Southern California, U.S. Army, and a static code analysis tool vendor.

For example, for *effort*, we might have this causal graph:



This graph tells us that increasing stakeholder reviews (SR) and domain experience (DE) improves the effectiveness of requirements, analysis, coding, and testing, thereby improving quality.

If the dataset is proprietary, the SEI trains the collaborator to perform causal searches on their own. The SEI then needs information only about what dataset and search parameters were used as well as the resulting causal graph.

Summary

Causal models offer better insight for program control than models based on correlation. Knowing which factors drive which program outcomes is essential to sustain the warfighter by providing higher quality, secure software in a timely and affordable manner.

Shaping the Next Generation of Practice

SESSION 2

KalKi: High Assurance Software-Defined IoT Security

The term “KalKi” is of Sanskrit origin, and is the name of an avatar of the god Vishnu, who is the destroyer of filth and malice and usherer of purity, truth and trust.

Problem
Despite the DoD’s current use of Internet of Things (IoT) devices in supervisory control and data acquisition (SCADA) systems, and its interest in using such devices in tactical systems, adoption of IoT has been slow mainly due to security concerns (e.g., reported vulnerabilities, untrusted supply chains).

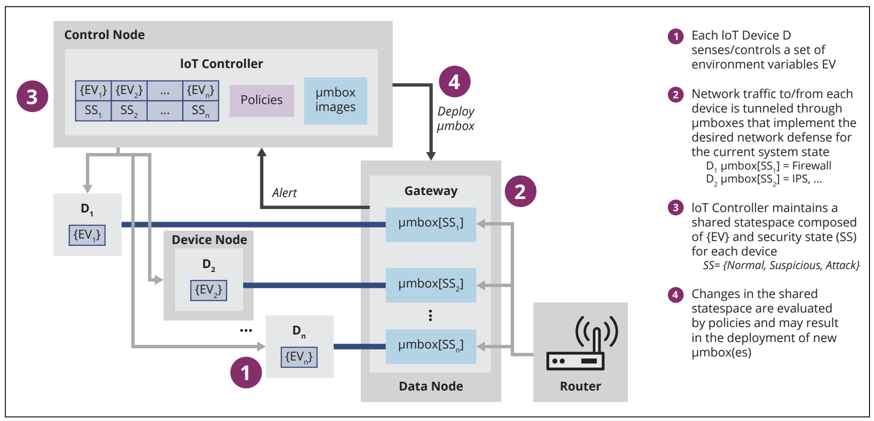
At the same time, the DoD recognizes the rapid pace at which the IoT commercial marketplace is evolving, and its urgency to embrace commodity technologies to match its adversaries.

Solution
Move part of security enforcement to the network to enable the integration of IoT devices into DoD systems, even if the IoT devices are not fully trusted or configurable, by creating an IoT security platform that is provably resilient to a collection of prescribed threats.

The “Software-Defined” Aspect
Use software-defined networking (SDN) and network function virtualization (NFV) to create a highly dynamic IoT security platform.

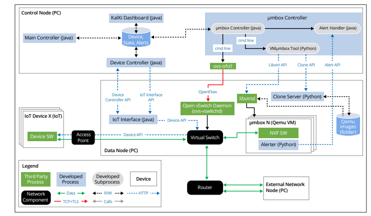
The “High Assurance” Aspect
Use überSpark (a framework for building secure software stacks) to incrementally develop and verify security properties of elements of the software-defined IoT security platform.

The KalKi IoT Security Platform **enables the integration of IoT devices** into DoD systems, even if the IoT devices are **not fully trusted** or configurable.



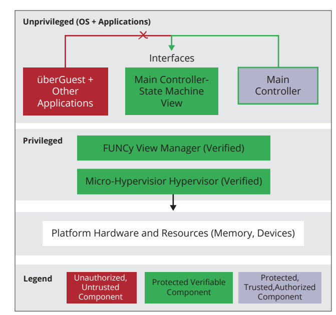
- **Sensitive areas of the system are protected via FUNCY Views – a novel, performant, isolated execution environment extension to überXMHF/überSpark**
  - **State machine** that controls the security state transitions for each IoT device in the Control Node
  - **Routing tables** and other sensitive data structures used by Open vSwitch (OVS) in the Data Node

Year 2 Highlights



End-to-end prototype of IoT Security Platform implemented and tested with different attack scenarios. Policies and μmboxes implemented for four representative devices.

Updated version of the KalKi Dashboard, which allows to fully configure and monitor the system.



überXMHF/überSpark extended to include überobject protections for sensitive areas of the Control node and Data node.

Principal Investigator

SEBASTIAN ECHEVERRIA
Member of the Technical Staff
Carnegie Mellon University
Software Engineering Institute

On Left

DR. GRACE LEWIS, Principal Researcher/ Tactical, AI-enabled Systems Initiative Lead, Carnegie Mellon University
Software Engineering Institute

Kalki: High Assurance Software-Defined IoT Security

Despite its use of Internet of Things (IoT) devices in supervisory control and data acquisition (SCADA) systems and its interest in using such devices in tactical systems, the DoD has been slow to adopt IoT. In particular, the DoD is reluctant to use commodity IoT devices, especially in tactical systems, because of untrusted supply chains and a growing amount of reported vulnerabilities in these devices. At the same time, DoD recognizes the rapid pace at which the IoT commercial marketplace is evolving and its urgency to embrace commodity technologies, to match its adversaries.

Our proposed solution moves part of security enforcement to the network to enable the integration of IoT devices into DoD systems, even if the IoT devices are not fully trusted or configurable, by creating an IoT security infrastructure that is provably resilient to a collection of prescribed threats. It uses

- software-defined networking (SDN) and network function virtualization (NFV) to create a highly dynamic IoT security framework
- überSpark (a framework for building secure software stacks) to incrementally develop and verify security properties of elements of the software-defined IoT security infrastructure [Vasudevan 2016]

IN CONTEXT: THIS FY2018–20 PROJECT

- builds on prior CMU SEI technical work in the mobile communication and computing needs of edge users and the authentication and authorization for IoT devices
- draws from our collaboration with CMU researchers and sponsored engagements to reduce risk through architecture analysis
- aligns with the CMU SEI technical objective to make software trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties

Rapid Certifiable Trust

The DoD recognizes the need to field new cyber-physical systems (CPS) capabilities at an increasingly rapid pace, which is why it maintains a number of initiatives on rapid deployment. The demand for more rapid deployment, however, creates a need for verification techniques that can adapt to a faster deployment cadence, especially for CPS that are too big for traditional verification techniques and/or involve unpredictable aspects, such as machine learning.

The goal of Rapid Certifiable Trust is to reduce the deployment time of CPS by reducing the overall development and assurance times. We will do this by enabling the use of unverified commodity software components (e.g., open source drone piloting software) guarded by verified enforcers that guarantee the containment of unsafe component behavior. We are developing compositional verification techniques to allow us to use multiple enforced components minimizing and automatically removing conflicting enforcer assumptions (e.g., reducing a plane's airspeed to avoid crash while increasing airspeed to prevent stalling). These techniques will allow us to assure full-scale systems, even if most of their functionality is implemented by unverified components. Our objective is to develop enforcement verification techniques that scale to at least 10 enforced controllers.

IN CONTEXT: THIS FY2019–21 PROJECT

- builds on line-funded work on Certifiable Distributed Runtime Assurance, the goal of which was to facilitate confident and rapid deployment of autonomous distributed real-time systems (DRTS) operating in uncertain and contested environments
- seeks to verify software-reliant systems that interact with physical processes (e.g., aircrafts) to which existing verification technology does not scale
- will develop enforcing algorithms to identify unsafe control actions and replace them with safe actions
- drones are used to validate our approach in the SEI's drone lab

Principal Investigator

DR. DIONISIO DE NIZ
Principal Researcher and CPS/ULS Initiative Lead
Carnegie Mellon University Software Engineering Institute

Rapid Certifiable Trust

Introduction

Fielding **new technologies** is essential to **preserve superiority**. However, this is only possible if these technologies are **validated for safety**.

Challenges for validation

- Growing system complexity
- Changing behavior at runtime (e.g., machine learning)
- Interactions with physical world (e.g., vehicles)
  - Correct value
  - At right time (before crash)

Methods

Formal automatic verification

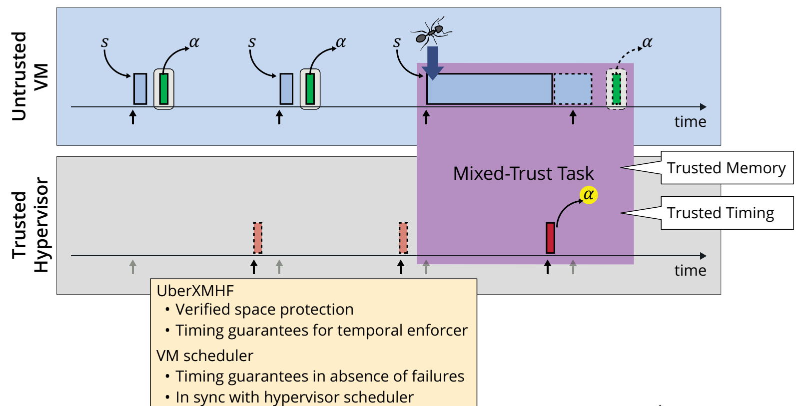
- Scalable
  - Unverified components
  - Monitored and **enforced** by verified components
  - Protected from unverified components
- Verified from
  - Physics: verify reaction of physical model (e.g., physical vehicle)
  - Logic: correct value, with correct protection
  - Timing: At the right time
- Verified protection

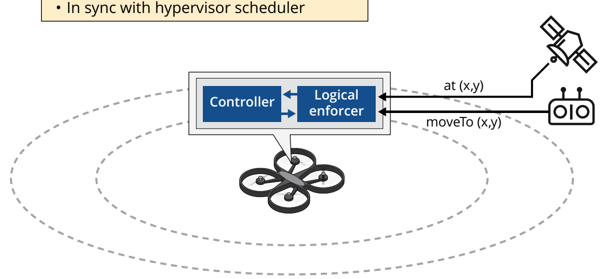
Results

- Real-Time **Mixed-Trust** Computation
- Verified protection mechanism (micro-hypervisor: UberXMHF)
- Timing verification of combined trusted/untrusted (mixed-trust)
- Physics verification of enforcement

Preserve safety by verifying only a small part of the system. **Assure trust** by protecting verified parts.

Trust = Verified + Protected





Verifying Physics

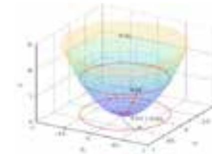
Ensure that an unverified controller cannot violate safety bounds

Controlled System: $\dot{x} = f_\varphi(x) \triangleq f(x, \varphi(x))$
Lyapunov Function $V_\varphi : \mathbb{R}^n \to \mathbb{R}$, $\mathcal{N}_{V_\varphi}(x_{eq}) \subseteq \mathcal{N}_\varphi(x_{eq})$, $V_\varphi(x_{eq}) = 0$ and $\forall x \in \mathcal{N}_{V_\varphi}(x_{eq}) - \{x_{eq}\}$: (i) $V_\varphi(x) > 0$,

$$\dot{V}_\varphi(x) = \frac{\partial V}{\partial x} \cdot f_\varphi(x) < 0$$

Lyapunov level set For $c > 0$,

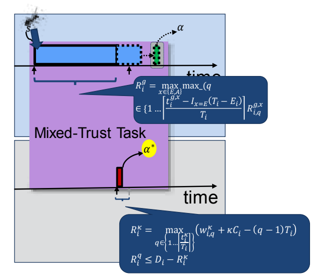
$$\mathcal{E}_\varphi(c) = \{x \in \mathcal{N}_{V_\varphi}(x_{eq}) | V_\varphi(x) \le c\}, \quad c \le 1$$



Mission: sequence of set points

Verifying Timing

Response time ≤ Deadline



Using All Processor Cores While Being Confident about Timing

Today, almost all computers use multicore processors. Unfortunately, satisfying hard real-time requirements of software executing on such computers is challenging because the timing depends on how resources in the memory system are shared, and this information is typically not publicly available. This project addresses this problem.

Multicore processors
Today, almost all computers use multicore processors. These computers have many processor cores such that one program can execute on one processor core and another program can execute on another processor core simultaneously (true parallelism). Typically, processor cores share memory. In today's memory system, a large number of resources are used to make memory accesses faster in general but, unfortunately, also make execution time more unpredictable and dependent on execution of other programs (because these other programs use shared resources in the memory system). A simplified view of a multicore processor with the memory system is shown in Figure 1.

Embedded real-time cyber-physical systems
These systems are pervasive in society in general, as shown by the fact that 99% of all processors produced are used in embedded systems. In many of these systems, computing the correct result is not enough; it is also necessary to compute the correct result at the right time.

These methods assume that **one knows** the resources in the memory system; unfortunately, most chip vendors do not make this information available.

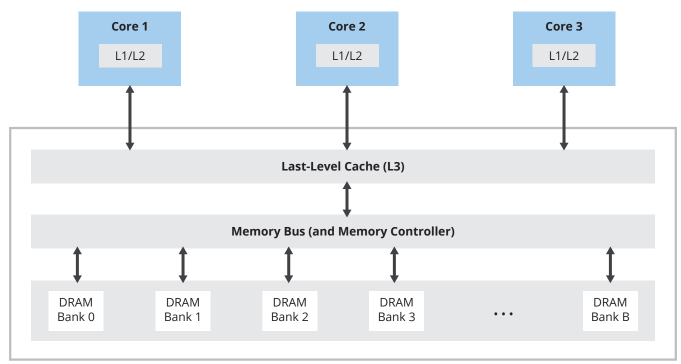


Figure 1: A simplified view of a multicore processor with shared memory

Department of Defense (DoD)
Embedded real-time cyber-physical systems are pervasive in the DoD. Because of the importance of achieving predictable timing, it is common for practitioners to disable all processor cores except one (hence making a multicore processor behave as a single processor system). The importance of timing was recently stressed by AMRDEC's S3I director [1]:

> *"The trick there, when you're processing flight critical information, it has to be a deterministic environment, meaning we know exactly where a piece of data is going to be exactly when we need to—no room for error," [Jeff] Langhout says. "On a multi-core processor there's a lot of sharing going on across the cores, so right now we're not able to do that."*

Current solutions
The current state of the art makes solutions available for managing contention for resources in the memory system and for analyzing the impact of this contention on timing for the case that we know the resources in the memory system.

Problem addressed
In this project, we have addressed the problem of verifying timing of software executing on a multicore processor assuming that we do not know the resources in the memory system.

Results
We have developed a preliminary method—see Andersson, B. et al., "Schedulability Analysis of Tasks with Co-Runner-Dependent Execution Times," *ACM Transactions on Embedded Computing Systems*, 2018.

[1] "Army still working on multi-core processor for UH-60V," May 2017, Available at https://www.flightglobal.com/news/articles/army-still-working-on-multi-core-processor-for-uh-6-436895/

Principal Investigator

DR. BJORN ANDERSSON
Member of the Technical Staff—Principal Researcher
Carnegie Mellon University
Software Engineering Institute

Using All Processor Cores While Being Confident about Timing

Complex, cyber-physical DoD systems, such as aircraft, depend on correct timing to properly and reliably execute crucial sensing, computing, and actuation functions. Any timing failure can have disastrous consequences—an unexpected delay translating sensor data into actuation can cause system instability and loss of control. What's more, the complexity of today's DoD systems has increased the demand for use of multicore processors because unicore chips are either unavailable or not up to the task. However, concerns about timing have led to the practice of disabling all processor cores except one.

In this project, we aim to develop a solution to overcome this obstacle. This is a difficult challenge, because timing is determined by many shared resources in the memory system (including cache, memory banks, memory bus) with complex arbitration mechanisms, some of which are undocumented. The goal of our research is to demonstrate multicore timing confidence by achieving the following sub-objectives:

- *Verification*
  Develop a method for timing verification that does not depend directly on undocumented design qualities and quantities.
- *Parameter extraction*
  Develop a method for obtaining values for parameters in the model of a software system suited for the timing verification procedure mentioned above.
- *Configuration*
  Develop a configuration procedure (such as assigning threads to processor cores or assigning priorities to threads) that takes a model as input and produces a configuration for which the verification will succeed (if such a configuration exists).

IN CONTEXT: THIS FY2019 PROJECT

- builds on prior DoD line-funded research and sponsored work on timing verification of undocumented multicore, verifying distributed adaptive real-time systems, high-confidence cyber-physical systems, and real-time scheduling for multicore architectures
- aligns with the CMU SEI technical objective to bring capabilities through software that make new missions possible or improve the likelihood of success of existing ones

Untangling the Knot: Recommending Component Refactorings

Software-reliant systems need to evolve over time to meet new requirements and take advantage of new technology. However, all too often the structure of legacy software becomes too complicated to allow rapid and cost-effective improvements. This challenge is common in long lived DoD systems, and it makes isolating a collection of functionality for use in a new context, or clean replacement by an improved version, difficult. Software refactoring can facilitate such changes, but can require tens of thousands of staff hours.

This project aims to use AI techniques to create software engineering automation to recommend a set of refactorings that isolates functionality from its tangle of system dependencies. We aim to reduce the time required for this kind of architecture refactoring by two-thirds. In one DoD example, a contractor estimated 14 thousand hours of software development work alone (excluding integration and testing) to isolate a mission capability from the underlying hardware platform. If successful, our work would reduce the development time required to less than 5 thousand hours.

Our solution combines advances in search-based software engineering with static code analysis and refactoring knowledge. It is unique in its focus on mission-relevant goals as opposed to improving general software metrics. This goal is incorporated in genetic algorithms through fitness functions that guide the search to solutions for the project-specific goal. The search algorithm relies on a representation derived from static code analysis and uses formalizations of refactorings as operators to apply during search.

This work has broad implications for moving existing software to modern architectures and infrastructures such as service-based, microservice, and cloud environments. It also addresses a pervasive research challenge in improving automated support for architecture refactoring tasks.

IN CONTEXT: THIS FY2019–21 PROJECT

- builds on prior DoD line-funded research in software architecture analysis, static code analysis, and identifying technical debt
- aligns with the CMU SEI technical objective to make software delivery timely so that the cadence of acquisition, delivery, and fielding is responsive to and anticipatory of the operational tempo of DoD warfighters
- addresses a widespread, recurring need in software organizations. As requirements and technology are never frozen in time, the need to adapt working software to new contexts is likely to remain a common need across many classes of software systems

Principal Investigator

JAMES IVERS
Senior Member of the Technical Staff
Carnegie Mellon University
Software Engineering Institute

On Right

DR. IPEK OZKAYA, Technical Director, Engineering Intelligent Software Systems,
Carnegie Mellon University
Software Engineering Institute

Untangling the Knot: Recommending Component Refactorings

Problem

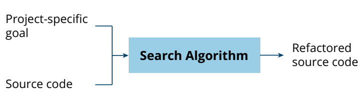
To quickly deliver new capabilities and take advantage of new technologies, DoD needs the ability to efficiently restructure software for common scenarios like

- migrating a capability to the cloud
- harvesting a component for reuse
- replacing a proprietary component

One recent anecdote estimates the effort to isolate a mission capability from the underlying hardware platform at 14,000 staff hours just for development.

Solution

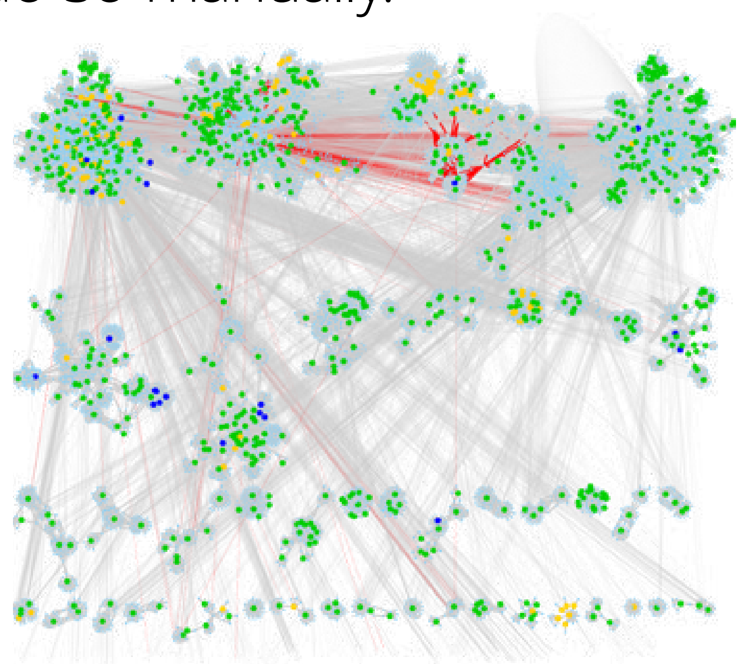
- Create a graph representation of software structure.
- Formalize refactorings as operations over the graph.
- Define fitness functions that evaluate the quality of candidate solutions.
- Apply search-based algorithms to resolve as many problematic couplings as possible while optimizing for multiple objectives.



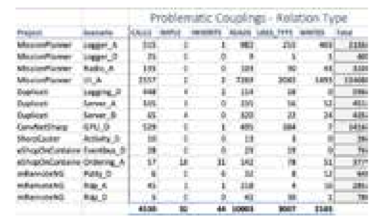
Intended Impact (FY19–21)

- Our refactoring recommendations outperform those based only on quality metrics, reducing problematic couplings by at least 75%.
- Developers accept recommendations.
- Our automation reduces the time to restructure software to 1/3 of the time compared to manual effort.

Automation using **search-based refactoring** can reduce the time required to restructure software to **1/3 of the time** it takes to do so manually.



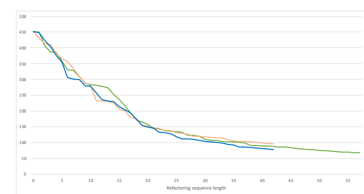
Our prototype automatically identifies problematic couplings that increase complexity and hinder software restructuring.



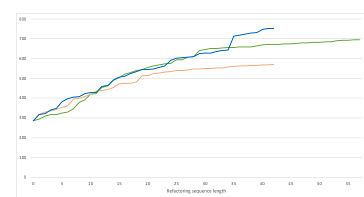
Source: All project data from **github.com/open-source**

Automated search finds sequences of refactorings that collectively solve as much of the project-specific goal as possible.

Number of Problematic Couplings



Amount of Code Included in Harvest



DoD programs with C# implementations can use our interim results now to

- size proposed changes and help prioritize software for migration
- provide input to cost analyses

Automated Code Repair to Ensure Memory Safety

Software vulnerabilities constitute a major threat to DoD. Memory violations are among the most common and most severe types of vulnerabilities.

The main technique that we use (fat pointers) has been previously researched as a compiler pass to repair the intermediate representation (IR) of a program. Our work is novel in applying it as a source-code repair, which poses the difficulty of translating the repairs at the IR level back to source code.

Repair of source code	As a compiler pass
Repairs can be easily audited if desired.	Must trust the tool.
Repairs can be manually tweaked to improve performance.	Difficult to remediate performance issues caused by repair.
Changes to the source code are frequent and easily handled.	Changes to the build process may be difficult and costly.

The C preprocessor can include or exclude pieces of C code depending on the configuration chosen at compile time. We repair configurations separately and merge the results, as illustrated in Figure 3.

We are developing **automated techniques** to repair C source code to **eliminate memory-safety vulnerabilities.**

Figure 1(a): Original Source Code

```c
#define BUF_SIZE 256
char nondet_char();

int main() {
    char* p = malloc(BUF_SIZE);
    char c;
    while ((c = nondet_char()) != 0) {
        *p = c;
        p = p + 1;
    }
    return 0;
}
```

Figure 1(b): Repaired Source Code

```c
#include "fat_header.h"
#include "fat_stdlib.h"
#define BUF_SIZE 256
char nondet_char();

int main() {
    FatPtr_char p = fatmalloc_char(BUF_SIZE);
    char c;
    while ((c = nondet_char()) != 0) {
        *bound_check(p) = c;
        p = fatp_add(p, 1);
    }
    return 0;
}
```

Figure 2. Pipeline for repair of source code

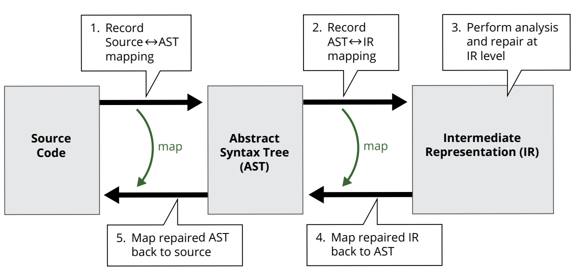
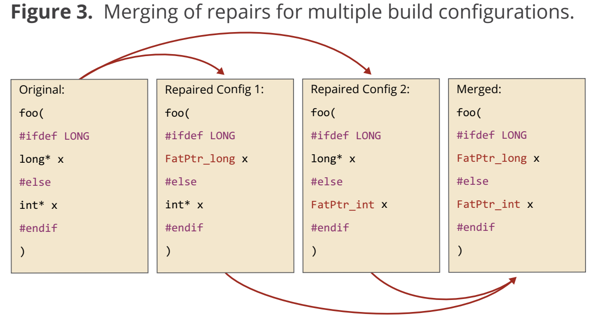


Figure 3. Merging of repairs for multiple build configurations.



We ensure spatial memory safety by replacing raw pointers with **fat pointers**, which include bound information.

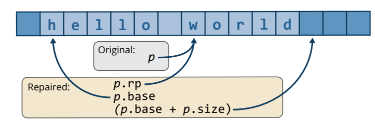
Before dereferencing a fat pointer, a bounds check is performed.

For each pointer type *T**, we introduce a new struct definition:

```c
struct FatPtr_T {
  T*     rp;   /* raw pointer */
  char*  base; /* of mem region */
  size_t size; /* in bytes */
};
```

Pointers stored in heap memory that is reachable by external binary code cannot be fattened. We identify such pointers using a whole-program points-to analysis with an allocation-site abstraction.

Figure 4. Example of fattening a pointer



Limitations: No guarantee of memory safety in the presence of concurrency and things that interact poorly with fat pointers.

Current status: Our tool works on small test cases. We are fixing remaining bugs and adding missing features to handle the SPEC2006 benchmarks.

FY20: Optimize to remove unnecessary fattenings and bound checks.

Future: Extend to other types of repairs and increase level of automation. Work with additional DoD transition partners.

On Left

DR. RUBEN MARTINS,
Systems Scientist,
Carnegie Mellon University
School of Computer Science

Principal Investigator

DR. WILL KLIEBER
Software Security Engineer
Carnegie Mellon University Software
Engineering Institute

Automated Code Repair to Ensure Memory Safety

A serious limitation in assuring the security of DoD software is the inability to take a codebase and either verify that it is memory safe or repair potential bugs to make it memory safe. Existing static analysis tools either report an enormous number of false alarms or fail to report true vulnerabilities.

We propose to design and implement a technique for automatically repairing (in the source code) all potential violations of memory safety so that the program is provably memory safe. For this, we do not need to solve the challenging problem of distinguishing false alarms from true vulnerabilities: we can simply apply a repair to all potential memory-safety vulnerabilities, at a cost of an often small runtime overhead. Usually, only a small percent of a codebase is performance critical; repairs to this part of the codebase might need to be manually tuned, but with an amount of manual effort much less than that of manually repairing all potential memory-safety vulnerabilities in the codebase.

IN CONTEXT: THIS FY2018–20 PROJECT

- extends prior DoD line-funded research in automated repair of code for integer overflow and the inference of memory bounds
- is related to CMU SEI technical work into advancements based on the Pharos static binary analysis framework, vulnerability discovery, and code diversification to avoid detection of vulnerabilities by adversaries
- aligns with the CMU SEI technical objective to make software trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties including known and yet unseen adversary capabilities

Exploring Disruptive Technology Elements for How DoD Will Use Software

SESSION 3

Spiral/AIML: Resource-Constrained Co-Optimization for High-Performance, Data-Intensive Computing

Commanders and warfighters in the field rely on data, and the Department of Defense and U.S. intelligence community have an overwhelming data collection capability. This capability far outpaces the ability of human teams to process, exploit, and disseminate information. Artificial intelligence (AI) and machine learning (ML) techniques show great promise for augmenting human intelligence analysis. However, most AI/ML algorithms are computationally expensive, data intensive, and difficult to implement in increasingly complex computer hardware and architectures. What's more, moving very large amounts of data through tactical and operational military networks requires forward deployment of advanced AI/ML techniques to support commanders and warfighters in theater.

As the military adopts AI/ML to augment human teams, the cost of implementing and re-implementing AI/ML software on new hardware platforms will be prohibitive. To address these challenges, we propose to develop a hardware-software co-optimization system that will

- automatically search and select hardware configurations optimized for a specified computation
- autonomously generate optimized codes for the selected hardware configuration and the irregular, data-intensive computations required for AI/ML algorithms

If successful, our solution will allow platform developers to realize high-performance AI/ML applications on leading-edge hardware architectures faster and cheaper. These advances will allow for rapid development and deployment of capabilities across the spectrum of national and tactical needs.

IN CONTEXT: THIS FY2019–21 PROJECT

- builds on DoD line-funded research and sponsored work on automated code generation for future-compatible high-performance graph libraries, big learning benchmarks, GraphBLAS API specification, and graph algorithms on future architectures
- is related to SEI technical work on building a COTS benchmark baseline for graph analytics
- aligns with the CMU SEI technical objective to bring capabilities through software that make new missions possible or improve the likelihood of success of existing ones

Principal Investigator

DR. SCOTT MCMILLAN
Principal Engineer

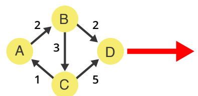
Carnegie Mellon University
Software Engineering Institute

CMU Collaborators

Elliott Binder, Mark Blanco, Paul Brower, Franz Franchetti, James C. Hoe, Tze Meng Low, Peter Oostema, Fazle Sadi, Daniele Spampinato, Upasana Sridhar, and Jiyuan Zhang

Spiral/AIML: Frontiers of Graph Processing in Linear Algebra

Introduction
Graph algorithms can be expressed as sequences of linear, algebra-like operations through the use of the adjacency matrix. Adjacency matrices are used to represent graphs instead of vertices and edges.



$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & . & 2 & . & . \\ B & . & - & 3 & 1 \\ C & 1 & . & . & 5 \\ D & . & . & . & . \end{array}$$

Our work extends the use of linear algebra beyond simple graph traversal.

Writing Graph Algorithms in LA
Many graph primitives can be cast in terms of LA operations.

- Neighbors of a vertex v—vector matrix products.
- Filtering—Hadamard products.
- Semi-rings—to represent Matrix Multiplication like operation (Single-Source Shortest Path uses the min-plus semiring).

This mapping gets rid of the need for “experts” to formulate graph algorithms in LA.

- Formally derived algorithms

Families of Graph Algorithms
Multiple graph algorithms can be enumerated for the same specification.

Our LA approach uses triangle counting:

$\Delta = 1/6\ \Gamma(A\text{^}3)$

This approach allows us to analyze graph algorithms for individual performance characteristics. The algorithm that best-fit the situation is chosen.

Our **linear algebraic approach** to graph algorithms provides a **flexible framework** that enables high performance code generation for **faster network analysis.**

Writing Graph Algorithms in LA

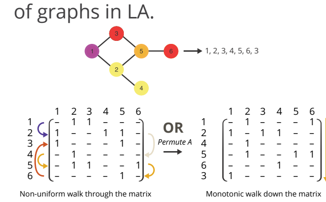
Operation	**Linear Algebra**
Edge Filtering	$A_b = A > \Delta$ $A_H = A_b \circ A$
Vertex bucketing	$v_{Bi} = (\Delta i \leq tent_{dists} < \Delta(i+1))$ $v'_{Bi} = ((\Delta i \leq new_{dist} < \Delta(i+1)) \circ (new_{dist} < old_{dist}))$
Neighborhood Traversal with Relaxation	$t_{Bi} = (tent_{dists} \circ v_{Bi})$ $y^T_{out} = t_{Bi}(min.+)\begin{bmatrix} . & 2 & . & . \\ . & . & 3 & 2 \\ 1 & . & . & 5 \\ . & . & . & . \end{bmatrix}$

Busting Myths about Linear Algebraic Approach
Unifying Edge and Vertex Centric Algorithms

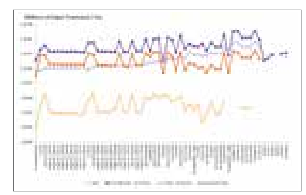
$$\begin{pmatrix} A_{00} & a_{01} & A_{02} \\ * & 0 & a_{T12} \\ * & * & A_{22} \end{pmatrix} \quad \begin{pmatrix} A_{00} & a_{01} & A_{02} \\ * & 0 & a_{T12} \\ * & * & A_{22} \end{pmatrix}$$

Both classes of algorithms can be expressed using a single linear algebraic framework.

Depth-First Search in Linear Algebra
First look at expressing depth-first traversal of graphs in LA.



Linear Algebraic Approach can Yield Good Performance



2019 Graph Challenge Champion
2018 Graph Challenge Finalist
2017 Graph Challenge Honorable Mention

References

Low, Tze Meng, et al. “First look: Linear algebra-based triangle counting without matrix multiplication.” 2017 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, 2017.

Lee, Matthew, and Tze Meng Low. “A family of provably correct algorithms for exact triangle counting.” Proceedings of the First International Workshop on Software Correctness for HPC Applications. ACM, 2017.

Low, Tze Meng, et al. “Linear Algebraic Formulation of Edge-centric K-truss Algorithms with Adjacency Matrices.” 2018 IEEE High Performance extreme Computing Conference (HPEC). IEEE, 2018.

Sridhar, Upasana, et al. “Delta-stepping SSSP: from vertices and edges to GraphBLAS implementations.” 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2019.

Spampinato, Daniele G., Upasana Sridhar, and Tze Meng Low. 2019. “Linear algebraic depth-first search.” In Proceedings of the 6th ACM SIGPLAN International Workshop on Libraries, Languages and Compilers for Array Programming (ARRAY 2019). ACM, New York, NY, USA, 93-104. DOI: https://doi.org/10.1145/3315454.3329962

A Series of Unlikely Events: Learning Patterns by Observing Sequential Behavior

Introduction

Modeling patterns of behavior is a task that underlies numerous difficult artificial intelligence tasks:

How do I detect when adversaries are deviating from normal routines?

How can I automate the teaching of novice analysts to perform complex tasks as if they were expert analysts?

In this work, we use a class of techniques called **Inverse Reinforcement Learning (IRL)** to model sequential behavior to answer questions like these and others.

Methodology

Given observations of behavior:

$$\mathcal{B}=\{((s_1,a_1),(s_2,a_2),\dots)_1,\dots,((s_1,a_1)\dots)_n\}$$

Learn a reward $R:\mathcal{S}\times\mathcal{A}\mapsto\mathbb{R}$ that best explains the observed behaviors.

How IRL algorithms work:

(Two steps)

1. Learn policy $\pi:\mathcal{S}\mapsto\mathcal{A}$ from reward R (policy trained to maximize expected reward as in reinforcement learning).
2. Compute expected behaviors computed from policy π, compared to observed behaviors $\mathcal{B}$, and use to update reward R.

How to use IRL for...

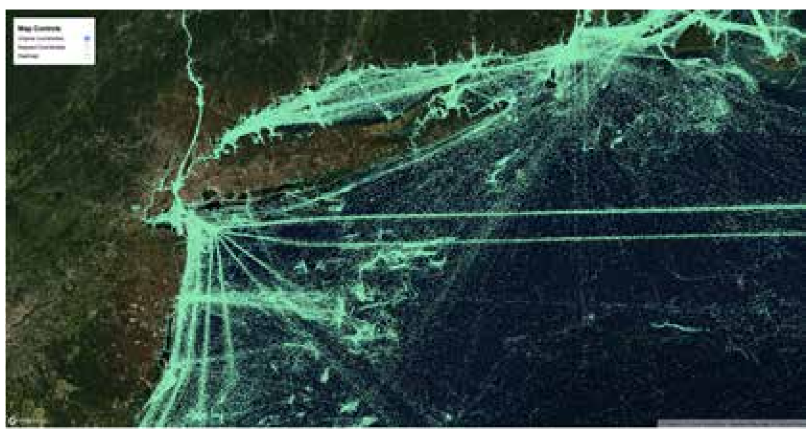
Activity-based intelligence:

1. Learn R from observed behaviors.
2. If new behavior exhibits low-reward actions in states, flag as abnormal.

Teaching expert behavior:

1. Learn R from expert behavior.
2. When a novice is in a state where she doesn't know the proper action, suggest the one with highest reward.

Inverse Reinforcement Learning techniques are an efficient and effective means to perform **activity-based intelligence** or to teach novices how to **perform tasks like experts.**



©Mapbox, ©OpenStreetMap, and ©DigitalGlobe

Goals of this work

1. Apply IRL Techniques to DoD/IC relevant problems:
   - Employ efficient implementations that scale to a large number of observations.
   - Build demonstration data ingestion to visualization tools.
2. Develop Novel IRL Techniques that are robust to rare events.
3. Develop techniques that are able to explain, simulate, and demonstrate expert behavior.

Progress

1. Implementation of Maximum Causal Entropy IRL that is up to **1000x faster** than academic implementation.

   Source: Ziebart, Brian D., J. Andrew Bagnell, and Anind K. Dey. "Modeling interaction via the principle of maximum causal entropy." Proceedings of the 27th International Conference on Machine Learning. Omnipress, 2010.

2. Data ingestion pipeline and visualization of IRL being used to model ship behavior on U.S. Coast Guard Automatic Ship Identification (AIS) data.
3. Investigation into current IRL techniques and how robust they are to rare events, leading to initial formulation of robust IRL technique.
4. Data scientist study to capture expert data scientist behavior.

Additional Figures

Principal Investigator

DR. ERIC HEIM
Senior Research Scientist, Machine Learning
Carnegie Mellon University
Software Engineering Institute

A Series of Unlikely Events: Learning Patterns by Observing Sequential Behavior

The Department of Defense (DoD) and the intelligence community (IC) frequently analyze activity based intelligence (ABI) to inform missions about routine patterns of life (POL) and unlikely events that signal important changes. For example, monitoring parking lots of military bases may indicate changing threat levels or upcoming military action. Despite growing research on general solutions for routine detection technologies, current algorithms are typically hand-crafted for particular applications, require labeled anomalous data, and have high false-positive rates that require verification by human analysts.

We propose an alternative approach, inverse reinforcement learning (IRL), that observes all states and actions in data and computes a statistical model of the world that includes whether each behavior is part of a routine. Deviations from routines have a low likelihood of occurrence within the model. The statistical model can also explain why an action is labeled as routine or anomalous and could be used by analysts to prioritize the anomalies and to retrain models to reduce false positives.

Though powerful, IRL techniques pose a number of both practical and fundamental challenges when applying them to dynamic, large-scale DoD and IC missions. In this project we focus on three of these challenges:

1. scaling IRL methods to DoD/IC-scale problem domains using efficient implementations of state-of-the-art techniques and high-performance computing
2. making IRL techniques robust to novelty, thus allowing them to reason about never-seen-before behaviors
3. developing IRL techniques that expose key characteristics in data that could explain observed behaviors

IN CONTEXT: THIS FY2018–20 PROJECT

- builds on DoD line-funded research, including graph algorithms and future architectures, big learning benchmarks, automated code generation for future-compatible high-performance graph libraries, data validation for large-scale analytics, and events, relationships, and script learning for situational awareness
- aligns with the CMU SEI technical objective to bring capabilities through software that make new missions possible or improve the likelihood of success for existing missions

Emotion Recognition from Voice in the Wild

The human voice contains traces of many bio-parameters, including emotional state [Singh 2016]. Accurately recognizing emotion from voice is important for a host of defense applications, including speaker profiling for intelligence and human-machine teaming.

For this project, we proposed a novel approach that capitalizes on recent advances in micro-articulometry, the measurement of speech properties at the phoneme level—the smallest unit of speech (for instance, /k/ in "cat"). A micro-articulometry approach is robust against noisy and short-duration signals and captures finer nuances than current approaches.

The desired outcome of this project was twofold: first, the creation of a novel, open-source, statistically labeled emotional speech database that maps to a continuum of emotions rather than discrete labels, and second, a working end-to-end emotion recognition prototype.

IN CONTEXT: THIS FY2018–20 PROJECT

- pursued DoD priorities for machine perception, reasoning and intelligence, and human/ autonomous system and collaboration
- built on CMU SEI expertise that has led to unique contributions in the field of machine emotional intelligence (e.g., heart rate extraction from video, facial micro-expression analysis)
- benefited from collaboration with CMU world leaders in micro-articulometry techniques
- aligned with the CMU SEI technical objective to bring capabilities through software that make new missions possible or improve the likelihood of success of existing ones

Principal Investigator

OREN WRIGHT
Research Scientist
Carnegie Mellon University
Software Engineering Institute

Emotion Recognition from Voice in the Wild

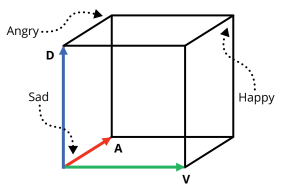
Introduction

Accurately recognizing emotion from voice is important in defense applications such as speaker profiling and human-machine teaming, but is currently infeasible. We introduce a new, continuous speech emotion recognition database, CMU-SER, and a set of micro-articulometry techniques that can capture finer nuances than the current state of the art.

The CMU-SER Database

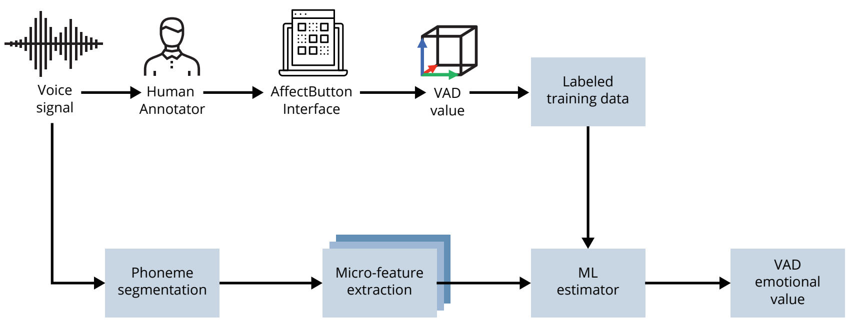
The CMU-SER database, to be released at the end of 2019, is the largest ever speech emotion recognition database. In-the-wild audio clips come from podcasts, radio, and television, and are annotated via crowdsourcing. Annotators used an interface based on the VAD emotional state model that allows users to pick from a continuum of emotions instead of discrete labels. Several annotators label each audio clip. CMU-SER features:

- Over **29,000** annotated audio clips, totaling over **54** hours of voice recordings
- Over **324,000** unique annotations

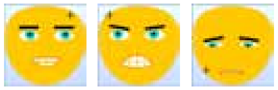


The VAD model: Valence, arousal, and dominance characterize affect in three dimensions.

We introduce a new in-the-wild **speech emotion recognition database,** and **novel extraction techniques** built with machine learning and micro-articulometry.



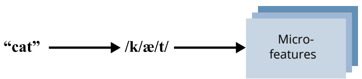
System diagram. Voice data is labeled by crowdsource participants using the AffectButton interface. This labeled data is then used to train classifiers and predict emotion.



The AffectButton: An interface first introduced for self-reporting human affect. The facial icon changes based on mouse movement, and each expression corresponds to a unique VAD score.

Micro-Articulometry and Machine Learning

Prior work in speech emotion recognition typically operates at the utterance level—the level of the spoken word or statement. Instead, our approach operates at the phoneme level—the level of the constituent units of speech—using micro-articulometry techniques. We use fine-grain voice features, such as formant position, in conjunction with deep learning to predict emotional state. Results will be published at the end of 2019.



Micro-articulometry: The measurement and modeling of articulatory properties at the phoneme level.

Example Voice Features:

- Diplophonicity
- Flutter
- Formant bandwidth
- Formant position
- Vocal fry
- Glottalization
- Nasality
- Raspiness
- Shimmer
- Tremor
- Voicing onset time
- Wobble

Micro-articulatory voice features are used to train classifiers and predict emotion. Each voice feature requires its own set of signal processing algorithms to extract and measure.

Field Stripping a Weapons System: Building a Trustworthy Computer

Introduction

The DoD finds itself increasingly reliant on COTS systems for mission critical applications. In particular, the length and opacity of the hardware supply chain presents a large attack surface for the insertion of hardware-based privilege escalation vulnerabilities that can lead to a total system compromise even in the theoretical presence of completely bug-free software.

This project demonstrates the feasibility of *field stripping* a computer by rebuilding it from complete hardware, software, and compiler tool chain source code.

Method

Our proposed solution starts with the observation that hardware is *compiled* from source code in a manner *very similar* to software.

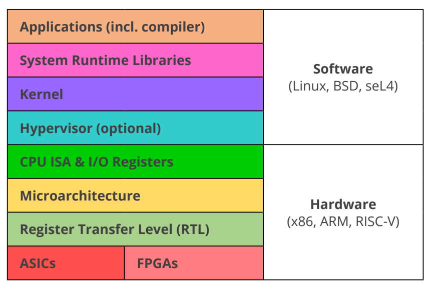
Next, we examine the hardware attack surface:

- design (source) defects
- malicious hardware compiler
- microchip (ASIC) fabrication attack

and observe that using FPGAs instead of ASICs helps mitigate against fabrication attacks.

The remaining problems can be addressed by requiring that the entire system be *fully rebuildable from source code:* to the hardware, the software it is intended to support, and to all hardware and software compilation tool chains required to build the fielded system.

We built a **fully functional** computer, as **trustworthy** as the **bounded, self-hosting** set of **sources** to all its parts: hardware **and** software, including compiler toolchains.

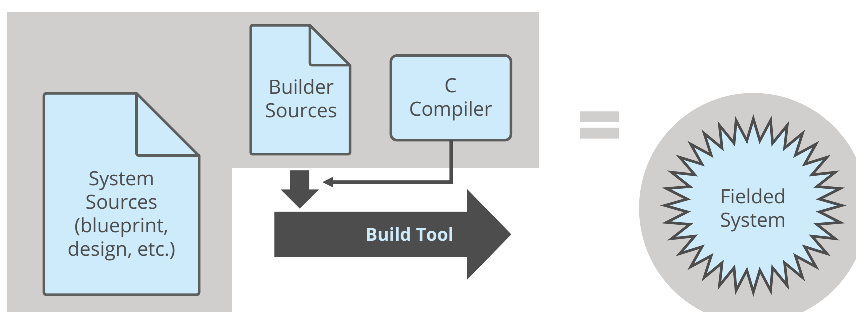


Field Stripping a Computer: Abstraction Layers

- "Trusting Trust" self-propagating compiler hack (Ken Thompson)
  - clean source → malicious binary (incl. self-build)
  - sources to hack not needed beyond 1st iteration!
- Solution: Diverse Double Compilation (DDC) (David A. Wheeler)
  - suspect compiler A: sources S_A, binary B_A
  - trusted compiler T: binary B_T

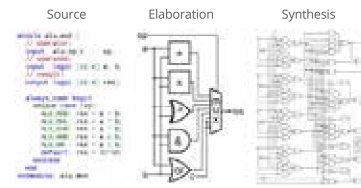
    $S_A \rightarrow B_A \rightarrow X$ $\quad$ $S_A \rightarrow B_T \rightarrow Y$
    - X, Y *functionally* identical, different binaries

      $S_A \rightarrow X \rightarrow X_1$ $\quad$ $S_A \rightarrow Y \rightarrow Y_1$
    - **X_1, Y_1 must be identical binaries!**



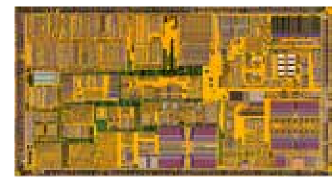
Anchoring Trust for Fielded Systems

Hardware Development



- Followed by Technology Mapping, Placement and Routing for either ASICs or FPGAs

ASICs (Application Specific Integrated Circuits)



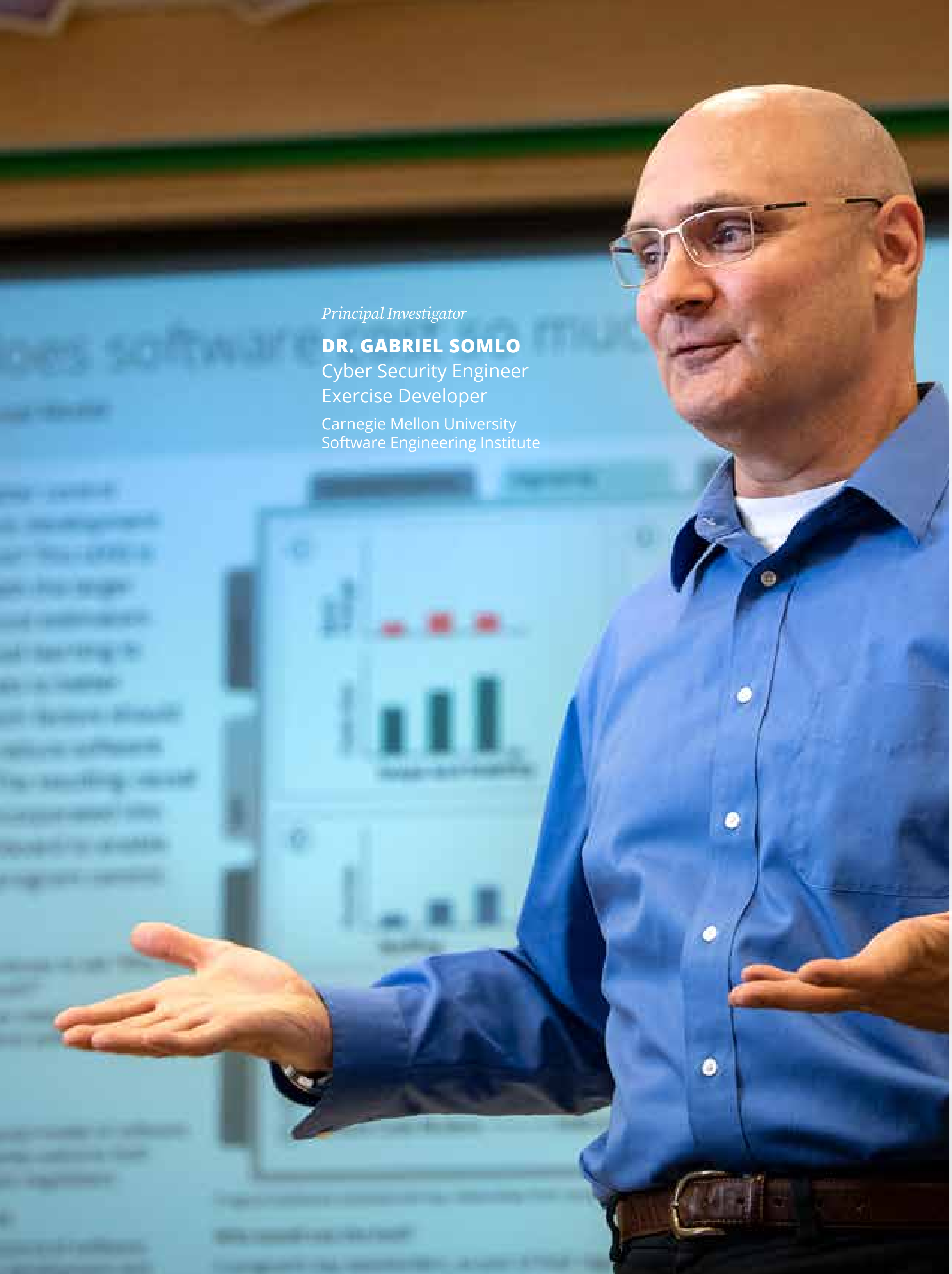
- dedicated, optimized etched silicon
  - photolithographic masks
- "hard" IP cores

FPGAs (Field Programmable Gate Arrays)

- grid of programmable blocks and interconnect
  - bitstream
- "soft" IP cores

Bootstrapping a Clean Room System

- Use DCC to verify clean C compiler.
- Compile clean HDL toolchain.
- Cross-compile clean target OS.
- Compile clean target FPGA bitstream.
- Boot self-hosting target system.



Principal Investigator

DR. GABRIEL SOMLO
Cyber Security Engineer
Exercise Developer
Carnegie Mellon University
Software Engineering Institute

Field Stripping a Weapons System: Building a Trustworthy Computer

The DoD needs to employ commercial off-the-shelf (COTS) hardware, but this hardware is designed, developed, and manufactured using proprietary means, in secret, by private organizations. In such an environment, the opportunity exists for accidental or malicious insertion of defects, backdoors, and Trojans. Consequently, there is no way to know whether COTS hardware is secure, a fact highlighted in 2018 by the Spectre and Meltdown cybersecurity vulnerabilities.

While completely trustworthy (self-hosting) software ecosystems are available today, the current state of the practice does not similarly allow us to prove the trustworthiness of the underlying hardware. The hardware description language (HDL) for chips, and the HDL compiler suites, are not available for anyone outside the vendor for use in auditing or rebuilding. This is a major concern for the DoD.

The goal of this project is to demonstrate the practicality of a comprehensive approach to guaranteeing the trustworthiness of computer systems. It will focus on building a fully Linux-capable computer on top of a field-programmable gate array (FPGA), using open source CPU and system-on-chip (SoC) designs compiled with an open source HDL toolchain. This would be the first known prototype of an open source, self-hosting hardware and software platform capable of empirically proving that a fielded system's trust is equivalent to that of its collected sets of sources (hardware, software, and build tool chains).

IN CONTEXT: THIS FY2019–21 PROJECT

- relates to the DoD's "Owning the Technical Baseline" concept, whereby it would possess the technical resources and competencies (skills) to manage lifecycle design and sustainment, engage effectively with vendors, and independently analyze and verify vendor product claims
- aligns with the CMU SEI technical objective to make software trustworthy in construction, correct in implementation, and resilient in the face of operational uncertainties

Summarizing and Searching Video

The U.S. relies on surveillance video to determine when activities of interest occur in a surveilled location. Yet, there is a lack of automated tools available to assist analysts in monitoring real-time video or analyzing archived video [Seligman 2016]. Consequently, analysts now need to dedicate full attention to video streams to avoid missing important information about ongoing activities and patterns of life; and, in tactical settings, warfighters miss critical information for improved situational awareness because they cannot stare at a tablet strapped to their chest.

In this work, we are developing algorithms to improve training of machine learning algorithms necessary for detecting objects, better track those objects, and recognize patterns of objects and object interactions.

IN CONTEXT: THIS FY2018–20 PROJECT

- builds on prior DoD line-funded research into the foundations for summarizing and learning latent structure in video, structural multitask transfer learning for improved situational awareness, and generalizing supervised latent Dirichlet allocation (a strategy to provide supervision hints to an unsupervised algorithm)
- draws from sponsored engagements for DoD programs and agencies
- aligns with the CMU SEI technical objective to bring capabilities through software that make new missions possible or improve the likelihood of success of existing ones

> ***Note***
> The illustrations on the following two pages describe additional threads related to this research.

Principal Investigator

EDWIN J MORRIS
Senior Member of the Technical Staff
Carnegie Mellon University Software Engineering Institute

Summarizing and Searching Video: Geometry-Aware Visual Surveillance

Introduction

Tracking moving objects in a video is a fundamental problem in surveillance. This problem is made even more challenging if the camera is constantly moving, as it is in drone surveillance. In this work, we develop a pipeline that estimates camera motion on-the-fly while tracking, allowing us to “cancel” the camera motion before deploying our tracking algorithm. The tracker works by matching detected objects against future stabilized frames, with the help of a trajectory forecaster.

Method

We operate on images from pairs of frames:

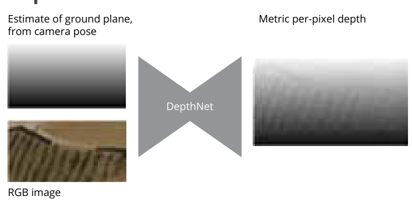
1. Using the frames and the drone’s approximate pose (height and angle from ground) estimate the depth of each pixel.
2. Stabilize for egomotion by re-projecting the 3D scene from a virtual camera placed in a stationary position.
3. Featurize the stabilized images with a learned Encoder-Decoder.
4. Crop the object to track from the first feature map; call this the template. Crop a relatively larger region from the second feature map; call this the search region.
5. Cross-correlate the template with the search region, obtaining a map of possible matches.
6. Refine the probability map using a velocity or trajectory estimate.
7. Take the best unique match.

Results

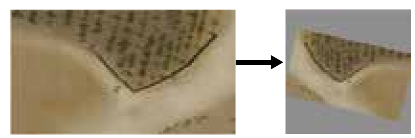
Our tracker achieves an average Intersection Over Union (IOU) score of 0.7958 on our synthetic dataset, whereas a similar un-stabilized tracker achieves 0.5185.

Our geometry-aware object tracker **stabilizes the 3D scene** during tracking, improving its ability to **match** and **forecast**.

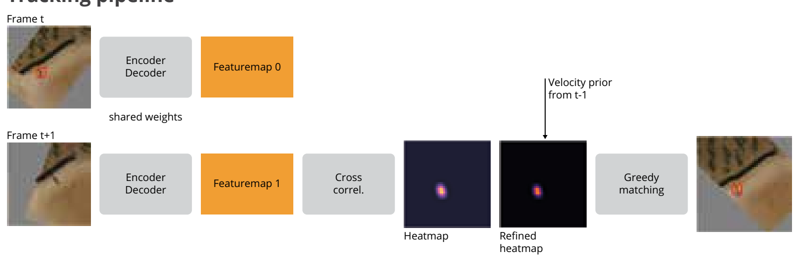
Depth estimation



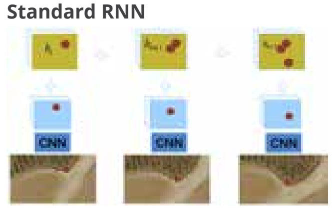
Egomotion Stabilization



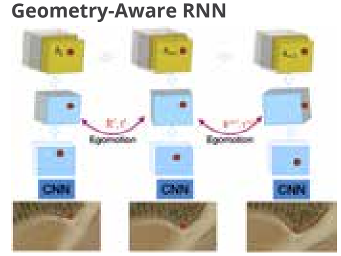
Tracking pipeline



Standard RNN



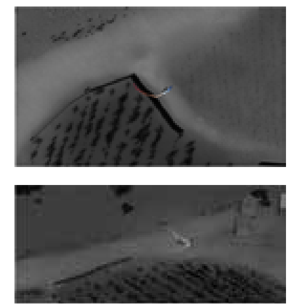
Geometry-Aware RNN



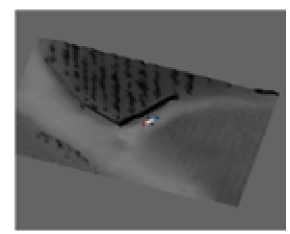
Stabilization boosts IOU by over 10%

Method	Mean IOU
Unstabilized	0.5185
Unstabilized+DataAug	0.5567
Unstabilized+DataAug+Hungarian	0.6240
Unstabilized+DataAug+Hungarian+VelocityPrior	0.6428
Stabilized+DataAug+Hungarian+VelocityPrior	**0.7958**

Trajectory forecasting is possible only in stabilized space



Raw Images and Trajectories



Stabilized Images and Trajectories

Summarizing and Searching Video: Domain Adaptation

Problem:
Despite impressive improvements in machine learning systems in recent years, **classifiers still struggle to perform when there is little or no training data in the target environment.** Semantic differences, such as perspective and object density, between source and target environments can significantly degrade classifier accuracy. Non-semantic differences, such as differences in object environment, can significantly degrade classifier accuracy. Differences between the trained and real world data sets also hamper classifier performance.

This is particularly problematic in the tactical setting, where there is limited image data. Data from target environments may be scarce, or have few examples of object classes. Data may be heterogeneous with regard to perspective, scale and quality. It may also have limited to no metadata.

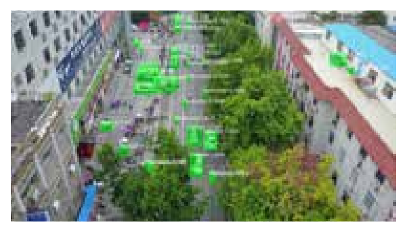
Solution:
- Stratification of existing data to match semantics of the target environment
- CycleGAN for unsupervised style transfer between images
- Object detection & classification with mask-RCNN trained on existing, labeled data which has had style transfer applied to it

Enabling use of existing labeled data sets to **train detectors in low data environments**

Object Identification

Poor Object Identification

Improved Object Identification



@article{zhuvisdrone2018,
title={Vision Meets Drones: A Challenge},
author={Zhu, Pengfei and Wen, Longyin and Bian, Xiao and Haibin, Ling and Hu, Qinghua},
journal={arXiv preprint arXiv:1804.07437},
year={2018}
}

Results:
Our work suggests that with significantly different datasets, style transfer is insufficient to create a substitute for training data within the target domain. Furthermore, supplemental data with or without style transfer to target environment has shown minimal benefit as a supplement to target domain data.

Future Directions: Improved semantic matching of target and source images
- Currently, the only semantics we match on are object density. This needs to be expanded to include other image features such as perspective and scale.
- After improved semantic matching, we can again ask if this observation holds to where the target and source environments are semantically similar.

Automatic determination of object identification quality
- Create a flag for when automatic object identification quality is suspect.

Summarizing and Searching Video: Patterns-of-Life Analysis

Introduction

Over the past five years, image recognition has improved dramatically thanks to new deep learning architectures. Coupled with systems such as the Navy's Minotaur System, it is now possible to start with a raw video feed and generate geo-located tracks of multiple human and non-human actors in real time. These tracks are themselves a new type of data which can be subjected to further downstream processing called Pattern-of-Life Analysis.

Pattern-of-Life (PoL) Analysis

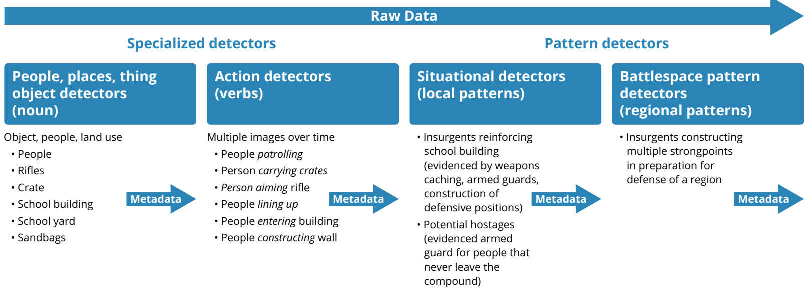
In PoL Analysis, we move from the "what is it?" question to the "what is it doing?" question. At the lowest levels, a PoL analyzer might take track data for a set of objects and either identify targeted types of behaviors (e.g., roadblock, insurgents planting an IED, patrol of a compound, etc), or report unusual or suspicious types of activity (i.e., anomaly detection). Higher-level PoL analyzers might further refine these analyses to identify situations (e.g., compound with high value target).

Methods

While much progress has been made, real-time tracking is still an emerging technology. Tracks can be lost, or registration errors can occur. In addition, it is very difficult to obtain ground truth for this type of data. For these reasons, we have chosen to use simulation to generate synthetic patterns to explore the possibilities of PoL analysis with perfect data. We use the SUMO traffic simulator and have explored both classification and anomaly detection tasks using this data.

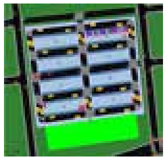
As **image recognition and tracking algorithms** improve, new types of data will emerge, enabling the design of **higher order detectors.**

Object Identification

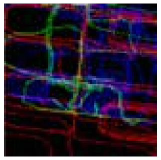


Metadata: Minimally who-what-when-where, but augmented by post-detector analysis producing additional metadata

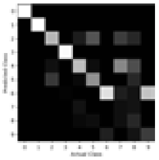
We demonstrate our PoL Analysis techniques using a "Shopping Plaza Parking Lot" example in which we analyze simulated spaciotemporal tracks to identify specific types of activity.



SUMO traffic simulator used to simulate vehicles and pedestrians including customers, employees, carpoolers and "drug dealers."



For each track, an ego-centric patch is generated characterizing the movement of that vehicle or pedestrian and its relationship to other vehicles, pedestrians and fixed reference points.

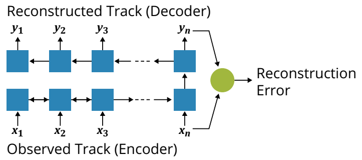


Using a CNN classifier, we correctly identified the track type with 86% accuracy out of 10 possible classes.

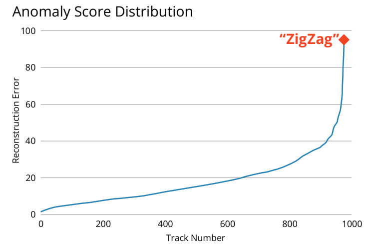
Anomaly Detection

In another experiment, we simulated 950 "best-path" vehicle trips into which we injected one with an unusual "ZigZag" path.

We used an LSTM autoencoder to learn typical track behavior and compute a "reconstruction error" representing the degree to which a track is consider "unusual."



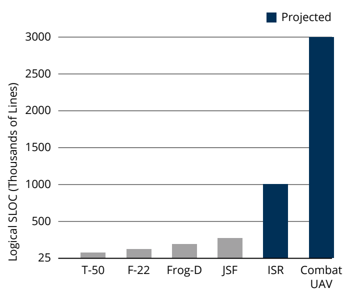
Of the 950 tracks, our injected "ZigZag" track consistently scored in the top 5 in terms of reconstruction error demonstrating the ability to identify anomalous behaviors.



Projecting Quantum Computational Advantage Versus Classical State of the Art

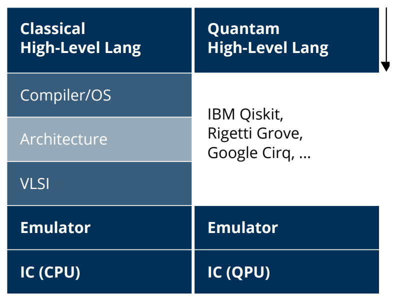
Introduction

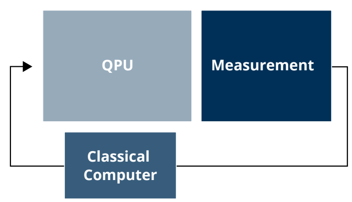
A major milestone in quantum computing research is to demonstrate quantum supremacy, which refers to a quantum computer performing a calculation that is unfeasible for a classical computer [1]. While quantum supremacy may be demonstrable in the near-term noisy intermediate scale quantum computing (NISQ) era [2]–[6], such a demonstration of supremacy does not afford an advantage for practical applications. A common practical problem used in benchmarking high performance classical and quantum computing is Maxcut, with applications in domains such as machine scheduling [7], image recognition [8], electronic circuit layout [9], and software verification and validation [10], [11].



T. Belote, C. Elliott, Safe & Secure Systems & Software Symposium 2014.

Achieving **quantum advantage** requires classical state of the art.





Infrastructure

For practical applications, quantum advantage has to be measured against the best performance that classical computing offers. This work uses the hybrid quantum-classical Quantum Approximate Optimization Algorithm (QAOA) [12].

QAOA
- GW (SDP) [13]
- AKAXSAT (BnB) [14]
- MBO (Laplacian) [15]
- UFO [16]

QAOA

AWS: C5 Intel Xeon
PSC: Bridges (12 TB)
IBM Melbourne 16

[1] John Preskill. “Quantum Computing in the NISQ Era and Beyond.”

[2] Sergio Boixo, et al. “Characterizing Quantum Supremacy in Near-Term Devices.” 2018.

[3] C. Neill. “A Blueprint for Demonstrating Quantum Supremacy with Superconducting Qubits.” 2018.

[4] H. Wang. “Toward Scalable Boson Sampling with Photon Loss.” 2018.

[5] Edward Farhi, et al. “Quantum Supremacy through the Quantum Approximate Optimization Algorithm.” 2016.

[6] Michael J. Bremner. “Achieving Quantum Supremacy with Sparse and Noisy Commuting Quantum Computations.” 2017.

[7] Bahram Alidaee, et al. “0-1 Quadratic Programming Approach for Optimum Solutions of Two Scheduling Problems.” 1994.

[8] Hartmut Neven, et al. “Image recognition with an Adiabatic.” 2007.

[9] Michel Deza and Monique Laurent. “Applications of Cut Polyhedra.” 1994.

[10] Manu Jose, et al. “Cause Clue Clauses: Error Localization Using Maximum Satisfiability.” 2011.

[11] L. Guo, et al. “A Complexity Metric for Concurrent Finite State Machine Based Embedded Software.” 2013.

[12] Edward Farhi, et al. “A Quantum Approximate Optimization Algorithm.” 2014

[13] Michel X. Goemans, et al. “Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming.” 1995.

[14] Adrian Kügel. “Improved Exact Solver for the Weighted Max-SAT Problem.” 2011

[15] Blaine Keetch, et al. “A Max-Cut Approximation Using a Graph Based MBO Scheme.” 2017.

[16] V. Ashcay, et al. “Reachability Deficits in Quantum Approximate Optimization.” 2019.

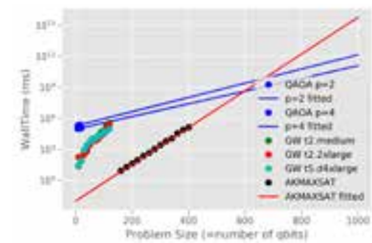
[17] M.B. Hastings. “Classical and Quantum Bounded Depth Approximation Algorithms.” 2019.

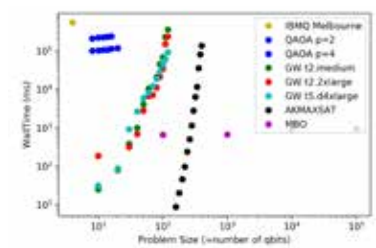
[18] S.S. Tannu, et al. “Mitigating Measurement Errors in Quantum Computers by Exploiting State-Dependent Bias.” 2019

[19] S.S. Tannu, et al. “Ensemble of Diverse Mappings: Improving Reliabitliy of Quantum Computers by Orchestrating Dissimilar Mistakes.” 2019.

Performance Results

The quality of solutions (cut value, bounds) are all comparable among algorithms (classical and QAOA).





Future Work

- Improve QAOA parameterization.
- Increase number of hardware configs. What is needed from hardware to achieve advantage?
- Understand MBO and UFO performance. Is there a quantum version?

Principal Investigator

DR. JASON LARKIN
Research Scientist
Carnegie Mellon University
Software Engineering Institute

Projecting Quantum Computational Advantage Versus Classical State of the Art

Software verification and validation (V&V) is crucial for DoD systems, but it is both computationally difficult and expensive. Some estimates put V&V at 50 percent of total cost for most systems. What's more, as DoD code bases become larger, the computational costs to perform state-of-the-art V&V becomes intractable and exceeds the limitations of current hardware. New computer paradigms are needed to tackle this problem, and one that holds great potential is quantum computing. The DoD recognizes this potential and is looking for ways to harness it for mission capability.

The goal of this project is to predict when (or whether) quantum computing will be mission-capable for hard combinatorial optimization problems in software V&V. It's inspired by challenges related to Lockheed's flight control systems. This work also seeks to determine whether currently available noisy intermediate-scale quantum (NISQ) computers, which are prone to high error rates, can be leveraged to perform useful computation.

Quantum processing units (QPU) with quantum algorithms offer a range of computational speedups over classical algorithms. These superior algorithms will require 1-10 million qubits (the quantum computing analog to a standard bit) with quantum error correction (QEC). Estimates in the field place this achievement about 10-20 years in the future. Our work addresses the challenges of quantum computing with a small number of qubits and large error rates in these qubits. Current research seeks to mitigate these effects through algorithms (Quantum Approximation Optimization Algorithm [QAOA]) and Margaret Martonosi's software engineering work on bridging the gap between hardware and the kind of software quantum computing hopes to enable.

IN CONTEXT: THIS FY2019–21 PROJECT

- relates to DoD interest in applying quantum computing to mission capability
- aligns with the CMU SEI technical objective to make software trustworthy in construction, correct in implementation, and resilient in the face of uncertainties, including known and yet unseen adversary capabilities
- aligns with the CMU SEI technical objective to bring capabilities through software that make new missions possible or improve the likelihood of success for existing missions
- provides a gateway into futuristic computing architectures and increased computational power for artificial intelligence and machine learning

Graph Convolutional Neural Networks

A growing number of Department of Defense (DoD) data problems are graph problems: the data from sources such as sensors, web traffic, and supply chains are full of irregular relationships that require graphs to elucidate. For example, testing and evaluation produces massive, heterogeneous datasets, and analysts can use graphs to reveal otherwise hidden patterns in these data, affording the DoD a more complete understanding of a system's effectiveness, survivability, and safety.

Large and complex datasets demand new approaches to graph problems. Deep learning, which uses brain-inspired algorithms to "learn" from massive amounts of data, shows promise but so far has been limited to Euclidean data. Convolutional neural networks (CNNs), which are used for analyzing visual imagery, have revolutionized image processing and computer vision, but haven't been successfully extended to graph problems.

This project used graph signal processing formalisms to create new deep learning tools for graph convolutional neural networks (GCNNs). Our approach employed topology-adaptive graph convolutional networks, introduced in 2017 by researchers at Carnegie Mellon University. This new class of GCNNs is grounded in graph signal processing theory and can be applied to any graph topology without the limiting assumptions and approximations of other methods. Our goal was to produce practical tools for mission problems such as cybersecurity, infrastructure monitoring, social network monitoring, and U.S. Army Research Laboratory work on translational neuroscience.

IN CONTEXT: THIS FY2019–21 PROJECT

- built on prior DoD line-funded research into the patterns and practices for future architectures and graph algorithms on future architectures
- benefited from collaboration with CMU experts in the fields of signal processing, scientific computing, compilers, computer architecture, machine learning, and mathematics
- aligned with the CMU SEI technical objective to bring capabilities through software that make new missions possible or improve the likelihood of success for existing missions

Principal Investigator

OREN WRIGHT
Research Scientist
Carnegie Mellon University
Software Engineering Institute

Graph Convolutional Neural Networks

Introduction

A growing number of DoD data problems are graph problems: the data from sources such as sensor feeds and web traffic require graphs to represent mathematically. Machine learning seems like a perfect tool for such datasets, but machine learning approaches for the irregularly structured data of graph problems are sharply limited.

We use graph signal processing formalisms to create new tools for graph convolutional neural networks (GCNNs), extending deep learning into the irregular world of graph problems.

Learning on Graphs

We address two classes of graph learning problems.

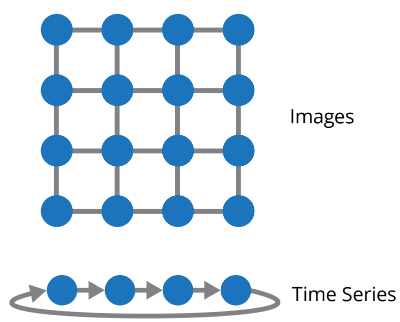
- **Node classification:** Predict information about unlabeled nodes in a graph, based on labeled nodes.
- **Graph classification:** Predict information about new graphs, based on labeled graphs. This is like image classification in computer vision.

Our Approach

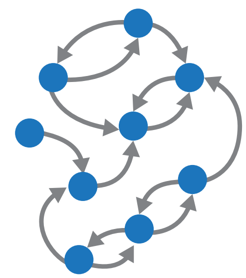
We built GCNNs using graph signal processing theory, yielding implementations that are computationally cheaper and empirically more accurate than other approaches. We also conducted experiments with pooling and sampling to further improve the state of the art and better understand when graph convolution is useful. These results will be detailed in forthcoming publications at the end of 2019.

We apply **graph signal processing** formalisms to create new tools for **deep learning on graphs.**

Regular Data Structures

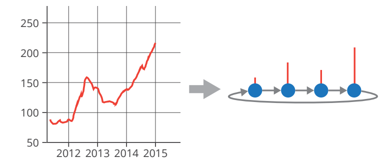


Irregular Data Structures

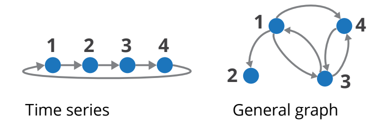


Social Networks
World Wide Web
Telecom Networks
Supply Chains
Biological Systems
Chemical Models
State Machines
Call Graphs
•••

Most deep learning techniques only work well on Euclidean data structures (i.e., data with a uniform, grid-like structure, as shown on the left) and don't extend to data with non-Euclidean structures (as shown on the right). GCNNs are an effort to extend the techniques that perform so well on regular data structures to irregular data structures.



Regular data structures like time series and images can be modeled as graphs, and the signal processing operations that work on these regular data structures can be considered special cases of a more generalized graph signal processing.



$$C = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

A graph interpretation of a time series shows that the circular shift matrix **C** is also the graph's adjacency matrix. This dual role of the adjacency matrix lets us develop a graph signal processing that can be applied to other graphs, such as the one on the right with its graph shift **A**.

$$\mathbf{G} = \sum_{k=0}^{K} g_k \mathbf{A}^k \qquad (1)$$

$$\mathbf{x}^{(\ell+1)} = \sigma\left(\mathbf{G}\mathbf{x}^{(\ell)} + \mathbf{b}\right) \qquad (2)$$

It can be shown in signal processing theory that convolution is polynomial in the shift, therefore, we can express the graph convolution **G** as a polynomial in the graph shift **A**, as shown in equation (1). This graph convolution can then be used as the kernel in a convolutional layer of a neural network, as shown in equation (2).

References

[Beller 2016] Beller, Moritz, et al. Analyzing the state of static analysis: A large-scale evaluation in open source software. 470-481. Proc. of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Vol. 1. March 2016.

[Delaitre 2015] Delaitre, Aurelien et al. Evaluating Bug Finders—Test and Measurement of Static Code Analyzers. Proc. of the 2015 IEEE/ACM 1st International Workshop on Complex Faults and Failures in Large Software Systems (COUFLESS). **https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=918370**

[Flynn 2016] Flynn, Lori. Prioritizing Alerts from Static Analysis with Classification Models. SEI Research Review, October 2016. **http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=474252**

[Flynn 2017] Flynn, Lori. Prioritizing Security Alerts: A DoD Case Study [blog post]. SEI Blog. January 2017. **https://insights.sei.cmu.edu/sei_blog/2017/01/prioritizing-security-alerts-a-dod-case-study.html**

[Friedberg 2017] I. Friedberg, K. et al. STPA-SafeSec: Safety and Security Analysis for cyber-physical systems. Journal of Information Security and Applications. 34. 183-196. **https://pure.qub.ac.uk/portal/files/132972897/Stpa.pdf**

[Heckman 2008] Heckman, Sarah & Williams, Laurie. On establishing a benchmark for evaluating static analysis alert prioritization and classification techniques. 41-50. Proc. of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. October 2008.

[Heckman 2008] Heckman, Sarah & Williams, Laurie. On establishing a benchmark for evaluating static analysis alert prioritization and classification techniques. 41-50. Proc. of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. October 2008.

[Pugh 2010] Ayewah, Nathaniel & Pugh, William. The google findbugs fixit. 241-252. Proc. of the 19th International Symposium on Software Testing and Analysis. July 2010.

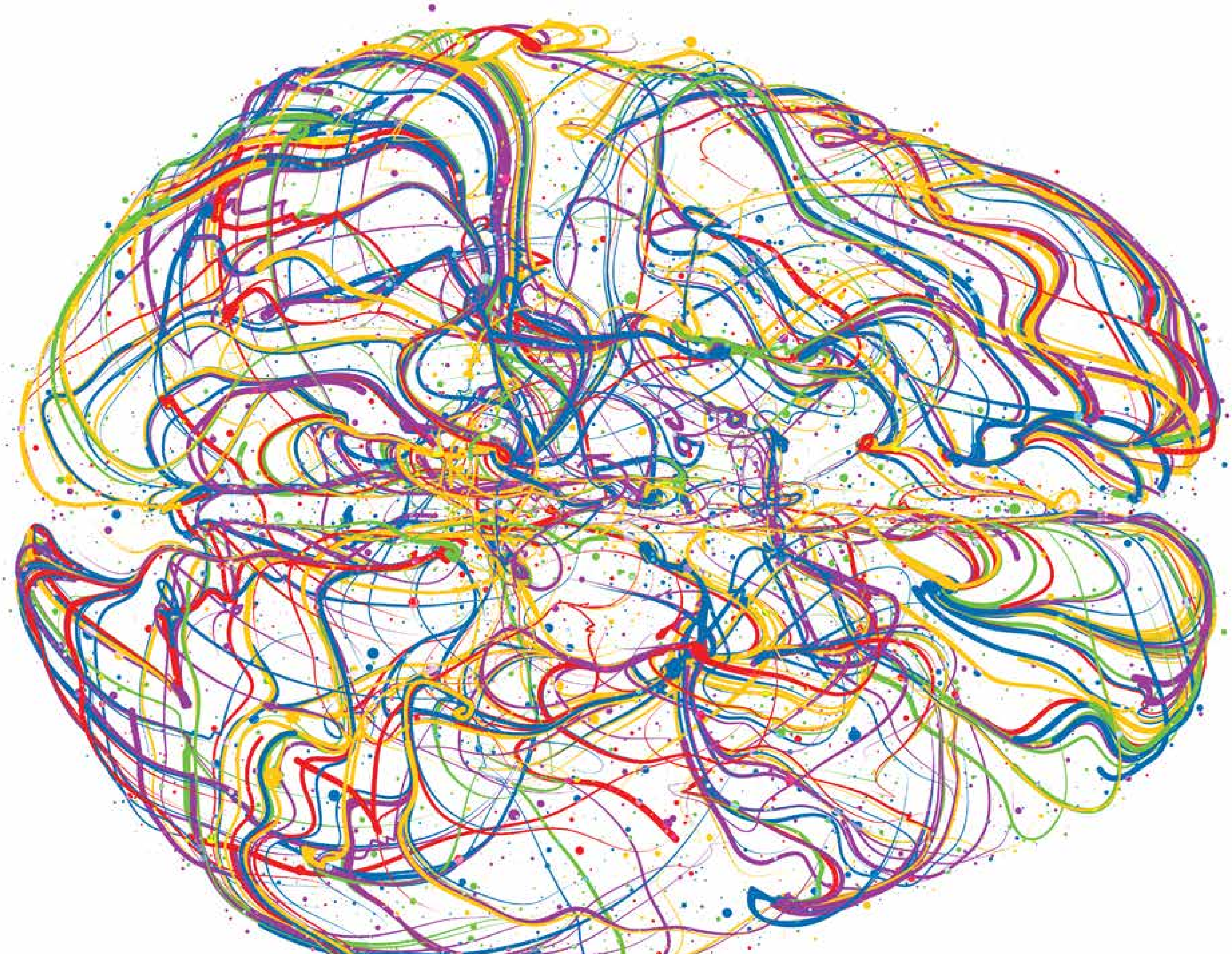
[Seligman 2016] Seligman, Lara. Interview: Air Force Chief Scientist Dr. Greg Zacharias. Defense News. February 20, 2016. **http://www.defensenews.com/story/defense/policy-budget/leaders/interviews/2016/02/20/interview-air-force-chief-scientist-dr-greg-zacharias/80424570/**

[Singh 2016] Singh, R. et al. Profiling hoax callers. 1-6. IEEE International Symposium on Technologies for Homeland Security. May 2016

[Spirtes 2010] Spirtes, Peter. Introduction to causal inference. Journal of Machine Learning Research 11. 1643-1662.

[Vasudevan 2016] Vasudevan, Amit; Chaki, Sagar; Maniatis, Petros; Jia, Limin; & Datta, Anupam. überspark: Enforcing verifiable object abstractions for automated compositional security analysis of a hypervisor. 87-104. Proc. of the 25th USENIX Security Symposium. August 2016.

Copyright



RESEARCH REVIEW

RESEARCH REVIEW

About Us

The Software Engineering Institute is a federally funded research and development center (FFRDC) that works with defense and government organizations, industry, and academia to advance the state of the art in software engineering and cybersecurity to benefit the public interest. Part of Carnegie Mellon University, the SEI is a national resource in pioneering emerging technologies, cybersecurity, software acquisition, and software lifecycle assurance.

Contact Us

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

412.268.5800 | 888.201.4479
sei.cmu.edu | info@sei.cmu.edu