Software Engineering Institute

# Results of SEI Independent Research and Development Projects

William Anderson, Archie Andrews, Nanette Brown, Cory Cohen, Christopher Craig, Tim Daly, Dionisio de Niz, Andres Diaz-Pace, Peter Feiler, David Fisher, David Gluch, Jeffrey Hansen, Jörgen Hansson, John Hudak, Karthik Lakshmanan, Richard Linger, Howard Lipson, Gabriel Moreno, Ed Morris, Onur Mutlu, Robert Nord, Ipek Ozkaya, Dan Plakosh, Mark Pleszkoch, Ragunathan (Raj) Rajkumar, Joe Seibel, Soumya Simanta, Charles Weinstock, and Lutz Wrage

**February 2011**

**TECHNICAL REPORT**
CMU/SEI-2011-TR-002
ESC-TR-2011-002



http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.



This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

# Table of Contents

# List of Figures

# List of Tables

# Abstract

The Software Engineering Institute (SEI) annually undertakes several independent research and development (IRAD) projects. These projects serve to (1) support feasibility studies investigating whether further work by the SEI would be of potential benefit and (2) support further exploratory work to determine whether there is sufficient value in eventually funding the feasibility study work as an SEI initiative. Projects are chosen based on their potential to mature and/or transition software engineering practices, develop information that will help in deciding whether further work is worth funding, and set new directions for SEI work. This report describes the IRAD projects that were conducted during fiscal year 2010 (October 2009 through September 2010).

# 1 Introduction

## 1.1 Purpose of the SEI Independent Research and Development Program

Software Engineering Institute (SEI) independent research and development (IRAD) funds are used in two ways: (1) to support feasibility studies investigating whether further work by the SEI would be of potential benefit and (2) to support further exploratory work to determine whether there is sufficient value in eventually funding the feasibility study work as an SEI initiative. It is anticipated that each year there will be three or four feasibility studies and that one or two of these studies will be further funded to lay the foundation for the work possibly becoming an initiative.

Feasibility studies are evaluated against the following criteria:

- mission criticality: To what extent is there a potentially dramatic increase in maturing and/or transitioning software engineering practices if work on the proposed topic yields positive results? What will the impact be on the Department of Defense (DoD)?
- sufficiency of study results: To what extent will information developed by the study help in deciding whether further work is worth funding?
- new directions: To what extent does the work set new directions as contrasted with building on current work? Ideally, the SEI seeks a mix of studies that build on current work and studies that set new directions.

## 1.2 Overview of IRAD Projects

The following research projects were undertaken in FY 2010:

- Trusted Computing in Extreme Adversarial Environments: Using Trusted Hardware as a Foundation for Cyber Security
  Archie Andrews (co-lead), Howard Lipson (co-lead), David Fisher, Jonathan McCune (CyLab, Carnegie Mellon University), and Anupam Datta (CyLab, Carnegie Mellon University)
- Communicating the Benefits of Architecting Within Agile Development
  Ipek Ozkaya (lead), Nanette Brown, and Robert Nord
- Multi-Perspective Reliability Modeling and Analysis for Cyber-Physical Systems
  John Hudak (lead), Jörgen Hansson (co-lead), David Gluch, Dionisio de Niz, Peter H. Feiler, Andres Diaz-Pace, Charles Weinstock, and Lutz Wrage
- Achieving Predictable Performance in Multi-Core Embedded Real-Time Systems
  Dionisio de Niz, Karthik Lakshmanan, Gabriel Moreno, Ragunathan (Raj) Rajkumar, Jeffrey Hansen, Christopher Craig, and Onur Mutlu.
- Automatic Generation of Hidden Markov Models for the Detection of Polymorphic and Metamorphic Malware
  Mark Pleszkoch (lead), Tim Daly, and Cory Cohen
- Advanced Technology for Test and Evaluation (T&E) of Embedded System Functionality and Security
  Richard Linger (lead) and Tim Daly

- An Investigation into the Feasibility of Tactical SOA
  Edwin Morris, Soumya Simanta, Dan Plakosh, William Anderson, and Joseph Seibel

These projects are summarized in this technical report.

# 2 Trusted Computing in Extreme Adversarial Environments: Using Trusted Hardware as a Foundation for Cyber Security

Archie Andrews, Howard Lipson, and David Fisher

## 2.1 Purpose

The SEI's stakeholders are increasingly operating in what we have termed extreme adversarial environments (i.e., highly resourced adversaries such as nation-states, well-funded terrorist organizations, and large criminal organizations). At the highest level of abstraction, an extreme adversarial environment (EAE) can be characterized by (1) one or more adversaries with nation-state-level resources (technical, economic, and other), (2) a complex, large-scale or ultra-large-scale information infrastructure based on open standards including internet technologies, and (3) very high consequences associated with a successful cyber attack.

Operating securely in extreme adversarial environments requires that we understand the range of characteristics of those environments and assess feasible approaches to maintain security and survivability properties under the severe condition these environments impose. We believe that a successful approach to operating securely in extreme adversarial environments will ultimately depend upon obtaining and leveraging a better understanding of the relationships among hardware, software, security, and trust.

The purpose of this IRAD study was to evaluate the promise and limitations of using trusted hardware as a foundation for achieving demonstrably high assurance of end-to-end security properties of applications executing in extreme adversarial environments.

## 2.2 Background

The exposure of our civilian and military critical infrastructures to the potential for very high consequence cyber attack from adversaries with nation-state level resources has never been greater. Yet the exposure continues to grow as our infrastructures evolve into ultra-large-systems employing new open standards and new internet-enabled technologies (whose security capabilities have not been thoroughly vetted, or vetted at all, for this more severe adversarial environment) in a mix with legacy systems (engineered for closed environments) whose designers never envisioned their use in open networking environments and which lack sufficient capabilities to operate securely in such environments.

A key characteristic of modern critical infrastructure is that the traditional logical and physical security perimeters (which also delineated boundaries of trust) have become far less distinct. In a sense, the notion of trust has been decoupled from well-defined logical and physical security perimeters, and so alternate (far more distributed) methods of establishing trust are essential for establishing and maintaining the security and survivability of modern systems. Survivability is dependent upon the ability to recognize, during an attack or accident, what parts of your system remain trustworthy [Lipson 1999]. Hence the ability to establish and maintain trust is essential for secure and survivable system operation.

Hardware-based assurance of trust is an emerging area of research and open standards development. The Trusted Computing Group (TCG) is an international industry standards group that has developed and defined open standards for hardware-enabled trusted computing. It produced and continues to evolve the standard (TPM v1.2, ISO/IEC standard 11889, dated August 2009) for the Trusted Platform Module (TPM) chip, which provides for secure storage and generation of cryptographic keys, platform authentication (based on each TPM having a unique cryptographic key), and remote attestation (the ability to assure a third party of some aspects of its trustworthy status, such as its TPM identity and software configuration). The TPM's hashing function creates a cryptographic digest (i.e., a nearly unforgeable summary) of code or documents that can enable the detection of tampering by comparing a current digest to a previous (known good) instance of the digest to see if anything has changed.

There is an emerging area of research on using TPMs (or other hardware support) as anchors of trust upon which to build attestations about the security properties of systems [Datta 2009, McCune 2009, McCune 2008]. TPM-enabled computing devices provide a set of initial building blocks for leveraging hardware to improve security. However, a significantly better understanding of the capabilities and limitations of applying the hardware features of the TPM to improve security is an important first step toward building trustworthy infrastructures and applications for secure and survivable operation in extreme adversarial environments.

## 2.3 Approach

This research required understanding the threat imposed by the extreme adversarial environment, assessment of feasible approaches to using the building blocks provided by the hardware-based trusted computing capabilities (e.g., the TPM), and an analysis of where and where not the solutions provided by commercially available components are appropriate.

The research relied on the SEI's CERT expertise in security, survivability, real-time control systems, and threat and vulnerability analysis to more fully characterize the nature of extreme adversarial environments. This characterization of extreme adversarial environment formed the basis for an investigation of the extent to which the existing characteristics and capabilities of TPMs can be used to establish and maintain trust so that high assurance of desired security properties can be obtained. The results of the initial characterization of the environment and its impact on potential implementation of hardware-based trusted computing allowed the researchers to characterize the capabilities and limitations of implementations of today's trusted computing platforms for use in such environments.

## 2.4 Collaborations

This research was a result of collaboration between CERT researchers experienced in the concepts of survivable systems, modeling, and threat analysis and CyLab experts in hardware-based security properties. This collaborative arrangement permitted CERT researchers to quickly ascertain the critical operating principles of the hardware enabled security provided by industry groups such as the Trusted Computing Group.

## 2.5 Evaluation Criteria

This IRAD in trusted computing accomplished the following:

- provided a detailed characterization of the properties of extreme adversarial environments
- determined, characterized, and described the capabilities and limitations of a hardware-based trusted computing platform (specifically, the TPM) for improving security and survivability in extreme adversarial environments
- laid the groundwork for future research that will explore and exploit the concepts of trust and trustworthiness and provide a scientific basis for understanding the relationships among hardware, software, security, and trust. The ultimate goal is to establish an engineering foundation for designing and implementing hardware-based trusted computing platforms that support secure and survivable operation in adversarial environments.

## 2.6 Results

### 2.6.1 Extreme Adversarial Environment

Our original conjecture was that a description of an extreme adversarial environment would provide insight sufficient to enable us to separate the activities of such an adversary into areas where a trust-enabled platform could mitigate risks and identify those activities where the trust model was inadequate to thwart a determined adversary.

An extreme adversarial environment is characterized by the presence of intelligent adversaries with the following resources, capabilities, and goals:

- highly adaptive and intelligent in the face of defensive measures
- the will to keep coming despite setbacks (if attacking a chosen target rather than simply a target of opportunity)
- purchasing power (people, capabilities [such as bot armies and other capabilities from the cyber-criminal underground], and equipment)
- blackmail and coercion
- extensive, persistent intelligence gathering (human and cyber, general information technology [IT] and domain-specific)
- forensic suppression capabilities (to inhibit detection and attribution)
- high performance computing assets support rapid data fusion and analysis (fusing public and private sources amplifies intelligence gathering capabilities )
- multi-realm coordinated attacks (e.g., physical and cyber) maximize impact
- high degree of technical proficiency and engineering expertise (general IT and deep knowledge of the technology, processes, and personnel characteristics of the specific target domain)
- hacking and malware expertise (zero day attack capability based on deep expertise in vulnerability discovery, vulnerability exploitation, malware creation, theft or forgery of key material for digital certificates)
- facilities for manufacturing/fabrication/counterfeiting

- major insider threat (infiltrate target or co-opt members via persuasion, bribery, or coercion)
- targets chosen for strategic importance (not necessarily targets of opportunity and financial gain)
- convoluted, highly distributed attack paths prevent traceback and attribution (e.g., use of multiple stepping stones to thwart tracking)
- complex game-theoretic strategic planning (highly adaptive, game-theoretic optimal strategies for attack and exploitation, such as IP exfiltration)

Given this extent of capabilities and intentions, it was impractical to partition the potential activities of an extreme adversary into distinct classes of adversarial environments, such that specific security approaches would be effective for specific classes of these environments. We were compelled to conclude that in the case of a determined adversary with extreme skills and resources, no reasonable application of the traditional security model (i.e., the "fortress model") could provide a feasible defense. The fortress model assumes that the defender can successfully construct a defense that completely separates critical operations from the persistent attacker and successfully defend against every attack. A more extreme and useful assumption is that operation in a malicious environment is inevitable. Given that assumption, the question becomes whether it is feasible to isolate critical operations from that malicious environment to the extent necessary to enable trusted operations. The notion of trusted, in this context, implies the ability to isolate a computing operation from adverse effects, whether expected or unanticipated, that could yield undesirable results. The team's research turned to consider what degree of trust in the isolation of critical operations is enabled by a hardware-based trusted computing approach that relies on a trusted security platform, and whether that degree of trust would yield a sufficiently trustworthy computing environment.

### 2.6.2 Trusted Security Platform

A trusted security platform is one in which confidence in the secured operations, applications integrity, and isolation from malicious actions is enabled by an embedded TPM. The TPM is a simple, passive, integrated circuit chip (see Figure 2-1) intended to serve as a hardware root of trust for trusted infrastructure leading to software applications that are trusted. The TPM is readily available having been installed in an estimated 250,000,000 platforms by 2010.[1] It is an inexpensive enabler for credential storage, device identity, and trusted applications and provides security capabilities that cannot be provided by software alone with the same degree of confidence. The trusted security platform and the TPM in particular offer a level of security capability unachievable in software alone. The TPM derives both trust and trustworthiness from its simple passive character. A more complex device, an active one with general purpose computing capabilities, or one that shares registers with a CPU could not engender similar levels of trust.

[1] From remarks by Steven K. Sprague at the Cyber NSF Trusted Information Workshop at Carnegie Mellon University, Pittsburgh, PA; June 7-10, 2010.

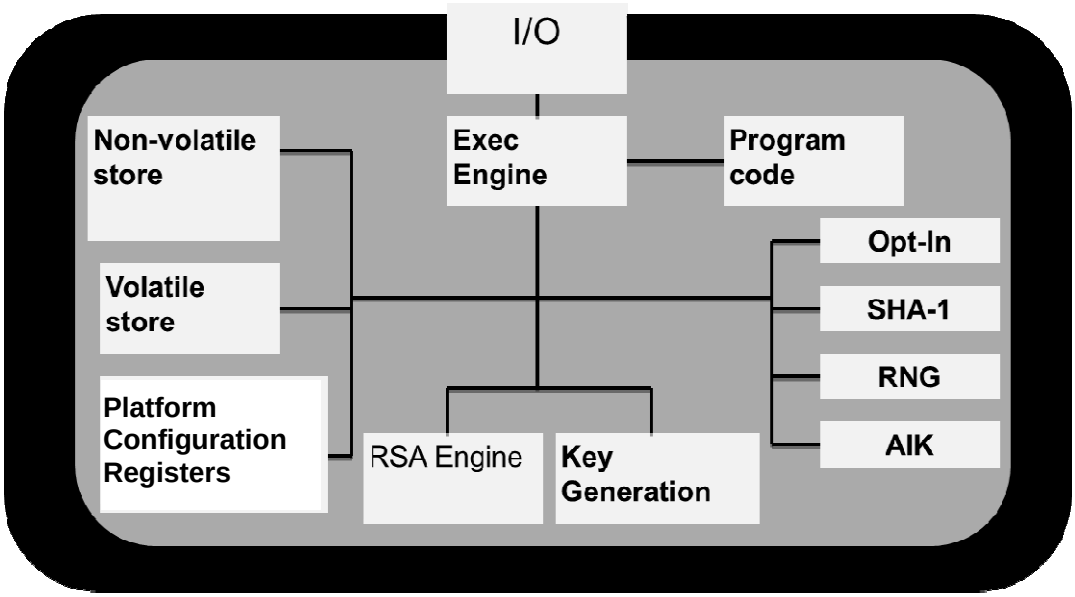



*Figure 2-1: Components of a Trusted Platform Module*

Despite these desirable attributes, in practice the TPM is rarely used even when the chip is present (less than 1 percent of the installed TPM base has been activated[2]). It is rarely used in the development of computational or network infrastructure and has had limited market penetration in end-user applications. To understand why, we examined the methods recommended and used for extending trust from the TPM to higher levels of application software. Building provably correct chains of trust starting from the basic input/output system (BIOS) and extending through several levels of operating system and application software is theoretically sound, but in practice is a tedious and brittle process that must be redone whenever any software along the way changes. More practical methods are used (such as a Dynamic Root of Trust [McCune 2008]) but with increased vulnerabilities and often lower levels of trust. The measurement technique used in conjunction with the TPM is vulnerable because it cannot detect changes made and restored between measurements. Neither does measurement provide protection for mutable storage.

A viable business model for exploiting trusted platform technology in a general purpose platform has not yet emerged. However, as the number of deployed TPMs increases, so do the opportunities for commercial products dependent on the existence of deployed TPMs. A key appears to be integration and support by operating systems and application use of the resulting application programming interfaces (APIs). As a practical matter, applications developers and end users will not leverage the TPM unless its functionality is easily accessible. End users cannot be expected to develop the chains of trusted software required at the operating system level. More encouraging are increased use of TPM chips in certain dedicated security applications such as Microsoft's BitLocker, storage of PKI private keys and other credentials (e.g., biometric identifiers), and potentially secure machine identities on sensitive networks. Part of the problem may be that without a large installed base of TPMs, there is little incentive for application developers, and without developers' demand, there is little incentive for the operating system (OS) to support them or for

[2] Ibid.

more systems to install them. The implication to the Department of Defense and others with a currently large installed base of TPMs is that the majority of those already deployed will unlikely be used without a critical mass of installations that triggers a broader market of applications and OS support. PricewaterhouseCoopers' recent decision to enable TPM-based credentials for 150,000 users may be an indication of this trend [Messmer 2010].

There is also a need for hardware support for security beyond that provided by currently available trusted platform devices. The potential and realized benefits of the TPM derive from the ability to ensure integrity of information, processes, or identity either by physical isolation (e.g., key storage in the TPM) or logical isolation (e.g., encrypted communication). Hardware support for isolation of storage and communication is needed at many levels, including central processing unit (CPU) register sharing through task switches, shared caches, uninitialized portions of memory pages, and direct memory access (DMA) access. Hardware could assist operating systems by isolating applications, eliminating security vulnerabilities inherent in current CPU architectures, and providing user-level security functions that are easily accessible through APIs. Hardware mechanisms to support dynamic roots of trust, to eliminate software intervention at the BIOS and the lowest levels of the OS, to provide immutable storage as an alternative to measurement, and to facilitate logically atomic sequences of operations have the promise to make chains of trust less brittle and more trustworthy.

Our effort also looked at the character of trust and trustworthiness. The business model that has been successful for engaging the TPM is to provide end-to-end security products that happen to leverage TPMs. Other models have not been widely adopted and if fully realized would result only in security products and "feature sets," not necessarily in more secure or trustworthy applications. A business model of potentially greater effectiveness is the use of the TPM internal to applications, products, or systems to maintain and preserve trustworthiness that could otherwise be undermined by security flaws in lower level infrastructure. The TPM could also be used to develop underlying infrastructures that are themselves more secure and trustworthy. Research is needed to define and assess business strategies for broader and more effective exploitation of hardware-based trusted platforms. As the installed base of TPMs continues to grow, there are ever-increasing opportunities for private industry to offer applications, operating systems, and infrastructures that are inherently more trustworthy by leveraging the TPM.

Trust is of critical importance in all human activity. Without trust, we can accomplish nothing. With unjustified trust, nothing can be adequately assured, safe, or efficient. Automated systems and networks impose additional problems of trust, but do not traditionally provide adequate support for trust. Automated support for trust is essential for effective use of automated systems and networks. The TPM and other hardware-based trust mechanisms are a step in the right direction but inadequate in current practice. While they provide automated support for certain security aspects of trust, issues of trust go far beyond security. There is a need for investigation and understanding of the potential role of technology in supporting these other aspects of trust.

Support for trust will require more rigorous understanding of trust and trustworthiness, effective strategies for achieving trustworthiness and assessing trust, and development of automated tools for trust. Research in these areas will likely borrow from other domains with overlapping concerns. These domains include, most conspicuously, security, survivability, dependability, emergent behavior, and modeling and simulation.

Research is needed to develop trust technologies applicable to automated systems. Trust technology has the potential to overcome the limitations of existing approaches for achieving survivability, security, and dependability. An effective trust technology will focus on mission fulfillment, survivability, and evolution of automated and networked systems. It will employ security methods but only when and where they are cost effectively needed. It will embrace many of the quality objectives of dependability but with greater adaptability and realism. It will seek practical cooperative solutions that are appropriate for competitive and adversarial environments. It will focus on end-to-end solutions specialized to particular needs. It will emulate and adapt proven trust methods from everyday life. Our ultimate goal is to provide the capability to build and operate critical automated systems that will behave in a sufficiently trustworthy manner to consistently fulfill their missions, even when these systems are built and operated in extreme adversarial environments.

## 2.7 Publications and Presentations

An analysis of the extreme adversarial environment is in working draft form.[3] A white paper, *Trust and Trusted Computing Platforms*, is complete [Fisher 2010]. The paper will be the basis for submission to the Embedded Systems Conference, May 2011 and the 4th International Conference on Trust and Trustworthy Computing, June 2011.

### 2.7.1 Bibliography

Challener, David; Yoder, K.; Catherman, R.; Saford, D.; & Van Doom, L. *A Practical Guide to Trusted Computing*. IBM Press, 2007 (ISBN 0-13-239842-7).

Grawrock, David. *Dynamics of a Trusted Platform: A Building Block Approach*. Intel Press, 2008 (ISBN1-934053-08-2).

Pearson, Siani, ed.; Blabcheff, B.; Chen, L.; Plaquin, D.; & Proudler, G. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall, 2003 (ISBN 0-13-009220-7).

Trusted Computing Group. *Summary of Features Under Consideration for the Next Generation of TPM.* Trusted Computing Group, 2009. www.trustedcomputinggroup.org/resources/summary_of_features_under_consideration_for_the_next_generation_of_tpm

### 2.7.2 References

**[Datta 2009]**
Datta, Anupam; Franklin, Jason; Garg, Deepak; & Kaynar, Dilsun. "A Logic of Secure Systems and Its Application to Trusted Computing," 221-236. *Proceedings of the 30th IEEE Symposium on Security and Privacy*, 2009. Oakland, California, May 2009. IEEE Computer Society, 2009.

**[Fisher 2010]**
Fisher, David & McCune, Jonathan. *Trust and Trusted Computing Platforms.* Software Engineering Institute White Paper, September 2010.

[3] Lipson, Howard. *Extreme Adversarial Environments—Characteristics and Implications for Trusted Computing*. White Paper (Working Draft), November 2010.

**[Lipson 1999]**
Lipson, Howard & Fisher, David. "Survivability—A New Technical and Business Perspective on Security," 33-39. *Proceedings of the 1999 Workshop on New Security Paradigms.* Caledon Hills, Ontario, Canada, September 22-24, 1999. Association for Computing Machinery, 1999. www.cert.org/archive/pdf/busperspec.pdf

**[McCune 2008]**
McCune, J. M.; Parnoy, Bryan; Perrigy, Adrian; Reiteryz, Michael K.; & Isozaki, Hiroshi. "Flicker: An Execution Infrastructure for TCB Minimization," 315-328. *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*. Glasgow, Scotland, U.K., April 1-4, 2008. The Association for Computing Machinery, 2008.

**[McCune 2009]**
McCune, Jonathan M.; Qu, Ning; Li, Yanlin; Datta, Anupam; Gligor, Virgil; & Perrig, Adrian. *Efficient TCB Reduction and Attestation* (CMU-CyLab-09-003). Carnegie Mellon University, 2009. www.cylab.cmu.edu/research/techreports/2009/tr_cylab09003.html

**[Messmer 2010]**
Messmer, Ellen. "PwC Lauds Trusted Platform Module for Strong Authentication: Firm Migrating 150,000 Users to TPM-Based Storage of Private Keys." *Network World.* September 15, 2010. http://www.networkworld.com/news/2010/091510-trusted-platform-authentication.html?page=1

# 3 Communicating the Benefits of Architecting Within Agile Development

Ipek Ozkaya, Nanette Brown, and Robert Nord

## 3.1 Purpose

Having a common understanding of the structure of a software system through its architecture is essential, especially in large-scale systems, for guiding the development effort, meeting the customer needs, focusing on improvement efforts, and maintaining the software over the years. Having disciplined project management that assures working software that delivers its intended customer needs while having the agility to respond to change is critical for success. Yet, asserting the importance of both architecture-centric techniques and agile software development principles raises many questions, especially ones that scrutinize their coexistence. In particular, how to use both architecture-centric practices with agile software development in practice isn't obvious when it comes to large-scale projects.

While agile methods are appealing to practitioners, and getting attention in industry, software development organizations are facing difficulties applying these methods to projects of increasing scale. One of the key issues we have identified in large industrial settings is the lack of enough attention to architectural design and development, often under the misconception that the goal of architecture practices is to produce architecture documents that often end up being shelfware. One of the tenets of agile software methods is to focus on delivering observable benefits to the end users through working software, early and often. This is unfortunately often achieved by focusing on the "skin" of the system, and deferring or completely ignoring some of the deeper architectural issues. Taking shortcuts, projects incur "technical debt" that continues to expand, causing some projects to collapse under its weight [Cunningham 1992]. Unfortunately, architectural tasks are not considered among those that deliver immediate utility to the customer.

Our goal is to address the question of "what is the benefit of software architecture?" In order to answer this question practically, we are interested in developing heuristics and methods to support tactical and strategic agile release planning by incorporating architectural aspects that are of critical importance to project management. We envision the heuristics and methods to serve the following purposes:

- better communicate the customer-observed benefit of architecture in different software development settings, especially within agile development
- improve project management and release planning by accounting for architectural tasks in a way similar to how capabilities are accounted for

## 3.2 Background

Agile software development methods, such as XP, Scrum, FDD (Feature Driven Development), Lean Software development, and the like, have had significant impact over the last five to eight years on industrial software development practices. However, there is increasing confusion about the role and importance of a system's software architecture in the context of agile approaches.

Advocates of the crucial role of architecture in achieving quality goals of large-scale software-intensive systems are skeptical of the scalability of any development approach that does not pay sufficient attention to architectural aspects of such systems, especially in domains like automotive, telecommunication, finance, and medical devices. But the proponents of agile approaches often perceive the up-front design and evaluation of architecture as being of little benefit to the customers of a system. There is a growing interest in separating the facts from myths about the necessity, importance, advantages and disadvantages of the co-existence of agile and architectural approaches. European software initiatives have been paying close attention to this lately. For example, the Flexi initiative has been collecting data from observing agile projects and how architects respond to agile project development environments [Flexi 2009]. *IEEE Software* recently published a special issue dedicated to understanding the relationship of agile development and architecture [IEEE 2010].

Government and government contractors are also looking closer into adapting agile practices to deliver capabilities quicker [Cohan 2007, Crowe 2009, Lapham 2010]. It is imperative that such organizations and teams obtain correct guidance. Dispensing with architecture-related design and implementation in large scale, multiyear projects will result in detrimental failures in the long run, while giving the false impression of achieving progress on the surface during initial release cycles by focusing on customer-facing user stories.

Individual stories cannot be regarded in isolation. Stories have dependencies on other stories. In *Software by Numbers*, Denne and Cleland-Huang use the term "greedy algorithm" to refer to a prioritization scheme that focuses strictly on implementing the story with the highest immediate value [Denne 2004]. They point out that, at times, higher value stories may depend upon (i.e., require prior implementation of) lower value stories. Thus, truly optimizing value to the user requires development teams to look ahead and anticipate future needs.

Similarly, stories have dependencies upon the architectural elements of the system. These dependencies exist regardless of the stability of the domain or the maturity of the technology. They exist regardless of whether the system is in its initial development stages or has been deployed and has been in the field for several years. The ability to identify and analyze architectural dependencies and incorporate dependency awareness into a responsive development model exemplifies the notion of bridging architectural practices and agile development.

In his book *Scaling Software Agility*, Leffingwell notes that neither XP nor Scrum nor FDD pays much attention to software architecture, which at best will gradually emerge from a succession of refactorings in the context of such agile development processes [Leffingwell 2007]. This will work in many cases, as noted by Highsmith when enough of a software architecture can be put in place in the first two or three iterations [Highsmith 2004]. But it can also lead to major failures, such as the ones we have witnessed in practice, involving large projects. Agile works well in contexts where the developers have experience and there is a mental model of a workable architectural framework that is context for their refactorings. Achieving such a shared mental model often requires either spending time architecting the system or relying on a previous mental model [Kazman 2004, Nord 2004].

The delivery of software functionality in increments has been studied in the area of release planning [Ruhe 2005, Saliu 2005]. Capabilities or features are usually the selling units provided to the stakeholders. Features are prioritized so that the most important ones are delivered in the first re-

leases and the benefits of the new system are gained early. Less important features are deferred to subsequent releases. Therefore, if the schedule or budget is not sufficient, the least important features are the ones more likely to be omitted. It is argued that this approach allows for better responsiveness to changes or additions to requirements. However, deciding when to deliver features in which iteration and deciding the order of features and releases are problems that require careful planning. Architectural concerns are at most implicit in existing techniques for release planning both in typical project management techniques and within agile project management practices, such as Scrum.

Lindgren and associates draw attention to the lack of architecture and architectural considerations during release planning activities based on the case studies they have conducted [Lindgren 2008]. They found that architecture-centric practices are not included in the most commonly used system evolution practices. They summarize these as

- product management generally has low architectural awareness
- there is no method for how to balance investments in quality improvements versus capabilities growth
- the role of "gut-feeling" and lobbying is lower in companies that involve the software architect

### 3.3 Approach

Our approach has the following thrusts:

1. Search the literature and investigate tactics for aspects of release planning including
   - allocating costs and benefits to invisible features
   - dealing with multiple iterations leading to a single release
   - allocating cost and/or benefit to different prioritization schemas
2. Build an agile architecture release planning framework of capability, benefit, work item, cost, time, and architecture.
3. Validate the model and the tactics above with fabricated data and real data.

### 3.4 Collaborations

We established research collaborations with Dr. Philippe Kruchten and his master's student Erin Lim from the University of British Columbia. During the project we also collaborated with Manuel Pais and his supervisor Dr. Eduardo Miranda from the Master of Software Engineering Program, School of Computer Science at Carnegie Mellon University.

### 3.5 Evaluation Criteria

Bridging agile development and architecture and scaling agile development for large-scale systems has been addressed both by agile development and architecture-centric engineering communities, as discussed in Section 3.2. However, the guidance to date is mostly on how to balance the development workflow by sparing some time for architecting without much quantifiable analysis techniques to support how to make such allocations. In order to improve project management and release planning in how architectural tasks are accounted for, we focused our work on creat-

ing quantifiable techniques in better communicating the customer-observed benefit of architecture. Since key criteria of improvement is being able to increase value to the customer, our evaluation focus on addressing the following two questions.

1. Can architecture capabilities be incorporated into agile release planning by taking a value-based approach?
2. Are there techniques to assist in evaluating different prioritization tactics (architecture first, versus capabilities first)?

We summarize how we address these questions in the next section.

### 3.6 Results

We developed a release planning dashboard in order to incorporate architecture into agile release planning, which typically focuses on planning for implementation and release of user stories.

Figure 3-1 shows a release planning board that represents the typical heuristics used within the Agile community for release planning. Desired stakeholder capabilities are represented as "user stories." These user stories are allocated to iterations in order of their priority to the end user.

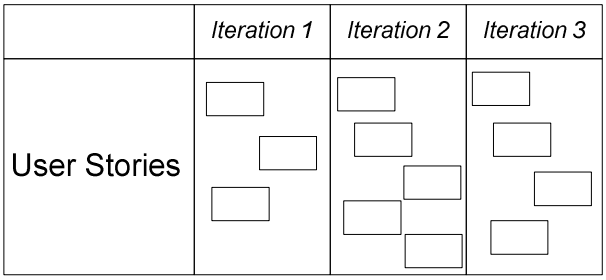



*Figure 3-1: Agile Iteration Planning—Focus on User Stories*

Figure 3-2 shows an enhanced release planning board that incorporates planning for development of the underlying software architecture. In addition to selecting stories to be developed within each iteration, the team identifies the architectural elements that must be implemented to support them. This version of the release planning board also incorporates a "technical research" activity, recognizing that architectural development frequently requires investigation and analysis of alternate approaches. Finally, the term "capabilities" has been used in place of user stories, reflecting a need to consider non-functional, quality attribute requirements, as well as the need to incorporate requirements across a broad range of stakeholders. For effectively bringing architecture and agile development together, dependencies between capabilities and architectural elements need to be identified not only to fulfill the current release, but to plan for future releases as well.

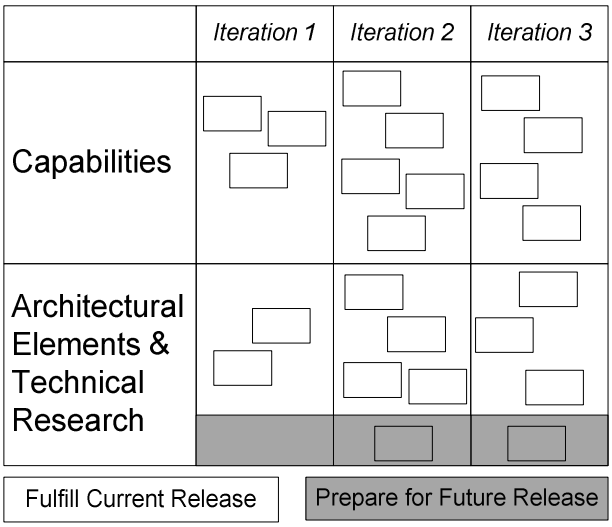



*Figure 3-2: Agile Iteration Planning—Enhanced With Architectural Elements*

Dependency management has been studied extensively at the level of code artifacts. Applying dependency management at the architecture level is beginning to show promising results due to increasingly effective tool support. These metrics can be extracted from the architecture, represented in the form of a dependency structure matrix (DSM). The DSM is a compact representation that lists all constituent subsystems/activities and the corresponding information exchange and dependency patterns. Domain mapping matrices (DMMs) augment DSM analyses and can be used to represent the dependencies between capabilities and architectural elements.

Metrics associated with dependency also provide data for inferring the likely costs of change propagation, especially when dependencies between architectural elements are also considered. One such example is discussed by Carriere and associates; the value of re-architecting decisions needed to be understood to determine if the expense to implement them was justified [Carriere 2010].

To demonstrate use of such metrics in balancing agility and architecture, we conducted a case study. The study involved an architectural path analysis of a model problem to identify the ideal implementation sequence based on

- path #1: maximizing value for the end user
- path #2: minimizing implementation cost

The model problem, Management Station Lite, is a centralized unit for managing a set of automated systems inside a building (air conditioning, access control, lighting, etc.) [Sangwan 2008]. The two paths are summarized in Figure 3-5.

In order to reason about alternative implementation paths for a given agile project, we took into account the dependencies between customer requirements (for example, the capability of "sending a command to lower the building temperature" requires having knowledge and therefore access to the capability of "querying the current temperature").

Customer requirements prioritization and relationships between requirements and architectural elements were core inputs for the path analysis. Relationships were analyzed using dependency

management techniques. Figure 3-3 shows a domain mapping matrix of this problem taking into account user stories and architecturally significant requirements as test cases mapped to key architectural elements.

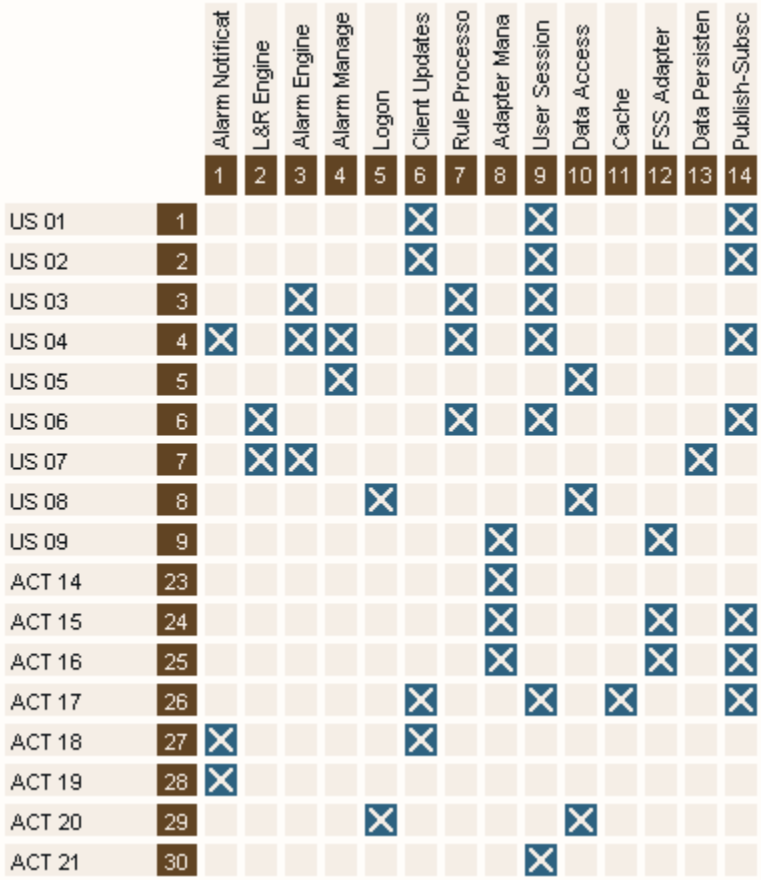
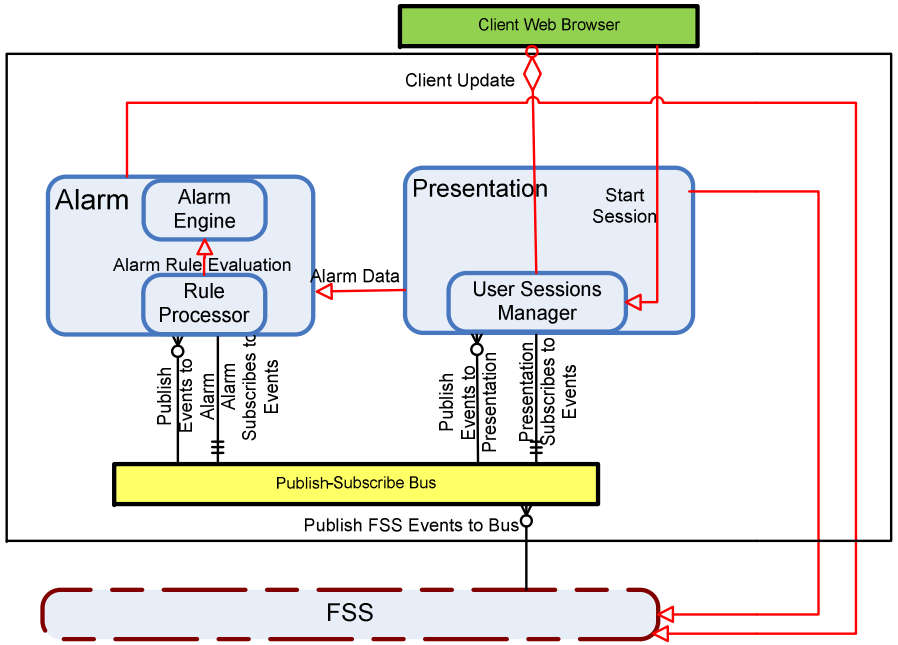
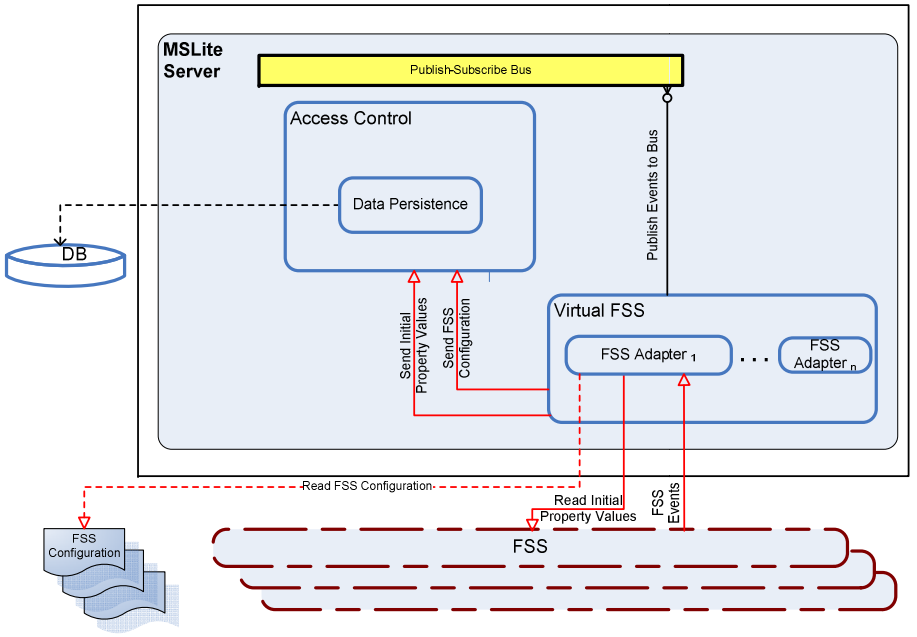



*Figure 3-3: Dependencies Between Requirements and Architectural Elements*

Results confirmed that path #1 delivered value to the user at a rate twice as fast as path #2. However, path #1 incurred additional rework costs (around 14 percent of implementation cost) due to delayed implementation of architectural components supporting systemic quality requirements (security and performance).

The two paths show differing architectural quality starting on step 1. As can be observed in Figure 3-4, the evolving design in path 1 has extensive dependencies that propagate through the system even in step 1.




| | Path #1 | Propagation Cost |
|---|---|---|
| Step 1 | $root<br>03 Alarm Engine 1<br>06 Client Updates 2<br>07 Rule Processor 3<br>09 User Sessions Manager 4<br>14 Publish-Subscribe Bus 5 | 32% |

| | Path #2 | Propagation Cost |
|---|---|---|
| Step 1 | $root<br>12 FSS Adapter 1<br>13 Data Persistence 2<br>14 Publish-Subscribe Bus 3 | 0% |

*Figure 3-4: Comparison of Architectures After First Step*

Comparing the two paths as shown in Figure 3-5, in path #1 the increase in value occurs more rapidly. For instance at step 2 the system provides already more than 50 percent of the total value to the user. In contrast, in path #2 at step 2, there is still no value accrued. In a real project context, these steps typically map to iterations, in some cases releases.

This is because path #2 prioritizes building the structural elements that provide the foundation for those quality requirements that cut across the entire system:

- publish-subscribe bus for performance and modifiability in general
- data persistence for keeping the state of the system and user preferences
- field device adapter for modifiability
- data access for security
- cache for performance

The cost also increases steeply in path #1, amounting to nearly 80 percent by the end of step 2 alone. This contrasts with a steady increase in cost for path #2.

We can identify two main reasons for this variation. The first is that the release identification for both paths did not take into consideration the estimated effort to develop the architectural elements. They were based on the requirements value for path 1 and on the number of dependencies for path 2. We can see for instance that step 1 in path #1 has an implementation cost of 30 while step 4 has an implementation cost of 2.

The second reason is that path #1 incurs a change cost starting from release 1 as implemented architectural elements depend on elements that are to be implemented in later releases.

Due to the added change cost the total cost for path #1 is about 14 percent higher than for path #2. This figure would be aggravated if we considered the fact that the scope for step 1 in path #1 was too large and the likely element to move to a later step without losing functionality was the publish-subscribe bus (could be replaced by call-return). Because three elements (user sessions manager, alarm engine, rule processor) in step 1 already depended on publish-subscribe bus then the change cost would increase further, assuming the net present value would not offset this difference.

In an agile project this type of analysis can raise awareness of the cost associated with deciding on an implementation path solely focused on delivering value. If it is not acceptable for the customer, then the team and the customer can decide together on which value/cost trade-offs they are willing to accept.

| Value | | Step 1 | Step 2 | Step 3 | Step 4 | Step 5 |
|---|---|---|---|---|---|---|
| **Path #1** | Value | 75 | 132 | 175 | 216 | 243 |
| | % | 30.86% | 54.32% | 72.02% | 88.89% | 100% |
| | Cost | 38.64 | 58.25 | 67.8 | 69.68 | 84.2 |
| | % | 52.22% | 78.72% | 91.62% | 94.16% | 113.78% |
| | **Contribution** | **30.86 – 52.22 = -21.35** | **54.32 – 78.72 = -24.40** | **72.02 – 91.62 = -19.60** | **88.89 – 94.16 = -5.27** | **100 – 113.78 = -13.78** |
| **Path #2** | Value | 0 | 0 | 47 | 126 | 243 |
| | % | 0% | 0% | 19.34% | 51.85% | 100% |
| | Cost | 21 | 33 | 43 | 58 | 74 |
| | % | 28.38% | 44.60% | 58.11% | 78.38% | 100% |
| | **Contribution** | **0 – 28.38 = -28.38** | **0 – 44.60 = -44.60** | **19.34 – 58.11 = -38.77** | **51.85 – 78.38 = -26.53** | **100 – 100 = 0** |

*Figure 3-5: Summary Comparison of Two Paths in the Case Study*

A focus on architecture is not in opposition to agile values and principles. As our case study and release planning board demonstrate, the focus on end user stories should be expanded to address the broader topic of capabilities, including quality attribute requirements and a diverse range of stakeholders. Dependency analysis practices can be used to facilitate a "just-in-time" approach to building out the architectural infrastructure.

## 3.7 Publications and Presentations

During the research, the team authored a number of publications and presentations:

Brown, N.; Nord, R.; & Ozkaya, I. "Enabling Agility Through Architecture." *CrossTalk 23*, 6 (November/December 2010): 12-17.

Brown, N.; Cai, Y.; Guo, Y.; Kazman, R.; Kim, M.; Kruchten, P.; Lim, E.; MacCormack, A.; Nord, R.; Ozkaya, I.; Sangwan, R.; Seaman, C.; Sullivan, K.; & Zazworka, N. "Managing Technical Debt in Software-Reliant Systems," 47-51. *Proceedings of FSE/SDP Workshop on the Future of Software Engineering Research*, Santa Fe, NM, November 2010. The Association for computing Machinery, 2010.

Brown, N. *Crossing the Great Divide*, Software Engineering Institute Webinar, April 22, 2010 http://www.sei.cmu.edu/library/abstracts/webinars/Agile-Development-and-Software-Architecture-Crossing-the-Great-Divide.cfm

Pais, M.; Miranda, E.; & Brown, N. *Architectural Value in Agile Projects: A Study on Architectural Path Analysis.* Independent Study White Paper, 2010.

Brown, N.; Kruchten, P.; Lim, E.; Nord, R.; & Ozkaya, I. *Hard Choices Game Explained*. Software Engineering Institute, Carnegie Mellon University White Paper, 2010. www.sei.cmu.edu/library/abstracts/whitepapers/hard-choices-game-explained-v1-0.cfm

Brown, N.; Kruchten, P.; Lim, E.; Nord, R.; & Ozkaya, I. *Hard Choices Board Game*. The Software Engineering Institute, Carnegie Mellon University, 2010. http://www.sei.cmu.edu/architecture/tools/hardchoices/

### 3.7.1 References


**[Carriere 2010]**
Carriere, J. Kazman, R. & Ozkaya, I. "A Cost-Benefit Framework for Making Architectural Decisions in a Business Context," Vol. 2, 149-157. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering.* Cape Town, South Africa, May 2010. The Association for Computing Machinery, 2010.

**[Cohan 2007]**
Cohan, Seam. "Successful Integration of Agile Development Techniques Within DISA," 255-261. *Proceedings of AGILE 2007*. Washington, D.C., August 2007. IEEE Computer Society, 2007.

**[Crowe 2009]**
Crowe, P. & Cloutier, R. "Evolutionary Capabilities Developed and Fielded in Nine Months." *CrossTalk 22*, 4 (May/June 2009): 15-19. www.crosstalkonline.org/storage/issue-archives/2009/200905/200905-0-Issue.pdf

**[Cunningham 1992]**
Cunningham, W. "The WyCash Portfolio Management System," 29-30. *Addendum to the Proceedings of OOPSLA '92*. Vancouver, British Columbia, Canada, October 1992. The Association for Computing Machinery, 1992.

**[Denne 2004]**
Denne, M., & Cleland-Huang, J. *Software by Numbers: Low-Risk, High-Return Development*. Prentice Hall, 2004 (ISBN: 978-0131407282).

**[Flexi 2009]**
Flexi Initiative, 2009. The Flexi Consortium, Information Technology For European Advancement (ITEA).http://www.flexi-itea2.org/index.php

**[Highsmith 2004]**
Highsmith, J. *Agile Project Management.* Addison-Wesley, 2004 (ISBN: 978-0321219770).

**[IEEE 2010]**
Abrahamsson, Pekka, Ali Babar, Muhammad, & Kruchten, Philippe. "Agility and Architecture: Can They Coexist?" *IEEE Software 27*, 2 (March/April, 2010): 16-22.

**[Kazman 2004]**
Kazman, R.; Kruchten, P.; Nord, R. L.; & Tomayko, J. E. *Integrating Software Architecture-Centric Methods into the Rational Unified Process* (CMU/SEI-2004-TR-011). Software Engineering Institute, Carnegie Mellon University, 2004.
www.sei.cmu.edu/library/abstracts/reports/04tr011.cfm

**[Lapham 2010]**
Lapham, M. A.; Williams, R. C.; Hammons, C.; Burton, D.; & Schenker, F. *Considerations for Using Agile in DoD Acquisition* (CMU/SEI-2010-TN-002). Software Engineering Institute, Carnegie Mellon University, 2010. www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm

**[Leffingwell 2007]**
Leffingwell, D. *Scaling Software Agility: Best Practices for Large Enterprises.* Addison-Wesley, 2007 (ISBN: 978-0321458193).

**[Lindgren 2008]**
Lindgren, M.; Norstrom, C.; Wall, A.; & Land, R. "Importance of Software Architecture During Release Planning," 253-256. *Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*. Vancouver, British Columbia, Canada. IEEE Computer Society, 2008.

**[Nord 2004]**
Nord, R. L.; Tomayko, J. E.; & Wojcik, R. *Integrating Software-Architecture-Centric Methods into Extreme Programming (XP)* (CMU/SEI-2004-TN-036). Software Engineering Institute, Carnegie Mellon University, 2004. www.sei.cmu.edu/library/abstracts/reports/04tn036.cfm

**[Ruhe 2005]**
Ruhe, G. & Saliu, M. O. "The Art And Science Of Software Release Planning." *IEEE Software, 22,* 6 (November/December 2005): 47-53.

**[Saliu 2005]**
Saliu, O. & Ruhe, G. "Software Release Planning for Evolving Systems." *Innovations in Systems and Software Engineering: A NASA Journal 1*, 2 (September 2005): 189-204.

**[Sangwan 2008]**
Sangwan, R.; Neill, C.; El Houda, Z.; & Bass, M. "Integrating Software Architecture-Centric Methods into Object-Oriented Analysis and Design." *Journal of Systems and Software 81*, 5 (May 2008): 727-746.

# 4 Multi-Perspective Reliability Modeling and Analysis for Cyber-Physical Systems

John Hudak, Jörgen Hansson, David Gluch, Dionisio de Niz, Peter H. Feiler, Andres Diaz-Pace, Charles Weinstock, and Lutz Wrage

## 4.1 Purpose

Cyber-physical systems are characterized by the tight conjoint of and coordination between computational and physical resources [Lee 2008]. The academic and industrial community had recognized model-based approaches as one of the most promising ways forward to produce reliable and predictable cyber-physical systems (CPS). These approaches use models to describe designs at a high level that can be analyzed for specific properties, such as timing, power consumption, and heat dissipation. Because the properties of the system are the result of the combination of software and hardware interacting closely with the physical environment, the system needs to be evaluated from multiple perspectives. For example, consider an automotive engine. From a physical point of view we need to ensure the heat generated by the mechanical parts of the systems is properly dissipated. From a hardware perspective, we need to ensure that the sensors can sample the physical variables (e.g., number of revolutions per second of the crankshaft) frequently enough to capture the changes of interest. From the software perspective, we need to ensure that the software is able to finish its computation at the proper periodic intervals.

The properties from these perspectives are modeled and verified in different models. These models are composed out of constructs tuned for the properties of the perspective of interest. Unfortunately *these constructs abstract away the interactions between the different perspectives.* For instance, dissipation models (say in SysML) of the physical aspect of the system may ignore heat variations due to the processor, which is separately modeled in VHSIC Hardware Description Language (VHDL). In turn the hardware models ignore the mixture of instructions that can produce variations in power consumption (e.g., CPU-intensive instructions produce more heat than memory-access instructions).

The rationale for the separate modeling approaches is driven by the combined need to model at a fine level of detail, the typical engineering and reduction approach of decomposing a complex problem into problems of more reasonable size and focus. The specificity and distinct characteristics with respect to domains and concerns have motivated and resulted in a number of design and modeling approaches tailored to the problem at hand (all modeling approaches and the model artifacts developed have intentionality). However, unanticipated effects of separated design approaches or changes often defer discovery too late in the life cycle, impeding larger costs. Only in the best case do the independent models created in the traditional approach reflect the same system architecture. Furthermore, any change to the architecture during its lifetime requires each model to be updated and verified. As challenging as that is, the need to consistently proliferate changes in one analysis to others adds difficulty.

The isolation of the models of the different perspectives of a CPS prevents proper verification of system-wide properties, often causing severe oversights of their interacting properties; these oversights come out as failures in the integration testing and in fielded systems.

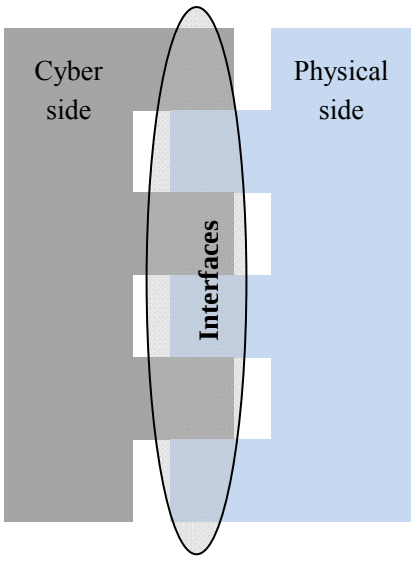
In this IRAD project, we have focused our attention on the challenges specific to inter-perspective interaction modeling and analysis that can propagate the effects of the properties in one perspective into the models of the other perspectives. An integrative and inter-perspective modeling and analysis framework can significantly reduce, or even prevent, critical property interactions oversights.

### 4.2 Background

CPSs are characterized by the tight relationship between software and hardware resources. This relationship implies that both the "cyber" and "physical" aspects of the system have to be integrated and coordinated (the majority of these systems are also system-of-systems—given their massively distributed nature—and highly interactive among the nodes). It is generally observed that both aspects are of equal importance and tightly coupled. Thus, it can be argued that it is not easy to isolate or abstract by means of a well-defined interface that encapsulates internal details/implementation. Traditionally, in many software systems the hardware is segregated to a platform layer (even platform-independent layers). Correspondingly, in heavily hardware-oriented systems, such as many embedded systems, the software is represented by well-defined components. CPSs are in this regard different, given there is a more significant flow of interactions, including stronger coupling, between what the cyber and physical parts.

A CPS goes beyond a "conventional" embedded system because of the possibility of having several embedded systems communicating with each other and relying on software function. Aspects related to connectivity include time, space, and locality. This makes the coordination of the parts and the analysis of quality of signal (QoS), for example, challenging.

To fully realize the potential of CPSs, and effectively develop CPSs, the "cyber" and "physical" parts need to be more strongly coordinated. Such coordination needs to consider real-time constraints, control-theoretic aspects, heterogeneity of environments, scale and connectivity (i.e., multiple nodes connected and operating together), as well matching of assumptions between the "cyber" and "physical" sides.




This weakens encapsulation and modularity in the sense that hardware details are increasingly exposed to the software and vice versa. Thus, there is no distinct orthogonal separation of concerns between the hardware and software. This suggests that concepts for coupling hardware and software resources, (e.g., interfaces), need to be examined further.

Interestingly, while coupling between hardware and software could be considered to be significantly tightened and integrated, there are also elements of an increased autonomy of (individual) parts in CPSs compared to the typical embedded system. For example, the CPSs to a larger degree exhibit distributed control and have increased requirements on being adaptable due to the dynamic and uncertain environments they act in; the systems have to be context-aware.

Observing the state of the art and the literature, we observe that development processes in the CPS area strongly emphasize co-analysis and co-design practices for hardware and software. That is, software and hardware parts have to be analyzed, designed, implemented, and validated together from the early development stages onward, and especially if there are critical quality attribute requirements concerning reliability or dependability.

CPSs represent a new class of systems with new challenges. Here we briefly discuss some of the research challenges we deem critical.

*The ability to achieve predictable and tunable behavior of CPS is critical.*
Ideally, analysis should be compositional, allowing component-level analyses to be conducted, the analyzed components to be assembled together according to specific rules, and the analyses similarly to be aggregated to ensure predictable overall system behavior. Currently, there are quality attribute analysis techniques for reasoning about either the hardware or software parts of a system. However, extending (or connecting) these techniques to reason about the compound system is difficult. The integration of heterogeneous cyber and physical components in a cost-effective way is problematic.

As already alluded to, *CPSs pose a multi-perspective problem related to the people and models used to design systems*. The cyber part will likely be taken care of by software developers, while the physical part will be likely taken care of by hardware and system engineers. Unfortunately, these two groups of people often have different mindsets, use different notations and abstractions, perform different types of analyses, rely on different assumptions, and so forth. Conciliating these parties and their (partial) perspective of a CPS constitutes an important challenge today. Some issues include

- connecting different abstractions and models that have to be somehow connected; this includes unification and harmonization of semantics across models
- co-analysis and co-design activities (e.g., tracing problems across different models) to facilitate early detection of mismatches, faults, and problems
- integrated debugging to (1) avoid interference with hardware functions while debugging, (2) reproduce possible faults and unpredictable behaviors
- determining function locality (i.e., allocation of functions to hardware, software, or both)

*Since interfaces do not decouple hardware and software distinctly, hardware and software can affect each as either part can change or evolve.* Adaptation and evolution of CPS is thus complex. For instance, imagine the scenario in which some platform-related resources are changed due to new features or maintenance reasons. In this context, how does one reason about the effects of that evolution on the hardware resources so as to make the right changes successfully without compromising critical behaviors? This is further exacerbated by hardware and software resources that are inherently heterogeneous in CPS settings.

Model-based techniques seem to be a viable approach to deal with the CPS challenges regarding co-analysis, co-design, and development. Early analyses should reduce testing efforts later and produce better-quality systems. However, tool support is indispensable to fulfill these promises in practice.

Approaches to modeling CPS include ad-hoc integration of software and hardware and physical languages and analyses. Common combinations include using MathLab and Simulink that integrate physical models with control-oriented simulations; Simulink/StateFlow; Modelica and UML/SysML to provide support for modeling and simulation of continuous dynamics via a UML profile (ModelicaML); and UML and Simulink for embedded systems.

A perceived limitation of the modeling approaches for CPS is that they either develop ad hoc integrations between pairs of existing languages, or they try to provide a "uniform" modeling framework for domain-specific languages. However, in spite of all these modeling approaches, capabilities for semantic composition and composed analyses are still less developed.

## 4.3 Approach

We present a model-based analysis framework (henceforth referred to as the framework) for cyber-physical systems (CPSs) that provides the information required across engineering perspectives and enables the discovery of the potential failures early in the development life cycle. In addition, within a composite model, we demonstrate the use of integrated analysis flows that enable the end-to-end analysis of errors across perspectives and mitigate the potential for their occurrence.

In defining a framework for cyber-physical systems reliability modeling and analysis, we consider engineering activities that support the development life cycle of a cyber-physical system, including design, implementation, and assurance. Since CPSs often have safety-critical, fault tolerant, or high dependability requirements, we consider the impact of these requirements on the phases of the development life cycle. Section 4.3.1 describes a model problem and technical motivation for the work. Sections 4.3.2 through 4.3.4 discuss the technical concepts developed to reason about CPS from a systems modeling perspective. Section 4.4 describes an implementation approach to operationalize the concepts. Section 4.5 describes the application of the framework to the model problem, and Section 4.6 describes the results and future work.

### 4.3.1 Model Problem

This model problem includes networked autonomous systems coordinating their actions. In this example, we consider fault/failure models and reliability approaches for each of the autonomous subsystems and for the complete system. Known and published architectures and designs for unmanned and autonomous vehicles and robots are used.

Each of the Autonomous Airborne Vehicles (AAVs) is capable of missions without human intervention such that they depart a support facility, join the team, execute missions, and return to a support facility. The role of the AAVs is to identify, locate, and in some cases transport personnel to treatment units. The medical robots are able to medically assess and move wounded/injured personnel onto an autonomous ground vehicle (AGV). They are transported by the AGVs; however, they can move autonomously for short distances (<1 kilometer) away from an AGV or treatment center. Furthermore, treatment centers are staffed by medical personnel, who off-load

the wounded and injured and provide treatment. Support facilities provide maintenance for the AAVs, AGVs, and robots. Once a specified area is defined, the system must define mission strategies and procedures, allocate and coordinate resources, and ensure mission completion.

For our purposes, an AGV contains cruise control and stability control subsystems. An AGV is provided waypoints for navigation according to mission goal planning. The control systems on an AGV use the waypoints to establish a route based on map data and determine speed set points that are given to the onboard cruise control systems. The stability control system is affected by environmental factors such as ice on the road and/or unpaved dirt roads and trails. Within the course of getting to its destination, a situation can occur that points out how particular faults can cross modeling boundaries and can be identified using the framework approach.

Potential errors in modeling of the interaction between the cruise control and the electronic stability control, when modeled both in Simulink and AADL, are considered.

The electronic stability control (ESC) brakes individual wheels to control the yaw of a vehicle. Cruise control (CC), on the other hand, accelerates the vehicle to take it to a target speed (if it gets below this target speed). Whenever the ESC kicks in, it needs to disable CC to avoid braking (by ESC) while accelerating (by CC).

Both the ESC and the CC are typically modeled by control engineers in Simulink (Figures 4-1 and 4-2) to evaluate the stability of their control. In particular, for the ESC they evaluate the effectiveness to avoid over and under steering, and—for the CC—its effectiveness to keep a vehicle at the specified speed.

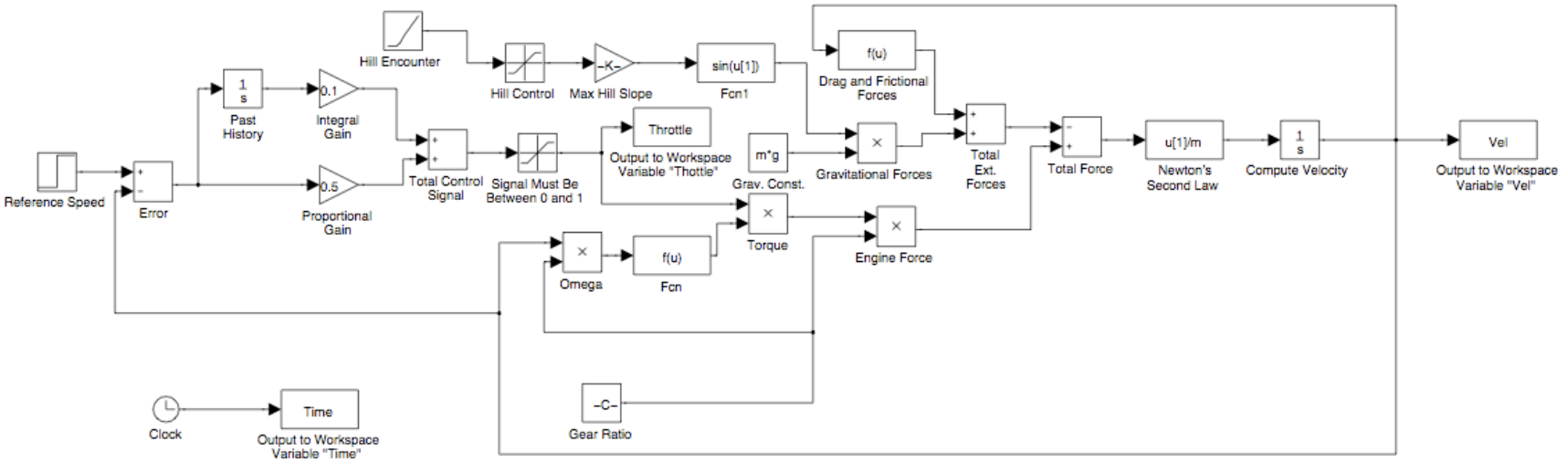



*Figure 4-1: Sample Simulink Model for a Cruise Control System*

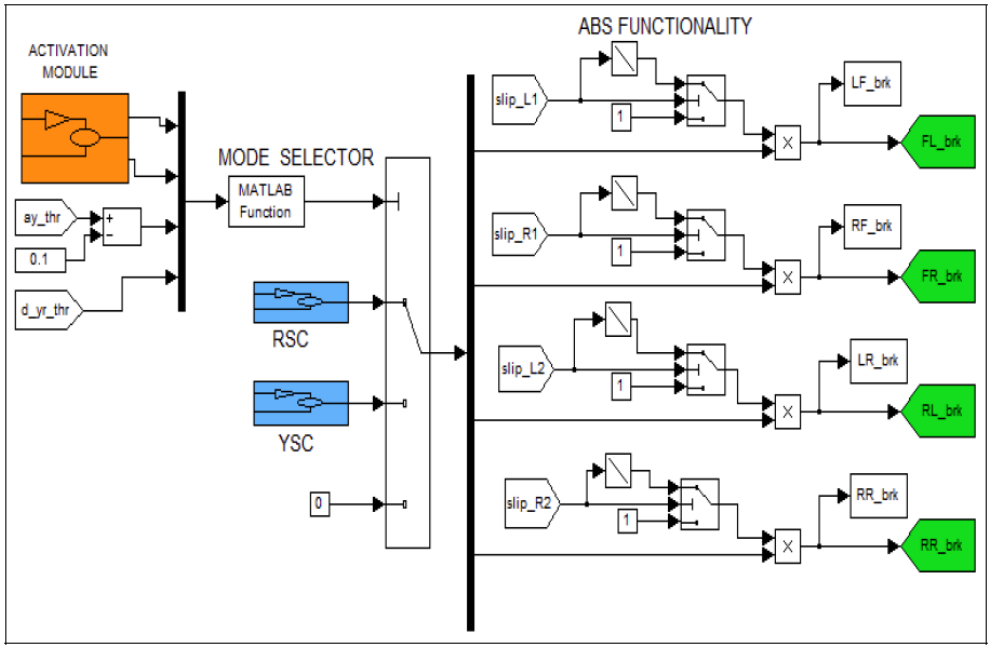



*Figure 4-2: Sample Simulink Model for an ESC System [Kinjawadekar 2009]*

The Simulink models in Figure 4-1 and Figure 4-2 are used by the control engineers to evaluate the stability of the control systems, as mentioned earlier. However, in the final implementation these subsystems need to be coordinated to disable the CC whenever the ESC needs to brake to preserve the stability of the vehicle. Unfortunately, in the end, such coordination is implemented in software where variations in the execution due to scheduling policies or the allocation of the subsystems to different processor can create synchronization defects.

To evaluate the timing properties of the integration of the system, engineers can build execution models using a modeling language such as AADL. A sample model of this integration is presented in Figure 4-3, and it depicts part of the execution model of the ESC and CC integration.

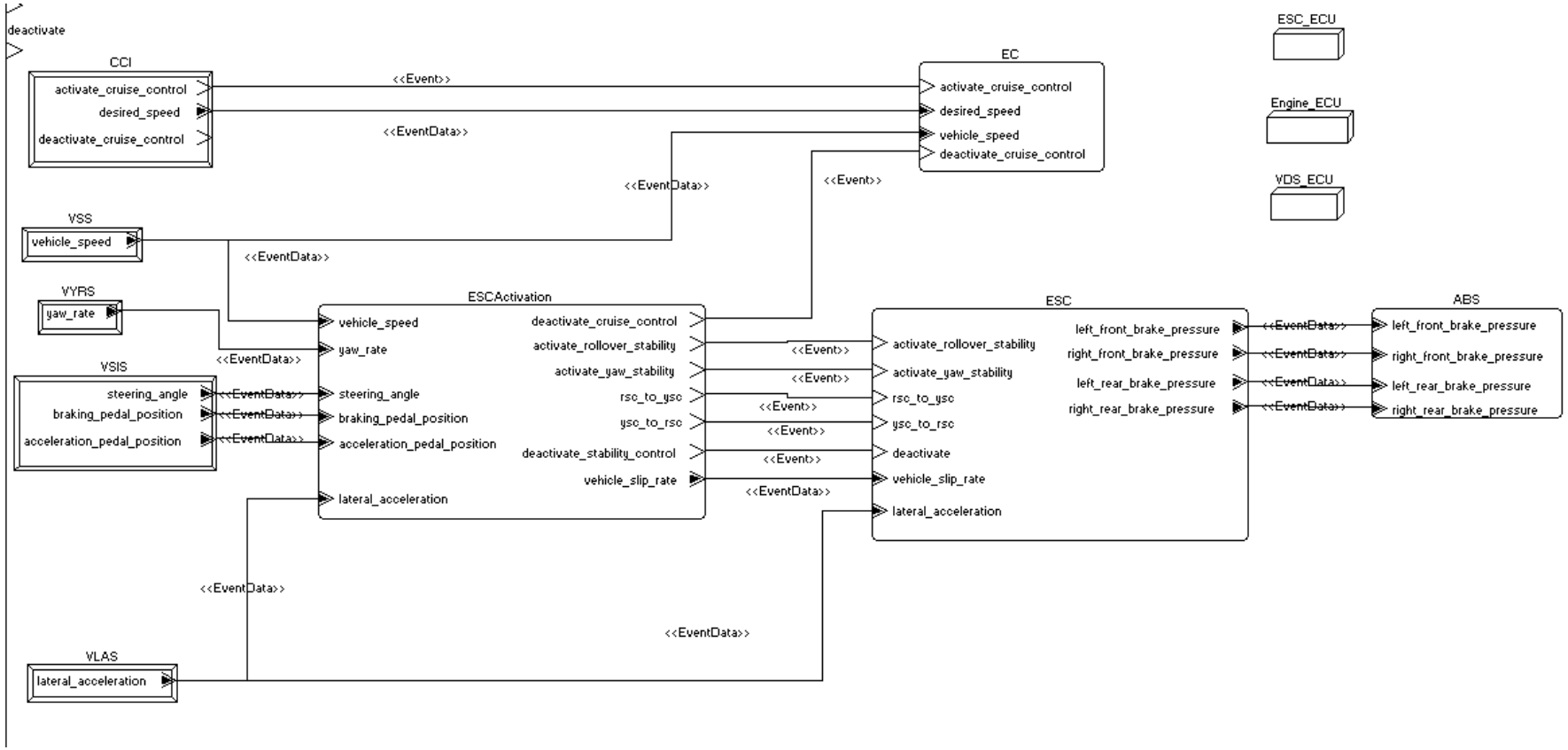



*Figure 4-3: AADL Execution Model of the Integration of the CC and ESC Subsystems*

Figure 4-4 illustrates the CC and ESC component along with an ESCActivation component that send two activation signals to the ESC model (*activate_rollover_stability* and *activate_yaw_stability*) to activate two different modes of the stability control and a deactivate signal to the Engine Control (EC) module, where the CC module resides. Inside the ESC model, three modes are used: *ysc_active* (Yaw Stability Control), *rsc_active* (Rollover Stability Control), and *inactive*.

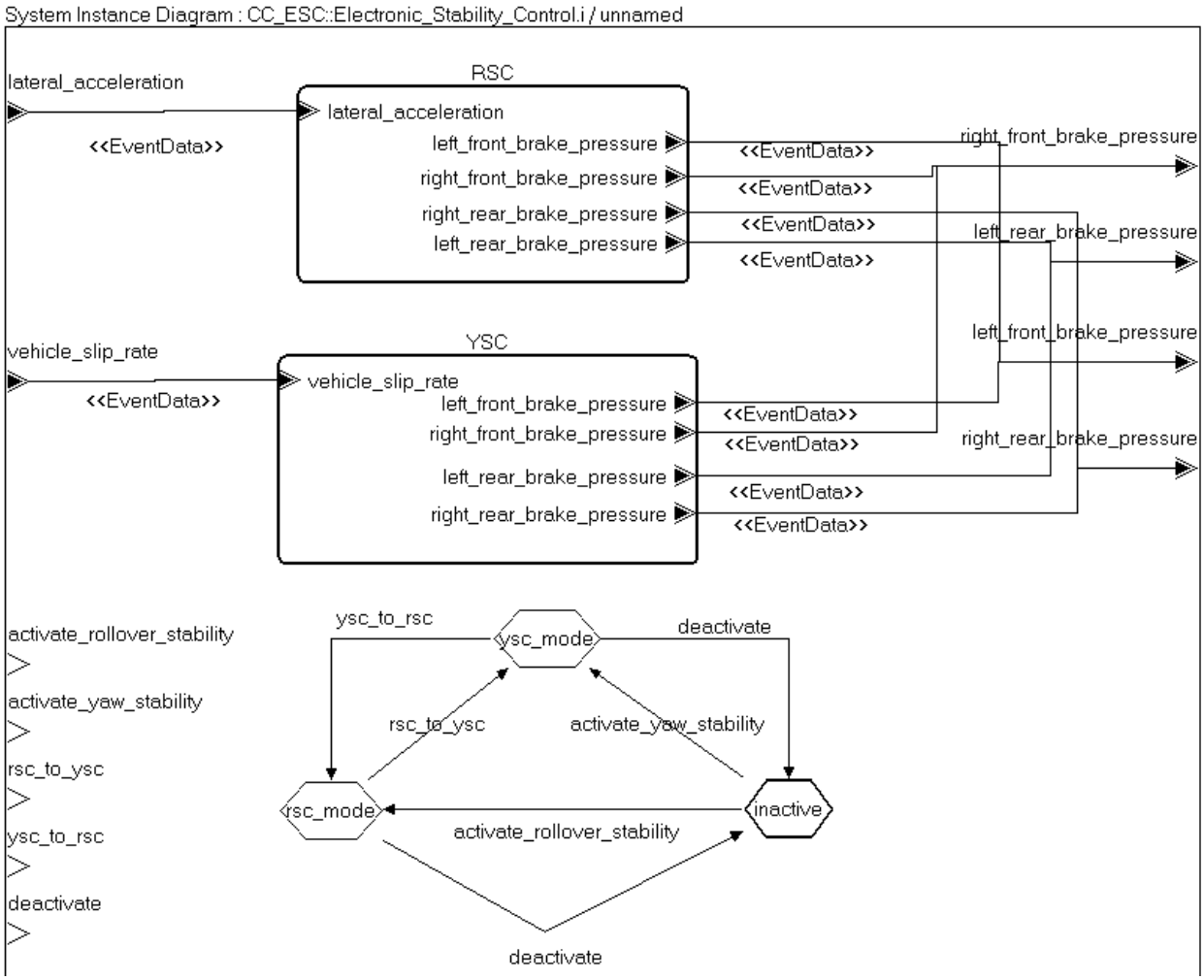



*Figure 4-4: ESC Modes*

On the other hand, the CC module has two modes: *cruise_control_on* and *cruise_control_off*. This is depicted in Figure 4-5.

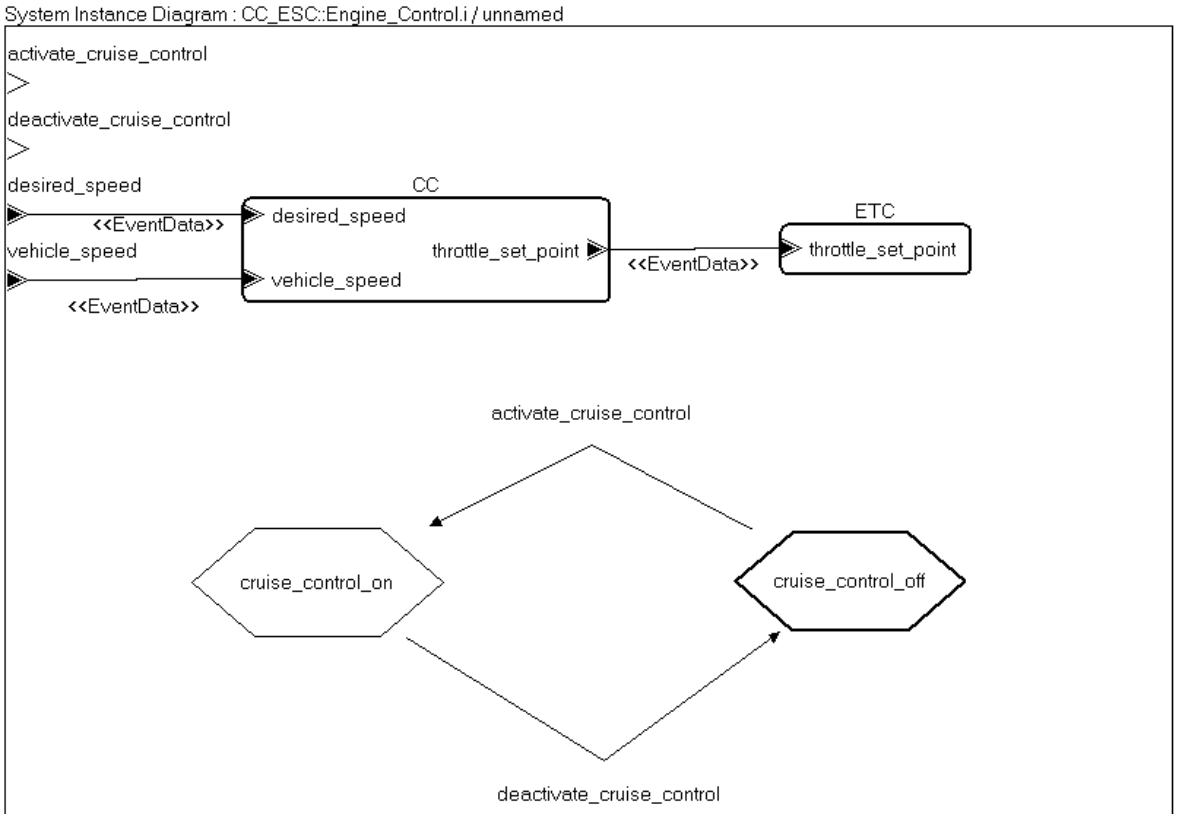



*Figure 4-5: CC Modes*

Whenever the ESC module in Figure 4-4 switches out of the inactive mode the CC module in Figure 4-5 needs to switch *cruise_control_off* instantaneously. If this is not the case, then the combined behavior of these subsystems can reach a situation where the ESC subsystem brakes while at the same time the CC subsystem accelerates to reach its target speed. This can lead to a vehicle being out of control at the same time that the ESC is trying to prevent such an event.

Two possible situations that can make the ESC and CC mode transitions not happen at the same time are (1) scheduling policy and (2) deploying to different processors. In the first case, the CC may be prevented by another task while the ESC is executing the mode transition. In such a case the CC could make the transition after the ESC mode change was executed. On the other hand, when the two subsystems are deployed on different processors, they can receive the signal to make the mode transition at different times or they can even miss a message (or receive a re-transmission late). In this case again, the transitions would happen at a different time.

#### 4.3.1.1 Example of State-of-the-Art in Modeling in Practice

Connections in Simulink represent data communication between ports where ports are variables in a component. The connection of two ports then represents the transfer of the value from one port to another. Because Simulink models a continuous time system, such a connection is in fact a data transfer (or value copying).

In AADL, on the other hand, there are different ways to represent a connection with different consequences. Specifically, connections can be represented as event connections or data connections. Event connections queue events in the port and can "remember" events that were received when the component was busy processing other events or not executing. A data connection, on the other hand, represents a "sampling" of the value that was put in the sending port. As a result, if the value changes two or more times while the receiving component was busy or not executing, then the receiving component will only be able to observe (sample) the last value (last change) of the component.

The combination of data connection and continuous time execution deletes details of the discrete computation that needs to be explicit in AADL. As a result, the Simulink model cannot encode enough detail to decide on the correct modeling in AADL if no additional information is provided.

As an example, consider the signal to stop that is sent from the ESC to the CC. A Simulink model can be successfully verified for a safe and stable operation. At the same time, this model can be translated into AADL and, due to the lack of additional information, the connection can be represented with a non-immediate data connection. The resulting system can also be verified for schedulability. Unfortunately, the stop signal sent through the data connection can effectively be ignored, if the scheduler decides not to run the CC when the ESC sends the stop signal and later is turned off (a Boolean signal).

### 4.3.2 Technical Foundation: Identifying Perspectives, Boundaries, and Couplings

Within cyber-physical systems, we focus on the boundaries that define the separation of engineering perspectives. Example engineering perspectives include mechanical, electrical, structural, software, and system. For this work, these are grouped into computational hardware perspectives such as digital systems (e.g., processors, ASIC), analog to digital conversion, and sensors; two sets of physical perspectives such as mechanical, electrical, and structural—the physical platform perspective and the physical environment; software perspectives such as code structure, runtime, and configuration control; and a systems perspective that includes abstractions of the physical, software, and computational hardware perspectives. The systems perspective captures the application domain-specific goals and requirements in abstract representations, often as component-connection architecture models. These are shown in Figure 4-6.

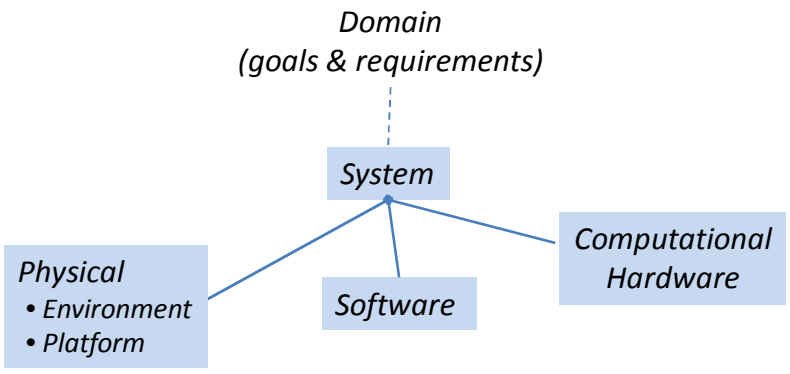



*Figure 4-6: Cyber-System Perspectives*

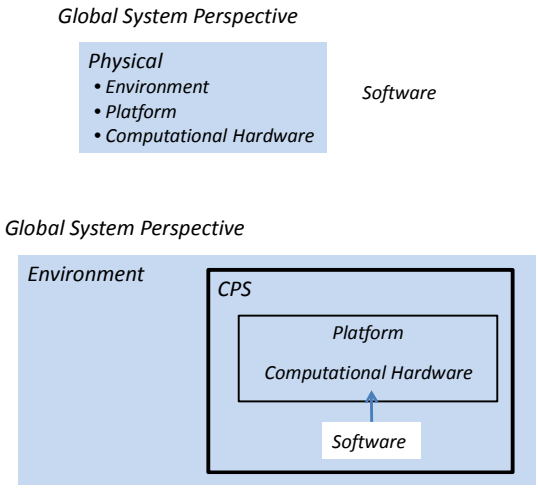
In this work, we explore the boundaries between perspectives as reflected in the integration of modeling and analysis tools from each perspective, addressing their impact on the semantics of reliability modeling and analysis at these boundaries. In doing so, we consider the interactions and the invariants (constraints) maintained across boundaries—highlighting semantic misunderstandings (e.g., misinterpretation, miscommunication), assumptions (e.g., generic and domain and design-specific), and lost information (e.g., through abstraction, non-relevance, omission).

The most common underlying assumption for accidents and the basis for accident models is that they are the result of an uncontrolled and undesired release of energy or interference in the normal flow of energy. However, in modern systems there is an increased reliance on information (cyber aspects) creating the potential for loss of information or incorrect information resulting in unacceptable physical, scientific, or financial losses.[4] This is especially the case for cyber-physical systems.

### 4.3.3 Identification of Perspectives in CPS

We further elaborate four perspectives for cyber-physical systems: three physical perspectives (environment, platform, and computational hardware) and the software perspective. These are shown in Figure 4-7 where physical perspectives are shaded. The environment is the physical realm in which a CPS exists. The platform consists of the physical components that comprise a CPS, excluding the computational hardware. Computational hardware includes all of the hardware that supports the execution of software.[5] The global system perspective is a composite of the four perspectives. Within each perspective there are multiple modeling and analysis representations. For example, in the physical perspectives there are mechanical, thermal, and electrical modeling and analysis representations.

The lower portion of Figure 4-7 shows the containment relationships of the implementations in each of the perspectives. The software is bound to the computational hardware, which is contained in the platform. Collectively these comprise the CPS that is contained in the environment.




[4] Leveson, Nancy G. *Engineering a Safer World*. To be published, Draft online: http://sunnyday.mit.edu/book2.pdf

[5] This would include general purpose processors, integrated circuits, and digital (and analog) hardware implemented neural networks.

*Figure 4-7: Cyber-Physical Perspectives*

### 4.3.4 Inter-Perspective Coupling in CPSs

In our consideration of cyber-physical systems, inter-perspective coupling is an interconnection or commonality of structure, behavior, or representation across perspective boundaries. First are phenomenological couplings between the perspectives of a CPS that relate phenomena and their measures. For example, heat energy and temperature are common to the physical perspectives of platform, environment, and computational hardware. Second are representational couplings that are transformations of abstractions across perspectives. These include the relationships between models that embody designs of a CPS, modeling approaches, and assumptions. Inter-perspective coupling is summarized in Table 4-1 and is discussed in the sections that follow.

*Table 4-1: Inter-Perspective Coupling*

| |
|---|
| **Phenomenological Coupling** |
| phenomena (shared & transformed) |
| coupling mechanisms |
| **Representational Coupling** |
| models (designs) |
| formalism |
| assumptions |

*Phenomenological couplings* are of two forms: connections of phenomena that are common in the two perspectives and transformational mappings of phenomena between perspectives.

A coupling matrix may involve the coupling of a physically equivalent phenomenon between perspectives. For example, temperature is a common phenomenon in the physical platform and physical environment and it is a common element in some of the modeling and analysis approaches for them.

A coupling matrix may involve a transformation of phenomena, reflecting phenomenological impacts across perspectives (e.g., heating that impacts the pliability of a material or deflection impacting stress in structural model). For example, there is a coupling of heat generation (and temperature) in the physical platform and the performance of a processor of the computational hardware perspective. The coupling would connect heat generation and dissipation models of the physical platform and processor performance models. In addition, there can be couplings between the heat generation/dissipation models of a physical system and power management models of a processor (i.e., slowing the clock speed to ensure a safe operating temperature or to reduce power draw).

*Representational couplings* address the relationships of models, the relationships of the formalism (form, syntax, and semantics) of the representations used to model and analyze phenomena, and the assumptions (coupling mechanisms and constraints) made in the coupling across perspectives. This is shown for the software and computational hardware perspectives in Figure 4-8.

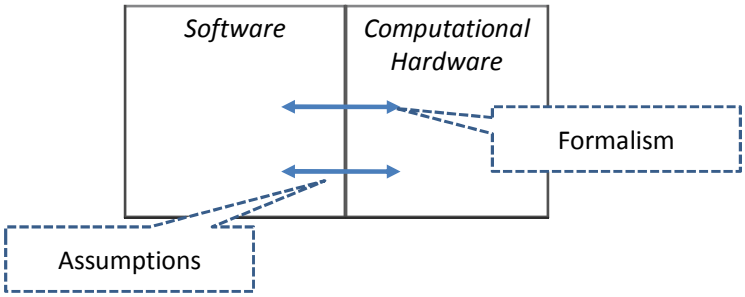



*Figure 4-8: Aspects of Representational Coupling*

The *coupling of the formalism* addresses the relationship between the modeling formalisms used in each perspective, recognizing common and idiosyncratic abstractions (e.g., discrete time representations of continuous time phenomena) that impact the coupling with that perspective. For example, in one perspective timed automata are used, whereas in another classic state machines or Petri Nets are employed. This coupling addresses how these abstractions are related.

*Coupling assumptions* address the application of formalism within and across perspectives. This includes the consistency of models and aspects of the mechanisms (e.g., sensors/actuators, accuracy, scaling, tolerances, and the like) and constraints associated with the coupling of perspectives. Models in one perspective must align purposely and semantically with those in other perspectives to ensure a consistent and complete system design. The couplings associated with models are relationships between the design-specific abstractions in each perspective, including shared (common) abstractions. For example, the dynamic mechanical models of the behavior of a UAV's aileron agree with the state machine representations in the models of the software controlling that aileron.

Assumptions can be explicitly identified. For example, pressure is used in both the physical environment and in the software perspectives for a CPS. It is documented (explicit assumption) that representations in both perspectives must use the same units (e.g., pounds per square inch) for expressing pressure. However, assumptions may be implicit in that they are understandings that developers and users have of a system that are not explicitly identified or communicated and may differ. Consider the case when the designers of the thermal control software for a satellite assume only a steady or decrease in temperature of an exposed surface unless the satellite's internal heating element is on. This ignores the potential for solar absorption heating of the surface.

As shown in Figure 4-9 we partition the four perspectives into two categories: non-computing and computing. The coupling between the two groupings is both phenomenological and representational; however, software has only representational couplings to the physical platform and environment perspectives. The coupling between software and computing hardware is more subtle. Software is a representation; however, when software is bound to and executes on computational resources, the behavior of the executing software drives the hardware manifesting physical phenomena (e.g., CPU-intensive instructions produce more heat than memory-access instructions, increased power draw due to frequent disk access). Similarly, physical phenomena impact the execution of software (e.g., external heating can impact processor performance). We consider this phenomenological coupling.

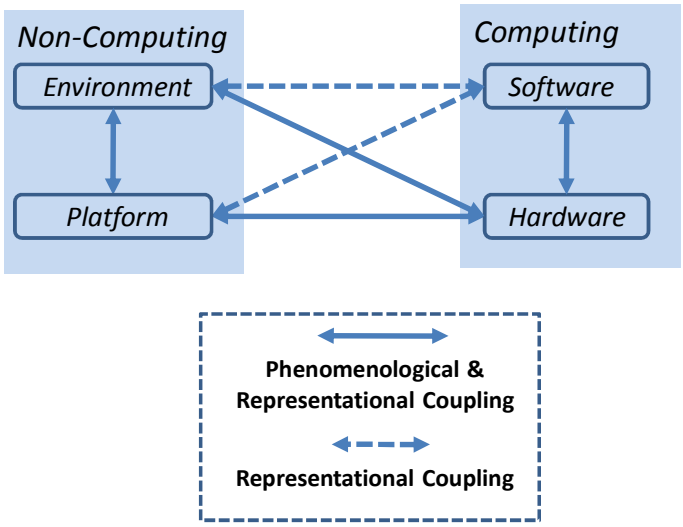



*Figure 4-9: Computing–Non-Computing Perspectives*

There are fault models and fault identification techniques available that can be used to uncover faults in a design or implementation within an individual perspective. We focus on *system-level design faults* that can arise from coupling between perspectives. These may result from a design in one perspective that adversely impacts another perspective because of coupling (i.e., the existing design, although not faulty, in one perspective when coupled with the design in another perspective will result in faulty system behavior) or design faults within the physical coupling mechanism itself. In addition, we consider the multiplicity of modeling and analysis representations within each perspective that may contribute faults, either within a perspective or impact another.

Operational faults occurring in the implementation within one perspective may propagate to another perspective due to phenomenological coupling (e.g., failure of a power source providing electrical power to computational hardware).

In defining inter-perspective coupling, we consider the categorizations shown in Table 4-1 to develop Table 4-2, Inter-Perspective Coupling Matrix, which provides more detail. The table shows the couplings that must be considered when analyzing a system across multiple perspectives. For example, the interface between Computational Hardware and the Software consists of both phenomenological and representational couplings. The binding of the software to computational resources is one form of coupling that should be considered by the developers when developing a system.

*Table 4-2: Inter-Perspective Coupling Matrix*

| System | | | |
|---|---|---|---|
| | **Software** | **Computational Hardware** | **Physical Platform** |
| **Computational Hardware** | Couplings<br>- phenomenological<br>- representational<br>Coupling mechanisms<br>- s/w binding | | |
| **Physical Platform** | Couplings<br>- representational<br>Coupling mechanisms<br>- sensor/actuator (input/output) interfaces | Couplings<br>- phenomenological<br>- representational<br>Coupling mechanisms<br>- physical mechanisms | |
| **Environment** | Couplings<br>- representational<br>Coupling mechanisms<br>- sensor/actuator (input/output) interfaces | Couplings<br>- phenomenological<br>- representational<br>Coupling mechanisms<br>- physical mechanisms<br>- transmission of shared phenomena, some mediated by physical platform | Couplings<br>- phenomenological<br>- representational<br>Coupling mechanisms<br>- physical mechanisms |

We have developed two more detailed tables for phenomenological (phenomena and coupling mechanisms) and representational (models, formalism, assumptions) coupling in CPSs. These tables provide guidance to the modeler in the form of probative questions relating to the boundaries between perspectives.

Table 4-3 includes questions to guide an engineer in identifying potential cross perspective faults relating to physical phenomena (i.e., *phenomenological coupling*). These consider shared or transferred phenomena as well as the mechanisms that enable the physical coupling between perspectives. (Note that the questions listed in the table are a subset of the extensive list generated during the work.)

*Table 4-3: Phenomenological Coupling Questions*

| **Phenomenological Coupling** | **Questions** |
|---|---|
| General<br>• Phenomena (shared/transferred)<br>• Coupling Mechanisms | What are the forms of energy transfer across boundaries?<br>What are the forms of material transfer across boundaries?<br>What is the potential (mechanisms) for common mode heating, cooling, and heat transfer across boundaries?<br>What is the potential (mechanisms) for common mode electrical interruptions, loss, minimum and peak values, or surge across boundaries? |
| Physical Platform – Environment | |
| Computational Hardware – Platform | |
| Computational Hardware - Environment | |
| Computational Hardware – Software | What are the levels and variations in percent CPU cycles, memory/disk accesses, input, output actions?<br>What are the software binding dependencies and variations? |

The coupling between software and computational hardware is both phenomenological and representational. Phenomenological coupling is such that structure (content and organization as reflected in execution) is transformed into physical phenomena such as heat and power draw variations. For example, if there is an event that flips a bit(s) in memory, software behavior is affected. Similarly, software characteristics can impact computational hardware and indirectly other perspec-

tives. For example, more disk access can result in increased heat generation, a direct impact on computational hardware, and greater power draw within the platform.

*Representational coupling* (models, formalism, and assumptions) between software and computational hardware is associated with the characteristics of the computational hardware, runtime environment, and programming languages (Table 4-4). Representational coupling into software from other perspectives is mediated, in part, through input and output interfaces (sensors and actuators). These interfaces are described as part of the coupling assumptions.

*Table 4-4: Representational Coupling Questions*

| Representational Coupling | Questions |
|---|---|
| General<br>• Models<br>• Formalism<br>• Assumptions (coupling) | Are there differences in value tolerances, units, range, or variations of quantities/variables?<br>Are there concepts that are invariants across perspectives (e.g., entropy, energy)? |
| Software – Computational Hardware | Is there consistency in the representation and metrics of the software structure?<br>What are the software binding dependencies and variations? |
| Software – Environment and Platform | Is there bias or aliasing in the sampling of sensors and sensor values? |
| Computational Hardware – Platform<br>Computational Hardware – Environment<br>Physical Platform – Environment | Have the phenomenological couplings been accounted for in the models?<br>Are there consistent abstractions in the formalism used in each perspective? |

## 4.4 Principal Activities Within the Framework

The framework consists of activities that identify hazards for a system, focusing on the interfaces between perspectives. It includes processes and methods that support development and maintenance practices for mission- and safety-critical systems as shown in Figure 4-10. They are

- identify design patterns
- identify hazards
- develop mitigation strategies

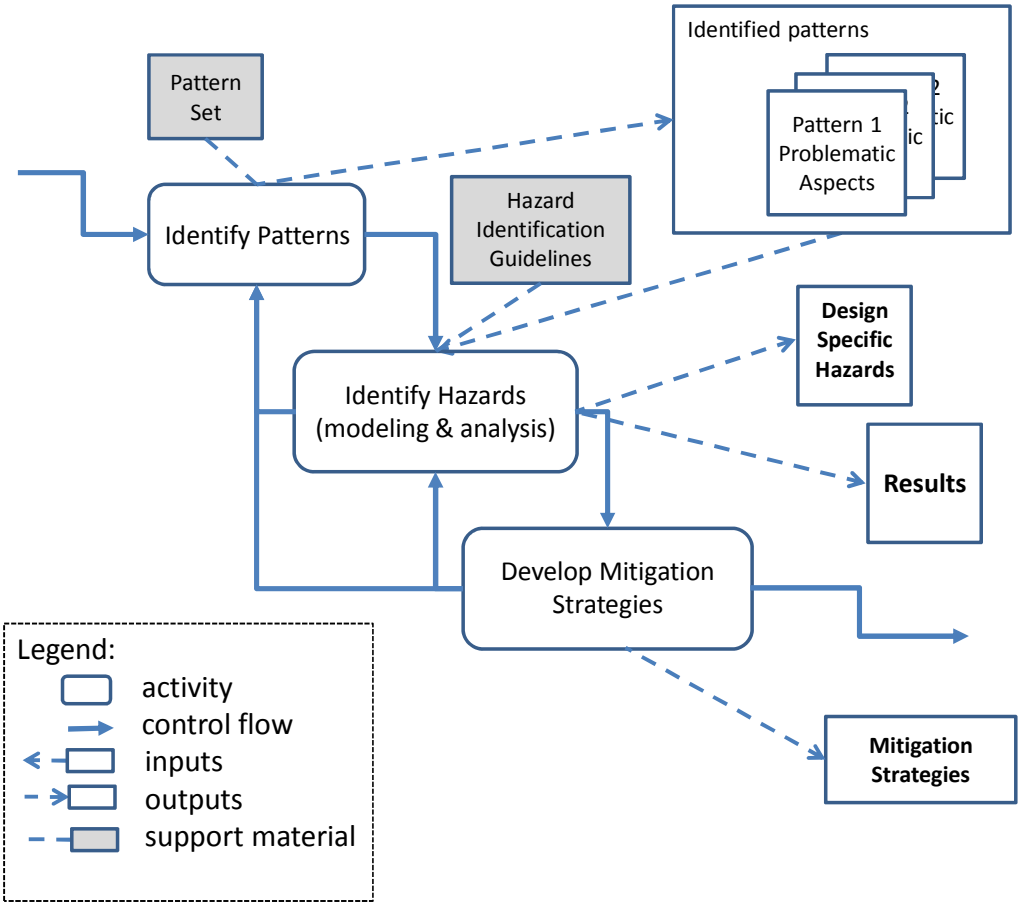



*Figure 4-10: Principal Activities of the CPS Reliability Modeling and Analysis Framework*

**Identify Design Patterns:** This activity identifies the relevant design patterns for a system from a pattern set, which is a collection of tables. We have generated a subset of tables for some patterns that lists historically problematic aspects across perspectives for the pattern. For example a publish-subscribe architecture pattern can have different operational characteristics of concern when implemented on single versus multiple computing hardware. Guidelines for relevant patterns are selected. Hazard identification guidelines define a process and supporting materials (e.g., a table of areas to consider) for uncovering hazards in a design.

**Identify Hazards:** This activity uses the pattern-based tables (guidelines) and modeling and analysis to identify (and elaborate) specific hazards present in a system. The guidelines are based upon historically problematic aspects associated with each design pattern.

We define hazards as conditions of a system and its environment that may result in an undesirable state of the system. This is more general than Leveson's definition that a hazard is "a state or a set of conditions of a system that, together with other conditions in the environment, will lead inevitably to an accident" [Leveson 1995].

This is an activity that leverages application patterns to define hazards (potential problem areas) and model, analyze, and interpret them.

**Develop Mitigation Strategies:** Mitigation strategies are techniques used to detect a fault and prevent its transmission across a boundary where it could put the target subsystem into a state that would cause a failure. Based upon the interpretation of the results of the analyses of the second activity, actions are defined to eliminate or mitigate potential consequences of hazards.

The techniques are based on the identification of the particular fault models associated with a particular engineering area. To that end, we have identified a generic set of fault types based on extensive reviews of fault models in various engineering disciplines.

## 4.5 Evaluation Criteria/Scientific Methodology/Verification

The research has been driven by a model problem and associated faults that modelers in different domains can make using invalid assumptions. We have used an autonomous casualty search and rescue system, which consists of autonomous airborne vehicles (fixed- and rotary-wing), autonomous ground vehicles (rough terrain and amphibious), medical robots, treatment centers, and support facilities. These elements work as a coordinated team to locate and transport wounded or injured military personnel to a treatment center.

### 4.5.1 Application of the Inter-Perspective Coupling Framework

The emergency vehicle example in Section 4.3.1 has illustrated a number of problems when employing different modeling approaches to develop a real-time system. For this small example, they include

1. different semantic basis of concurrent execution
2. different semantic basis of instantaneous (and synchronous) execution

Either of these mismatches would result in incorrect operation of the vehicle that would lead to catastrophic effects. The errors may surface at integration time (if the appropriate test cases and test scenarios would be applied). Given the subtle timing dependencies they likely would not surface until well into the life cycle of the vehicle. Applying the classification and checklist questions embodied in the coupling framework would allow the designer to consider the characteristics in this example.

Referring to Figure 4-9, Computing–Non-Computing Perspectives, one can intuitively map the components of our example to the four perspectives. The Environment would contain the water and temperature (forming the ice) that causes the wheel(s) to lose traction. The Platform would contain the sensors that would sense wheel rotation. The Software would contain the algorithms that implement the control laws, and the Computational Hardware would contain the execution engine that the software runs on. The phenomenological and representational coupling tables contain the questions that can be applied to each perspective pair and its associated coupling mechanism. For example, the Environment-Platform perspectives would be analyzed with respect to both the phenomenological (Table 4-3) coupling and representational (Table 4-4) coupling questions.

Considering our ESC and CC example, the control algorithms (Simulink models) would map to the Software perspective and the execution (AADL models) would map to the Computational Hardware Perspective. Both phenomenological and representational coupling mechanisms are present for this pair of perspectives. In Table 4-3, Phenomenological Coupling Questions, the question “What are the software binding dependencies and variations?” would point to the con-

current execution issue. This same question is also present in the Table 4-4, Representational Coupling Questions, but also the question "Is there a correlated approach to behavior representations (e.g., state machines)?" would serve to uncover mode semantic differences concerning instantaneous and synchronous execution between the two modeling notations.

## 4.6 Results

Significant research has been conducted and industry advances have been made in the areas of embedded real-time systems, model-based development, and verification and validation of quality attributes. The design and development of CPSs can make use of previous advances, given the similar importance and criticality of satisfying quality attribute requirements. But it is important to stress that they are intrinsically different, as we have elaborated. A more comprehensive understanding and characterization is important to help identify potential undesirable operations of CPSs.

In this work we chose to develop a framework to critically asses the interactions of components within CPS. To accomplish this, several constituent areas needed to be comprehended and correlated. This includes

- identification, definition, and categorization of the four perspectives of CPS
- identification of coupling mechanisms between the perspectives
- generation of analysis and probative questions relating to the boundaries between perspectives to uncover potential and latent faults

Our research builds on previously identified fault models in electrical, mechanical, and hydraulic systems and realized that the concept of design patterns (common in software development) is applicable in the design process of CPSs, particularly from the stand point of indentifying fault models within the pattern, and generating introspective questions regarding the modeling of, for example, computer-based control of hydraulic control systems.

In order to better understand the application of the framework, we developed a set of activities designed to identify and search for hazards within the CPS system. The activity process makes use of existing and newly discovered design patterns in each of the applicable disciplines that compose the CPS. The outcome of these activities would be to develop mitigation strategies for the identified hazards, and also develop integration test guidance to test for the existence of the hazards.

In summary, the results from this project are

- improved understanding of the semantic gaps in modeling approaches of CPS
- introduction of a categorization and classification of CPS components to aid in reasoning about coupling of system artifacts
- development of a framework for helping to identify design and modeling problems in CPSs

### Further Work and Related Areas

This work considered a broad area of the CPS world and touched on many disciplines and current research efforts that could potentially impact the analysis framework that we have outlined. While

the research area is vast and many concerns call for attention, we consider the following research problems most promising for future research:

- One area is the *further identification and use of patterns to better understand hazards and the general and specific mitigation strategies* between platform and software perspectives; in particular, feedback loops exemplified by the four-variable model where quantities in the environment such as speed, pressure, number of operators are expressed as numerical values with or without units [Parnas 1991].
- As we developed the notion of representational coupling across perspectives we made *specific questions regarding the consistency of units, valid range, representation resolution, and the like, both intra- and extra-perspective*. This issue points to a larger area of investigation generally referred to as assumption management. Several themes for further work in this area include semantics and consistency checking within and across modeling perspectives, automated consistency-checking techniques across formalisms (models), and semantics of coupling expressions.
- Somewhat related to assumption management is to *understand constraints as they apply to all perspectives*. Constraint languages have been used in some modeling environments to express ranges and limits of physical entities, computational resources, and so forth. Expression of and consistency checking of constraints for variables and state across models is a research area that would have a large impact on the minimization of a large class of defects when modeling in different formalisms.

## 4.7 Collaborations

The SEI team consisted of Peter Feiler, David Gluch, Jörgen Hansson, John Hudak, Dionisio de Niz, Andres Pace-Dias, Lutz Wrage, and Chuck Weinstock. The U.S. Army Aviation and Missile Research Development and Engineering Center (AMRDEC) was a partner in identifying problems that typically cause unforeseen consequences in avionic systems during integration testing and into deployment.

## 4.8 Publications and Presentations

### 4.8.1 Bibliography

SAE. *Architecture Analysis and Design Language (AADL) SAE Aerospace Standard AS5506*, Issued 2004-11.

Abdelzaher, Tarek; Gill, Christopher D.; Rajkumar, Raj; & Stankovic, John A. "Distributed Real-Time and Embedded Systems Research in the Context of GENI." *National Science Foundation Workshop on Distributed Real-Time and Embedded Systems Research in the Context of GENI*. October 2005.
http://groups.geni.net/geni/attachment/wiki/OldGPGDesignDocuments/GDD-06-32.pdf

Akella, Ravi & McMillin, Bruce M. "Model-Checking BNDC Properties in Cyber-Physical Systems," 660-663. *Proceedings of 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2009)*. July 2009, Seattle, Washington. IEEE Computer Society, 2009.

Atkins, Ella M. "Cyber-Physical Aerospace: Challenges and Future Directions in Transportation and Exploration Systems." *Proceedings of the 2006 National Science Foundation Workshop On Cyber-Physical Systems.* October 2006, Austin, TX.

Black, Jennifer & Koopman, Philip. "System Safety as an Emergent Property in Composite Systems," 369-378. *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks 2009 (DSN'09).* June 2009, Estoril, Portugal. IEEE Computer Society, 2009.

Bujorianu, Marius C.; Bujorianu, Manuela L.; & Barringer, Howard. "A Unifying Specification Logic for Cyber-Physical Systems," 1166-1171. *Proceedings of 17th Mediterranean Conference on Control & Automation (MED '09).* June 2009, Thessaloniki, Greece. Institute of Electrical and Electronics Engineers, 2009.

Campbell, Roy H.; Garnett, Guy; & McGrath, Robert E. "CPS Environments." *Proceedings of the 2006 National Science Foundation Workshop On Cyber-Physical Systems.* October 2006, Austin, TX.

Cheng, Betty H. C. & Atlee, Joanne M. "Research Directions in Requirements Engineering," 285-303. *Proceedings of the Future of Software Engineering (FOSE'07).* May 2007, Minneapolis, MN.

Cook, Jeffrey A. "Position Paper: Introduction: Cyber-Physical Systems and The Twenty-First Century Automobile." *Proceedings of the 2006 National Science Foundation Workshop On Cyber-Physical Systems.* October 2006, Austin, TX.

Dominguez, Alma L. Juarez. "Feature Interaction Detection in the Automotive Domain," 521-524. *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering(ASE 2008).* September 2008, L'Aquila, Italy. IEEE Computer Society, 2008.

Feron, Eric, et al. *Report: Cyber-Physical Systems Summit*, April 2008. St. Louis, MO.

Fuhrman, Tom. "Position Paper: Introduction: The Automotive Context." *Proceedings of the 2006 National Science Foundation Workshop On Cyber-Physical Systems.* October 2006, Austin, TX.

Gill, Christopher. "Cyber-Physical System Software for HCMDSS," 176-177. *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability.* June 2007, Boston, MA. Institute of Electrical and Electronics Engineers, 2008.

Lin, Jing; Sedigh, Sahra; & Miller, Ann. "A General Framework for Quantitative Modeling of Dependability in Cyber-Physical Systems: A Proposal for Doctoral Research," 668-671. *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*. July 2009, Seattle, WA. IEEE Computer Society, 2009.

McMillin, B.; Gill, C.; Crow, M. L.; Liu, F.; Niehaus, D.; Potthast, A.; & Tauritz, D. "Cyber-Physical Systems Engineering: The Advanced Power Grid." *Proceedings of the 2006 National Science Foundation Workshop On Cyber-Physical Systems.* October 2006, Austin, TX.

Passerone, Roberto; Hafaiedh, Imene Ben; Graf, Susanne; Benveniste, Albert; Cancila, Daniela; Cuccuru, Arnaud; Gérard, Sébastien; Terrier, Francois; Damm, Werner; Ferrari, Alberto; Mange-

ruca, Leonardo; Josko, Bernhard; Peikenkamp, Thomas; & Sangiovanni-Vincentelli, Alberto. "Metamodels in Europe: Languages, Tools, and Applications." *IEEE Design & Test of Computers 26,* 3 (May/June 2008): 38-53. IEEE Computer Society, 2009.

Porter, Joseph; Volgyesi, Peter; Kottenstette, Nicholas; Nine, Harmon; Karsai, Gabor; & Sztipanovits, Janos. "An Experimental Model-Based Rapid Prototyping Environment for High-Confidence Embedded Software," 3-10. *Proceedings of the Twentieth IEEE International Symposium on Rapid System Prototyping (RSP'09).* Paris, France, June 2009. IEEE Computer Society 2009.

Paul, Raymond; Yen, I-Ling; Bastani, Farokh; Dong, Jing; Tsai, Wei-Tek; Kavi, Krishna; Ghafoor, Arif; & Srivastava, Jaideep. "An Ontology-Based Integrated Assessment Framework for High-Assurance Systems," 386-393. *Proceedings of the IEEE Conference on Semantic Computing (ICSC 2008)*, Santa Clara, CA, August 2008. IEEE Computer Society, 2008.

Sha, Lui; Gopalakrishnan, Sathish; Liu, Xue; & Wang, Qixin. "Cyber-Physical Systems: A New Frontier," 1-9. *Proceedings of 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008*), Taichung, Taiwan, June 2008. IEEE Computer Society 2008.

Simulink Modeling Tool. Developed by Mathworks, Natick, MA. http://www.mathworks.com/

Stankovic, John A.; Lee, Insup; Mok, Aloysius; & Rajkumar, Raj. "Opportunities and Obligations for Physical Computing Systems." *Computer, 38*, 11 (November 2005): 23-31.

Sun, Yan; McMillin, Bruce; Liu, Xiaoqing (Frank); & Cape, David. "Verifying Noninterference in a Cyber-Physical System—The Advanced Electric Power Grid," 363-369. *Proceedings of Seventh International Conference on Quality Software (QSIC '07)*. Portland, OR, October 2007. IEEE Computer Society, 2007.

Sztipanovits, Janos. "Composition of Cyber-Physical Systems," 3-6. *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07).* Tucson, AZ, March 2007. IEEE Computer Society, 2007.

Talbot, Steve C. & Ren, Shangping. "Comparison of FieldBus Systems, CAN, TTCAN, FlexRay and LIN in Passenger Vehicles," 26-31. *Proceedings of the 29th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2009 Workshops).* Montreal, Québec, Canada, June 2009. IEEE Computer Society, 2009.

Tan, Ying; Vuran, Mehmet C.; & Goddard, Steve. "Spatio-Temporal Event Model for Cyber-Physical Systems," 44-50. *Proceedings of the 29th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2009 Workshops).* Montreal, Québec, Canada, June 2009. IEEE Computer Society, 2009.

Tang, Qinghui' Gupta, Sandeep Kumar S.; & Varsamopoulos, Georgios. "Energy-Efficient Thermal-Aware Task Scheduling for Homogeneous High-Performance Computing Data Centers: A Cyber-Physical Approach." *IEEE Transactions on Parallel and Distributed Systems, 19,* 11 (November 2008): 1458-1472.

Tirumala, Ajay Sudarshan. “An Assumptions Management Framework For Systems Software.” PhD diss., University of Illinois at Urbana-Champaign, 2006.

Valls, Marisol García; Val, Basanta Pablo; & Ayres, Iria Estévez. “Concurrency Programming Models in Mobile Real-Time Platforms,” 538-543. *Proceedings of 2009 International Conference on Advanced Information Networking and Applications Workshops (WAINA 2009).* Bradford, United Kingdom, May 2009. IEEE Computer Society, 2009.

Verissimo, Paulo; Alysson Neves, Bessani; Miguel, Correia; Neves, Nuno Ferreira; & Sousa, Paulo. “Designing Modular and Redundant Cyber Architectures for Process Control: Lessons Learned,” 1-8. *Proceedings of the 42nd Hawaii International Conference on System Sciences.* Waikoloa, HI, January 2009. IEEE Computer Society, 2009.

West, Richard & Parmer, Gabriel. “A Software Architecture for Next-Generation Cyber Physical Systems.” *Proceedings of the 2006 National Science Foundation Workshop on Cyber-Physical Systems.* Austin, TX, October 2006.

Wolf, Wayne. “Cyber-Physical Systems.” *Computer 42*, 3 (March 2009): 88-89.

Woo, Honguk; Yi, Jianliang; Browne, James C.; Mok, Aloysius K.; Atkins, Ella; & Xie, Fei. “Design and Development Methodology for Resilient Cyber-Physical Systems,” 525-528. *Proceedings of the 28th International Conference on Distributed Computing Systems Workshops.* Beijing, China, June 2008. IEEE Computer Society, 2008.

Xie, Le & Ilić, Marija D. “Module-Based Modeling of Cyber-Physical Power Systems,” 513-518. *Proceedings of the 28th International Conference on Distributed Computing Systems Workshops.* Beijing, China, June 2008. IEEE Computer Society, 2008.

Zhang, Fumin; Szwaykowska, Klementyna; Wolf, Wayne; & Mooney, Vincent. “Task Scheduling for Control Oriented Requirements for Cyber-Physical Systems,” 47-56. *Proceedings of the 2008 Real-Time Systems Symposium.* Barcelona, Spain, December 2008. IEEE Computer Society, 2009.

### 4.8.2 References

**[Kinjawadekar 2009]**
Kinjawadekar, Tejas Shrikant. “Model-Based Design of Electronic Stability Control System for Passenger Cars Using CarSim and Matlab-Simulink.” Master thesis, Ohio State University, 2009.

**[Lee 2008]**
Lee, Edward A. “Cyber Physical Systems: Design Challenges,” 363-369. *Proceedings of 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC).* Orlando, Florida, May 2008. IEEE Computer Society, 2008.

**[Leveson 1995]**
Leveson, Nancy G. *Safeware: System Safety And Computers*. Addison-Wesley, 1995 (ISBN 0-201-11972-2).

**[Parnas 1991]**
Parnas, D. L. & Madey, J. *Functional Documentation for Computer Systems Engineering, Vol. 2* (Technical Report CRL 237). McMaster University, 1991.

# 5 Achieving Predictable Performance in Multi-Core Embedded Real-Time Systems—2nd Year

Dionisio de Niz, Karthik Lakshmanan, Gabriel Moreno, Ragunathan (Raj) Rajkumar, Jeffrey Hansen, Christopher Craig, and Onur Mutlu

## 5.1 Purpose

The community at large has already recognized the risk involved in the trend to base the speedup of processors on an increasing number of cores instead of faster single cores. To harvest such a speedup enough executable elements (tasks, processes, threads, instructions) must be available to be executed in parallel in each of the cores. New allocation, scheduling, and synchronization techniques are needed in order to utilize the capacity of the cores. There are two aspects to this effort: (i) ensuring that the system maintains its predictable timing behavior when executing on a multi-core, and (ii) maximizing the parallelism in order to minimize idleness of some of the cores. It is paramount that we get predictable timing of such execution to be able to guarantee deadlines. For any power-constrained or energy-conserving embedded system, it is also essential that no core is idly consuming energy. On top of this, the consequence of wasting processor performance can have a higher penalty in embedded systems where the cost is an important driving force. For example, this is the case in automotive systems, where economy of scale magnifies the cost of hardware underutilization. Missing the performance improvements of multi-core processors in embedded systems can have huge economic consequences. Current multi-core chips contain typically four to eight homogeneous cores. Within three to five years, multi-core chips will exceed 32 cores, and also be heterogeneous (i.e., they will be able to run at different speeds, and possibly be tailored for specific functions).

During the first year of this project we focused our work on three areas of the scheduling of multi-core processors. First we explored the tradeoffs of global versus local scheduling of multi-core processors with the real-time queuing theory analytical framework. Second, we developed new coordinated allocation, scheduling and synchronization mechanisms to minimize the timing delays introduced by tasks synchronizing across cores. Finally, we developed new scheduling algorithms for tasks sets with mixed-criticality levels where the temporal protection has no side effects to critical systems. At the same time we have explored the application of the technology developed during the first year to a model problem defined with our collaborators from Lockheed Martin.

In the second year of this project we extended our investigation on the protection mechanisms of mixed-criticality systems in multi-core architectures in two fronts:

- on the input/output (I/O) subsystem, where I/O requests from one core can interfere with requests from other cores
- on the hardware shared across cores (e.g., caches, memory, and core interconnects)

Finally, we will add a new component to this project to investigate the conflicts between the scheduling decision of the hardware schedulers that manage the hardware shared across cores and the decisions of the operating system (OS) schedulers, and propose solutions to such conflicts.

The importance of the new part of the project on the hardware/OS schedulers derives from the fact that multi-core processors have shifted the way we think about performance improvement from minimizing the time to execute a single instruction to increasing the number of instructions we can execute in parallel. In an ideal multi-core processor, all cores in the processor can execute instructions in parallel with each other. In practice, these cores share common hardware, such as memory bus, cache, and interconnect links that can only be used by one core at a time and need to be scheduled. Such sharing can stall the execution of an instruction that requires a hardware resource (e.g., memory bus) in one core because an instruction in another core is using the same resource. As more cores are added to a single chip and the number of cores sharing a single hardware resource increases, the growth rate of parallelism and hence overall system performance decreases. In the extreme, if every instruction has to use a shared resource, then all of the cores (hence instructions) will take turns and the execution becomes a single sequence, reducing the performance of the multi-core system to an equivalent single core. In fact, due to the destruction of locality and parallelism in the shared resources for a single core, in the extreme, performance of the multi-core system can be less than that of the single-core system.

The objective of the hardware schedulers is to minimize the performance bottlenecks when scheduling shared hardware. Unfortunately, for real-time systems, hardware schedulers typically optimize for throughput, disregarding any quality of service (QoS) requirements from the different cores. Furthermore, the scheduling decisions of the OS scheduler (what task to run next) can be in direct contradiction to the hardware scheduler decisions. For instance, while the OS scheduler may decide to stop a task to give the processor to one with a higher priority, the hardware scheduler may decide to service a memory transfer from another core with a task with lower priority, stalling the newly activated high-priority task. As the number of shared resources and cores increase in modern processors, the consequences of the lack of coordination between the OS and hardware schedulers can destroy any possibility of predictable performance for real-time systems.

## 5.2 Background

The shift of the processor evolution toward multi-core architecture has changed both the way we think about resource management and schedulers and the way we think about applications. For real-time systems this means that we need to revisit the assumptions about the temporal properties that they need to provide to the application. At the same time the requirements of the real-time applications had been changing. Nowadays their timing characteristics (e.g., worst-case execution time) are less predictable given the extensive use of artificial intelligence algorithms that do not have a fixed computation budget. Their requirements have also changed, since they need to deal with more uncertainty in the environment (e.g., airborne objects that come and go) and it is possible to tolerate longer deadlines and miss some of them. While at first sight these new characteristics look like an extra burden on top of the shift in processor architecture, we have discovered synergies that can be exploited. In particular, during our first year we discovered that the variability in the execution time is better supported in multi-core architectures than in single core ones. In addition, we discovered that for mixed-criticality systems, multi-core architectures enable new criticality mixtures that increase the utilization of the whole system. As a result, in this second year of our project we explored the new performance models for modern real-time applications as well as the new resource scheduling challenges coming from a combination of new hardware sharing schemes and new hardware schedulers embedded in the multi-core architectures.

One important trend for our investigation is the increasing pressure to reduce cost and physical resources such as power and heat. This has resulted in a corresponding trend to consolidate more and more functionality into a smaller number of processors. Unfortunately, this consolidation involves resource sharing (e.g., processor, memory, I/O, and network bandwidth sharing) that can lead to interference across tasks from different functionality (e.g., one task using the processor longer than expected). This problem escalates when tasks from the consolidated functionality have different levels of criticality because a lower criticality task can interfere with a higher criticality one. For instance, suppose we deploy a task from the anti-lock braking system on the same processor as one from the navigation system of the car. If the navigation task does not release the processor on time it could make the braking task miss its deadline. This is a mixed-criticality problem that is increasingly prevalent in today's cyber-physical systems and a growing concern. In particular, the U.S. Air Force Research Laboratory has been leading the Mixed-Criticality Architecture Requirements (MCAR) initiative to investigate building blocks to safely construct these mixed-criticality systems. At a higher level, the question is whether safety-critical and mission-critical functions can be co-located on the same processor(s).

Resource partitioning is the key mechanism used to prevent interferences due to shared resources. This partitioning has taken two main forms: those based on rate-monotonic analysis (RMA), and those based on time-division multiplexing (TDM). The RMA-based schemes are typically implemented as resource reserves [Oikawa 1998] and have been applied to processor, disk, network, and memory. The TDM-based schemes include TTA [Kopetz 1998] and FlexRay [Mores 2001] for network, and the ARINC 653 standard [ARINC 2005] for processor. Resource partitioning incurs its own costs. In particular, we need to provision for the worst-case resource demand (e.g., worst-case execution time—WCET) even though such a demand is only exercised in rare occasions.

Obtaining this WCET is a challenging task due to hardware unpredictability (e.g., caches, pipelines, speculative execution) and the dependency of the execution time of some algorithms on the environment (e.g., number of obstacles to avoid). As a result, WCET numbers are often obtained through measurements and supplemented by an added cushion factor as an educated and typically conservative estimate. This leads to a twofold problem: poor average utilization of the processor and occasional enforcement (temporary stopping) of tasks that need to run longer than their WCET. This enforcement ensures that the tasks attempting to execute longer than specified do not steal cycles from others. This property is known as temporal isolation.

Such enforcement prevents low-criticality tasks from interfering with higher criticality ones and can make the low-criticality task miss its deadline while preventing the higher criticality task from missing its deadline. Unfortunately, such enforcement also affects the higher criticality task in order to prevent its interference on a low-criticality task. As a result, the enforcement can make the higher criticality task miss its deadline to allow the low-criticality one to meet its own. We call this type of enforcement symmetric enforcement because it acts the same way in both directions: low-to-high criticality and high-to-low criticality. This enforcement can have the opposite effect to our original intent: a lower criticality task is favored over a higher criticality tasks and criticality inversion is said to occur.

In the area of hardware schedulers on multi-core architectures, we have observed that these schedulers are mainly application-agnostic. This can cause important performance bottlenecks that can

compromise the predictability of real-time systems. For instance, in Mutlu and Moscibroda the authors describe a denial-of-service (DoS) situation when two cores access shared memory [Mutlu 2007]. This denial of service happens when the memory system services first the request (schedule the request) whose address is closer to previously served ones (in the same memory row). This is because it takes less time to service the request (read bytes) from the same row than changing to a different row. The result of this unfair preference by the memory system is that the number of bytes read per second (overall memory throughput) increases, but the other requests are delayed and their throughput is reduced significantly. Specifically, in Mutlu and Moscibroda the authors performed experiments where this reduction reached up to 2.9 times in real dual-core processors. In simulations of 16-core processors, performance losses of up to 22 times were observed. In all these cases, it was shown that applications with characteristics that are not favored by the memory scheduler were starved for long time periods, making their performance unpredictable and low.

The significance of the performance degradation that shared hardware has on multi-core processors has also been highlighted by researchers at the Sandia National Laboratories. In particular, a simulation presented by Moore shows that with the current trends of hardware sharing, the performance of a 16-core computer will be the same as one with a single core due to memory stalls [Moore 2008]. Furthermore, as the number grows to 64 cores, the performance is downgraded beyond an order of magnitude (see Figure 5-1).

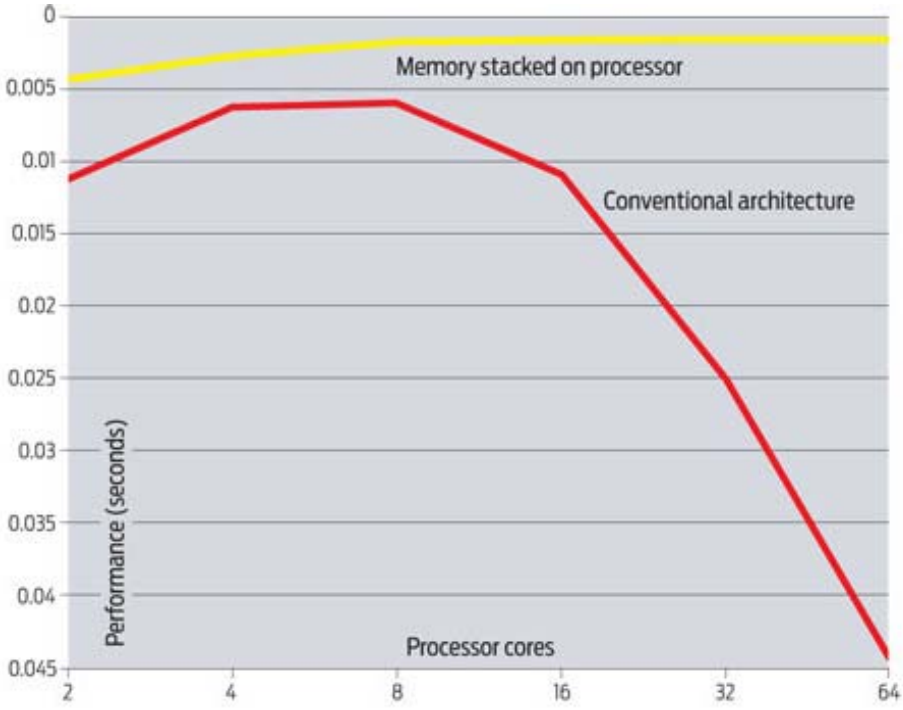



*Figure 5-1: Performance Degradation in Multi-Core Processors*

In the context of real-time systems, the conflict created by the hardware scheduling of shared resources creates unpredictable results and significant utilization degradation. For instance, in Pellizzoni, the authors discuss the problems of the delays created by the shared access to the memory bus from the processor and direct memory access (DMA) devices [Pellizzoni 2008]. In this paper,

Pellizzoni and associates show that slowdowns of tasks of up to 44 percent had been observed in practice due to this effect.

With the arrival of heterogeneous multi-cores to the multi-core mix, the scheduling decisions and its consequences become even more significant. As an example, consider a typical smartphone. Today's smartphones typically have two separate processors, one to process the sound of the conversation in real time and another to execute general-purpose programs such as calendar, browser, email, and the like. As many cores become available, these two processors can be merged into a heterogeneous multi-core with one of the cores specialized for sound processing (a digital signal processor—DSP) and another for general-purpose computing, both sharing memory. In such a case, the decision on how to handle a memory request can severely affect the functionality of the system. In particular, if we make the DSP wait too much for memory requests, then the sound of the conversation can break due to the late processing of sound packets. On the other hand, the general-purpose application may be able to wait longer without the user noticing the difference. As a result, the knowledge about the different requests, and the specific timing tolerance to the requests, are things that must be taken into account to properly schedule shared resources in many cores.

## 5.3 Approach

Our approach to address the issues of the current software development theory and practice for embedded real-time systems when using multi-core platforms is as follows.

- Characterize the stochastic nature of real-time systems in multi-core processors and the possibilities to relax deadline guarantees.
- Develop extensions to the real-time queuing theory to match the characteristics of the workload of the real-time systems running in multi-core processors providing probabilistic deadline guarantees.
- Study the new criticality mixtures that are possible to get in multi-core architectures. This includes the study of I/O scheduling and shared hardware across cores and how to combine them into a coherent framework to satisfy mixed-criticality workloads.
- Study the conflicts between hardware and software schedulers in multi-core processors. Create new coordination protocols and compatible hardware and software schedulers. We would prototype the hardware schedulers in field-programmable gate arrays.
- Take advantage of our collaboration with Bosch and foster the continuation of our collaboration with Lockheed Martin to obtain additional information and configure new model problems for the areas of study of the IRAD.

## 5.4 Collaborators

From the Carnegie Mellon University campus we collaborated with two professors: Prof. Raj Rajkumar and Prof. Onur Mutlu. Rajkumar, of Carnegie Mellon's Department of Electrical and Computer Engineering, is co-director of the GM Collaborative Research Laboratory and co-director of the GM Autonomous Driving Laboratory. We will fund one month of his time and one doctoral candidate for the duration of the project.

Mutlu, also of Carnegie Mellon's department of electrical and computer engineering, is an expert in computer architecture and multi-core architectures in particular. We will fund one month of his time and one doctoral candidate for the duration of the project.

From industry we collaborated with Lockheed Martin Corp. and with Robert Bosch. Our Lockheed Martin collaborators were Ben Watson, Russell Kegley, and Gautam Thaker, funded by their own organization.

## 5.5 Evaluation

The evaluation was based on the following deliverables:

- a paper on a new task allocation algorithm called *Compress-on-Overload Packing* for multiprocessor scheduling for real-time mixed criticality systems at the 2010 International Conference on Distributed Computing Systems
- submission of a paper on synchronization of real-time mixed-criticality systems to the 2011 Real-Time and Embedded Technology and Applications Symposium
- a graphics processing unit resource allocation experimental platform for games
- a multiprocessor radar demo for real-time mixed-criticality systems
- a new link scheduling for inter-core networks to ensure real-time requirements are properly isolated

## 5.6 Results

### 5.6.1 Multiprocessor Scheduling for Real-Time Mixed-Criticality Systems

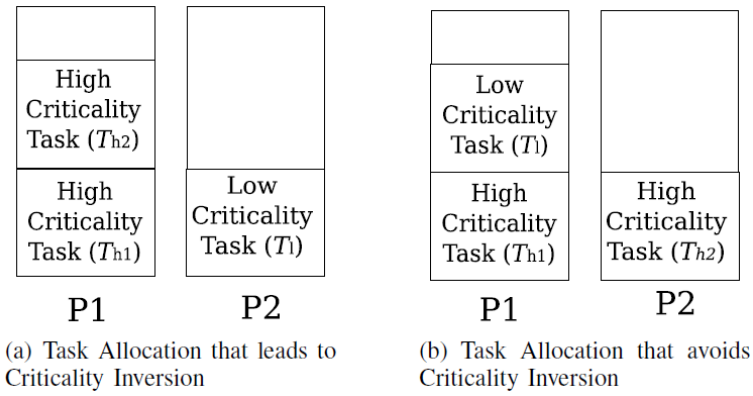



*Figure 5-2: Criticality Inversion in Allocation*

Partitioned scheduling in distributed systems with a single criticality level is decomposed into (1) task allocation (or assigning tasks to processors) and (2) uni-processor task scheduling. When multiple levels of criticality are involved, the system is required to ensure that under overload scenarios the most critical tasks are guaranteed to meet their deadlines. This is known as *graceful degradation*. Providing graceful degradation presents an important resource allocation problem in distributed mixed-criticality systems, which needs to be addressed at both the task allocation and uni-processor scheduling phases.

When mixed-criticality tasks are scheduled with traditional real-time scheduling algorithms—and the system gets into an overload—providing graceful degradation can be inefficient. Specifically, in mixed-criticality systems there is a need to ensure that the tasks do not execute longer than their specified worst-case execution time. This is to prevent low-criticality tasks from interfering with higher criticality ones. While this is, in general, considered a fault, an explicit protection against it is needed because it can create (temporal) overload situations. Since the traditional priority assignment made by the scheduler was tailored to increase schedulable utilization, it is agnostic to criticality. In particular, under priority-based preemptive scheduling, a low criticality task can have a higher scheduling priority than a higher criticality task. Such priority assignment would schedule the low criticality task earlier than the higher criticality task, potentially making the latter miss its deadline (i.e., a criticality inversion). On the other hand, if we assign priorities based on criticality, then we eliminate criticality inversion. However, this assignment can potentially create significant priority inversion from the perspective of priority-based preemptive schedulers. Last year we developed zero-slack scheduling for uni-processor real-time systems (zero-slack rate monotonic, or ZSRM) to ensure that lower criticality tasks do not interfere with higher criticality tasks but that the latter can steal cycles from the former in case of overload, ensuring graceful degradation [de Niz 2009].

In a distributed setting, the criticality inversion problem can arise at the task-allocation level, since a given allocation could favor a low-criticality task at the expense of a high-criticality one. For instance, consider three tasks $\tau_{h1}$, $\tau_{h2}$, and $\tau_l$ of high, high, and low criticality respectively, each having a normal utilization $\frac{C_i}{T_i}$ of 40 percent (shown in Table 5-1). Say we only have two processors, $P_1$ and $P_2$, and $\tau_{h1}$ is already deployed on $P_1$, and the three tasks do not fit on $P_1$ together. We are then forced to pack either $\tau_{h2}$ or $\tau_l$ on $P_1$ and the other to $P_2$. Packing $\tau_{h1}$ and $\tau_{h2}$ together, and $\tau_l$ by itself is a possible task allocation decision (see Figure 5-2a). In fact such an allocation decision is commonly used in legacy systems that try to isolate criticality levels.

However, observe that such task allocation leads to a criticality inversion problem. In this scenario, if all the tasks overload, $\tau_{h2}$ may miss its deadline but $\tau_l$ will not (i.e., our allocation decision protected $\tau_l$—a low criticality task—at the expense of $\tau_{h2}$, a high criticality task). Conversely, deploying $\tau_{h1}$ and $\tau_l$ together and $\tau_{h2}$ by itself removes this criticality inversion (see Figure 5-2b). Note that using a criticality-aware uni-processor scheduling algorithm such as ZSRM will ensure that $\tau_l$ cannot steal cycles from $\tau_{h1}$ within processor $P_1$ under overload scenarios. As illustrated by the above example, allocating tasks to processors can introduce criticality inversion, which affects the system performance under overloads. A criticality-aware task allocation algorithm can mitigate this problem by enabling more critical tasks to meet their deadlines under overload scenarios.

### 5.6.2 Compress-On-Overload Packer

The task allocation problem in distributed mixed-criticality systems has a dual objective:

1. We need to minimize the deadline misses of high-criticality tasks under all possible overload cases.
2. We also need to minimize the number of processors needed by the system.

In order to achieve this, we design a two-phase algorithm. In the first phase, we allocate the tasks, ensuring that all of them are schedulable even if they run their corresponding overload budget

($C^o$). For this packing we explore three variants of bin packers based on: best-fit decreasing (BFD), first-fit decreasing (FFD), and worst-fit decreasing (WFD). These packers try to fit tasks ordered first by criticality, and then in decreasing order of overload utilization ($\frac{C^o}{T}$). Each task considers each available processor in an order based on the overloaded fullness level ($\sum_i \frac{C_i^o}{T_i}$) as defined by the different variants: decreasing for BFD, increasing for WFD, and without any order for FFD. Any task that is unschedulable is kept aside. This first phase thus completely eliminates the consequence of overloading and ensures that no allocated task can miss its deadline on overload. At the end of this phase, we will have a set of tasks that did not fit. These tasks are now packed using a modified WFD packing. Under this packing, tasks are ordered first by criticality, and then in the decreasing order of normal utilization ($\frac{C}{T}$). Each task considers each available processor in the increasing order of normal fullness level. For this second phase, we let ZSRM compress the schedule by allowing the higher criticality tasks to suspend lower criticality tasks (stealing their cycles) if they get into an overload. Such a compression reduces the number of processors needed to schedule the tasks under ZSRM at the expense of deadline misses incurred by the lower criticality tasks under these overloads. At the end of this second phase, some tasks might still be unschedulable and left outside of the packed set.

Mixed-criticality systems comprise applications with different criticality levels and timing constraints co-located on a distributed set of processors. Such co-location is largely driven by cost and physical space considerations. The desirable property in mixed-criticality systems is that in the face of overload scenarios the most critical applications continue to meet their deadlines. In order to capture this property formally, we introduce a matrix called the ductility matrix to fully describe the potential behaviors of the system with respect to two factors: (1) the level of overload faced by tasks, and (2) tasks that miss their deadlines due to a given overload (see Lakshmanan for a full description) [Lakshmanan 2010].

We evaluated our packing algorithms (COP (FFD), COP (BFD), and COP (WFD)) with respect to ductility and were able to achieve up to five times more ductility than traditional packing algorithms, as seen in Figure 5-3.

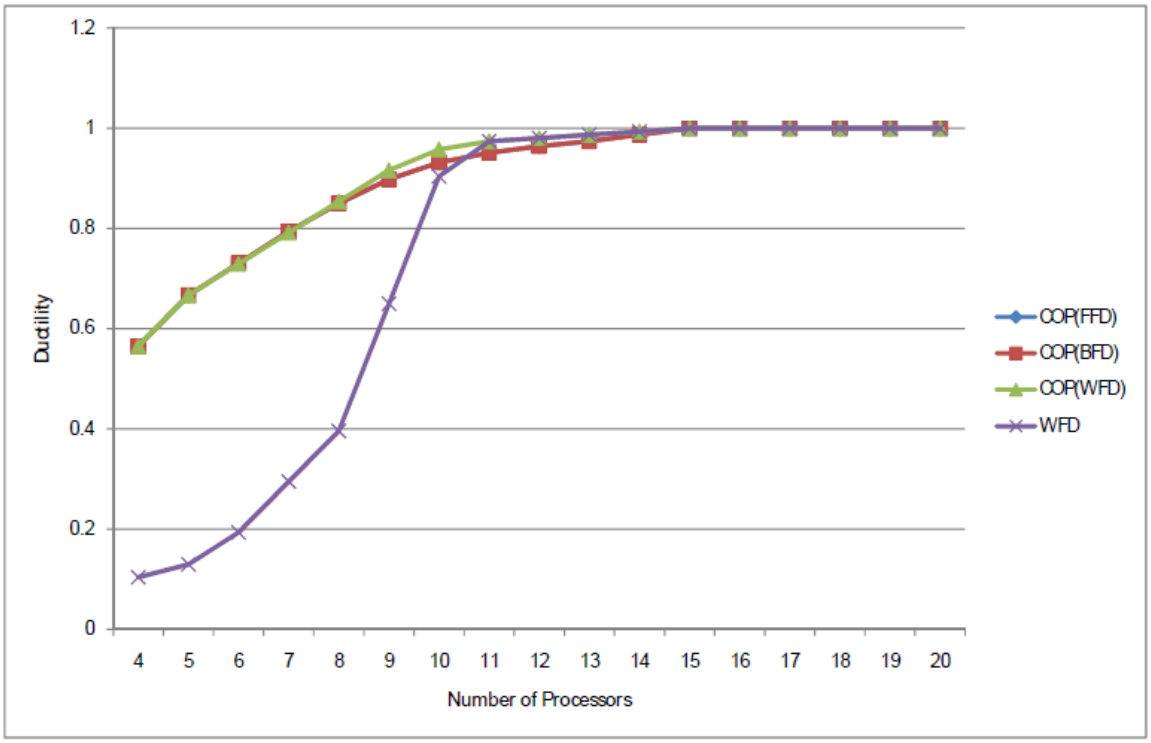



*Figure 5-3: Ductility Evaluation*

We extended the radar demo developed in our previous year to use our COP allocation algorithm in a multi-processor platform. The tasks' parameters are shown in Table 5-1 and the deadline misses observed in the experiment in Table 5-2. It is worth noting that in Table 5-2 COP only show deadlines from HP Friendly tasks that are less critical than the NP Hostile tasks that miss the deadline when allocated with WFD.

*Table 5-1: Radar Task Set*

| $Task$ | $C_i$ (ms) | $C_i^o$ (ms) | $Period$ $T_i$ (ms) | $Criticality$ $\kappa(\tau_i)$ |
|---|---|---|---|---|
| HP Hostile | 40 | 58 | 100 | 1 |
| NP Hostile | 83 | 106 | 200 | 1 |
| HP Friendly | 40 | 58 | 100 | 2 |
| NP Friendly | 83 | 106 | 200 | 2 |

*Table 5-2: Deadline Misses in Overload*

| Packer | Deadline misses (% misses { Task }) | | |
|---|---|---|---|
| | 2300 Tracks | 2400 Tracks | 2500 Tracks |
| WFD | 0 | 24.32 {NP Hostile} | 69.56 {NP Hostile} |
| COP | 0 | 10.34 {HP Friendly} | 59.18 {HP Friendly} |

### 5.6.3 Criticality Inheritance in Mixed-Criticality Scheduling

Zero-slack scheduling algorithms are designed to cope with overload conditions. When these conditions develop, critical tasks must still meet their timing constraints at the expense of less critical tasks. Zero-slack scheduling algorithms guarantee that all tasks meet their deadlines when no overload occurs, and that criticality ordering is satisfied under overloads. Unfortunately, when mutually exclusive resources are shared across tasks, these guarantees are voided. Furthermore, the dual-execution modes of tasks in mixed-criticality systems violate the assumptions of traditional real-time synchronization protocols such as PCP, and hence the latter cannot be used directly.

Hence, we developed extensions to real-time synchronization protocols that coordinate the mode changes of the zero-slack scheduler [Sha 1990]. We analyze the properties of these new protocols and the blocking terms they introduce. We maintain the deadlock avoidance property of our PCP extension, called the priority and criticality ceiling protocol (PCCP), and limit the blocking to only one critical section for each of the zero-slack scheduling execution modes. We also develop techniques to accommodate the blocking terms arising from synchronization, in calculating the zero-slack instants used by the scheduler. Finally, we conduct an experimental evaluation of PCCP. Our evaluation shows that PCCP is able to take advantage of the capacity of zero-slack schedulers to reclaim unused over-provisioning of resources that are only used in critical execution modes. This allows PCCP to accommodate larger blocking terms.

### 5.6.4 GPU Resource Allocation

Graphics processing units (GPUs) are multi-core processors specialized in processing 3D scenes and displaying them on the screen. GPUs are typically included in most graphic cards of desktop computers and a number of laptops. Game designers use GPUs to accelerate the processing of complex 3D scenes and they are required to fine tune the performance of such scenes. Unfortunately, the game industry has followed mostly an ad-hoc approach to the resource allocation and scheduling problem within GPUs. In this project we developed a principled approach based on the quality-of-service resource allocation model (QRAM) [Rajkumar 2007]. To experiment with this approach we built our own experimental platform where we ran multiple experiments to evaluate different allocations policies to maximize utility.

### 5.6.5 Inter-Core Network Link Scheduling

One of the key problems in multi-core technology is the communication between cores, memory, and I/O devices. This communication happens through an inter-core network similar to traditional computer networks we know today (e.g., the internet). As a result, congestion can also be experienced at the inter-core networks, and guarantees for real-time tasks can be violated if the resource allocation is not properly handled. In this area we conducted experiments with the implementation of a fixed-priority scheduling at the link level to evaluate the effectiveness of this scheme in the isolation of real-time tasks from non-real-time tasks. Our experiments showed that fixed-priority is able to achieve this isolation to a large degree, as presented in Figure 5-4. This figure shows the percentage of degradation suffered by a real-time task when additional non-real-time tasks send messages through the inter-core link. Unfortunately, this isolation comes with a price. In particular, it degrades the capacity of the links to support the speed-up that parallel execution brings to application when running in this processor. This can be observed in Figure 5-5, where the speed up of traditional link scheduling is compared with fixed priority.

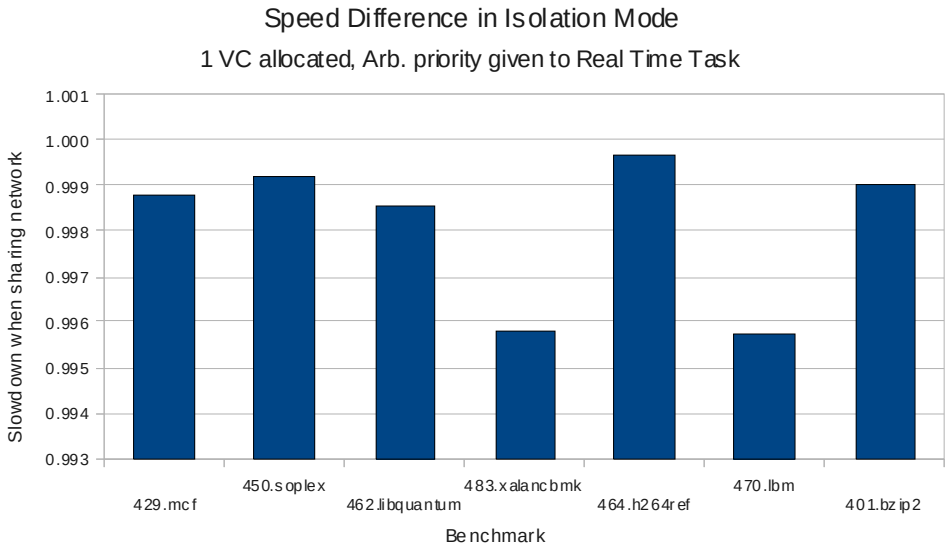



*Figure 5-4: Speed Difference in Isolation Mode*

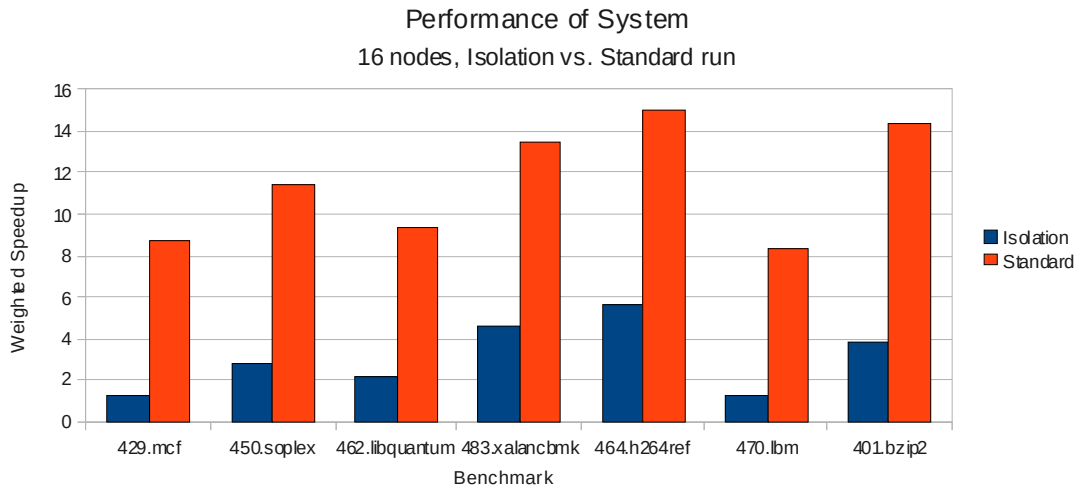



*Figure 5-5: Performance of System*

## 5.7 Publications and Presentations

Karthik Lakshmanan, Dionisio de Niz, and Raj Rajkumar wrote *Resource Allocation in Distributed Mixed-Criticality Cyber-Physical Systems*, which was presented at the 30th International Conference on Distributed Computing Systems in June 2010 in Genoa, Italy.

### 5.7.1 References

**[ARINC 2005]**
ARINC. *Avionics Application Software Standard Interface.* ARINC, 2005. http://www.arinc.com/

**[de Niz 2009]**
de Niz, Dionisio; Lakshmanan, Karthik; & Rajkumar, Raj. "On the Scheduling of Mixed-Criticality Real-Time Tasksets," 291-300. *Proceedings of the 2009 30th IEEE Real-Time Systems Symposium*, 2009. Washington, D.C., 2009. IEEE Computer Society, 2009.

**[Kopetz 1998]**
Kopetz, H. "The Time-Triggered Architecture. International Symposium on Object-Oriented Real-Time Distributed Computing," 22-29. *Proceedings of the First International Symposium on Object-Oriented Real-Time Distributed Computing*, Kyoto, Japan, April 1998. IEEE Computer Society, 1998.

**[Lakshmanan 2010]**
Lakshmanan, Karthik; de Niz, Dionisio; & Rajkumar, Raj. *Resource Allocation in Distributed Mixed-Criticality Cyber-Physical Systems.* International Conference on Distributed Computing Systems 2010. Genoa, Italy.

**[Moore 2008]**
Moore, S. "Multi-Core Is Bad News For Supercomputers." *IEEE Spectrum 45*, 11 (November 2008): 15.

**[Mores 2001]**
Mores, R., Hay, G., Belschner, R., Berwanger, J., Ebner, C., Fluhrer, S., et al. *Flexray: The Communication System for Advanced Automotive Control Systems.* SAE, 2001.

**[Mutlu 2007]**
Mutlu, O. & Moscibroda, T. "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors," 146-158. *Proceedings of 40th Annual Symposium on Microarchitecture,* Chicago, IL, December 2007. IEEE Computer Society, 2007.

**[Oikawa 1998]**
Oikawa, S. & Rajkumar, R. "Linux/RK: A Portable Resource Kernel in Linux." *Proceedings of the IEEE Real-Time Systems Symposium Work-In-Progress,* Madrid, Spain, December 1998.

**[Pellizzoni 2008]**
Pellizzoni, R., D; Bui, B.; Caccamo, M.; & Sha, L. "Coscheduling of CPU and I/O Transactions in COTS-Based Embedded Systems," 221-231. *Proceedings of the 2008 Real-Time Systems Symposium*, Barcelona, Spain, November 2008. IEEE Computer Society, 2008.

**[Rajkumar 2007]**
Rajkumar, R.; Lee, C.; Lehoczky, J.; & Siewiorek, D. "A Resource Allocation Model for QoS Management," 298-307. *Proceedings of the 18th Real-Time Systems Symposium*, San Francisco, CA, December 2007. IEEE Computer Society, 2007.

**[Sha 1990]**
Sha, L.; Rajkumar, R.; & Lehoczky, J. P. "Priority Inheritance Protocols: An Approach to Real-Time Synchronization." *IEEE Transactions on Computers 39*, 9 (September 1990): 1175-1185.

# 6 Automatic Generation of Hidden Markov Models for the Detection of Polymorphic and Metamorphic Malware

Mark Pleszkoch, Tim Daly, and Cory Cohen

## 6.1 Purpose

CERT developed the Pithos tool [Cohen 2008] to cluster and classify malware artifacts based solely on their entry point sequence, that is, the first 100 bytes of code or data starting from the artifact entry point. When Pithos was deployed, it was found to work extremely well in identifying most malware families, but it performs sub-optimally on polymorphic and metamorphic malware. Such malware is constructed from the output of a code generation engine that randomly produces varying Intel code sequences to accomplish the same desired behavior. Polymorphic and metamorphic malware makes use of code generation techniques that randomly produce varying Intel code sequences that accomplish the same desired behavior in order to thwart detection and signature classification methods. These techniques interfere with CERT's existing clustering and identification system, Pithos, and result in the failure to identify approximately nine percent of the entries in the CERT artifact catalog [Cohen 2008]. Accurate identification and classification of malware artifacts is essential for other CERT analysis activities, including malware trend analysis and group analysis of malware families.

The malware identification process for the CERT artifact catalog would significantly benefit from having state machine models for each of the known polymorphic and metamorphic malware families. However, due to the large amount of effort (several person-months) required to construct a state machine model, and the number and size of polymorphic and metamorphic malware families (several hundreds of such families, each with only a few thousand artifacts), it is not feasible to construct state machine models for each polymorphic or metamorphic malware family by hand.

The intent of this research is to automate the construction of state machine models by applying the Hidden Markov Model (HMM) induction framework of Andreas Stolcke and Stephen Omohundro originally developed for speech recognition [Stolcke 1993, Stolcke 1994, Stolcke 1996]. Application of existing work in HMM to the challenge of detecting and classifying malware builds upon prior successful work within CERT in developing a detector for a specific family of malware called Allaple [Pleszkoch 2009]. The prior work was specific to the Allaple family of polymorphic and metamorphic malware. This research attempts to adapt the HMM framework to automate the labor-intensive discovery of an appropriate model for the polymorphic or metamorphic engine, and thus the correct identification and classification of polymorphic and metamorphic malware.

## 6.2 Background

In the context of our problem domain, an HMM is a non-deterministic finite state automaton where the transitions between the states have been assigned probabilities, and the states themselves have been assigned code bytes that are to be generated when the state is visited. For example, in the following example (Figure 6-1) HMM randomly generates an Intel code sequence to advance the EAX register by two.

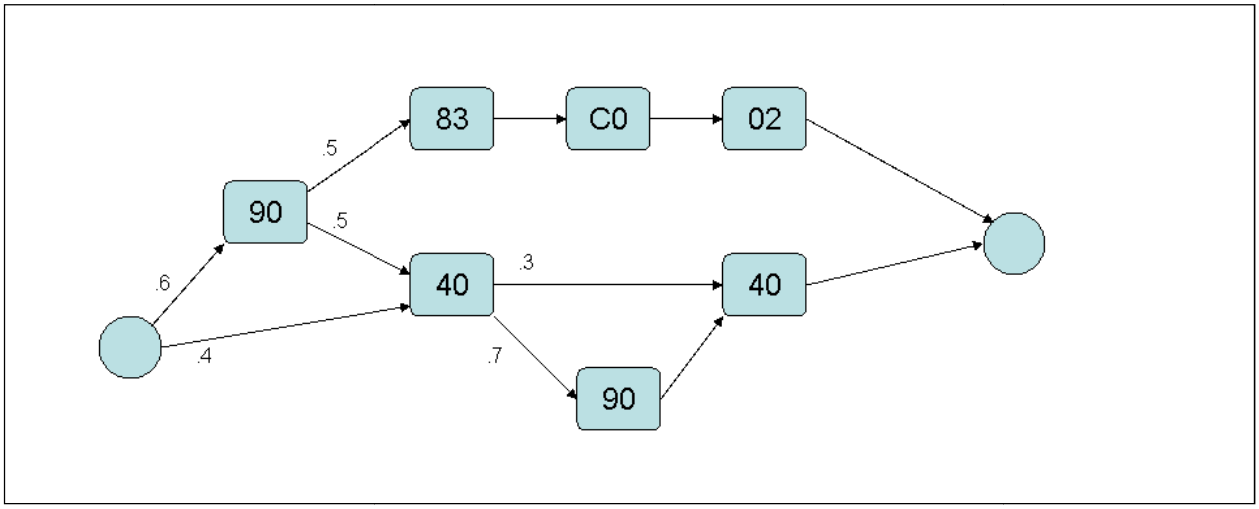



*Figure 6-1: Simple HMM To Generate Behaviorally Equivalent Code Sequences*

The HMM output accomplishes this task either by generating an "add eax, 2" instruction consisting of code bytes "83 C0 02" or by generating two "inc eax" instructions (which increment the EAX register by 1), each consisting of the code byte "40." In either case, null operation "nop" instructions, consisting of the code byte "90," are randomly interspersed or not.

The minimal amount of polymorphism present in this HMM is enough to force the Pithos tool to have to create three separate signatures, one for each of the possible different lengths (two, three, or four bytes) of the generated code sequence.

The adjective "hidden" in Hidden Markov Model refers to the fact that the same byte can be output by more than one state. In the above example, the "40" byte can come from either of two states, as can the "90" byte. This feature makes it much more difficult to reconstruct the state machine when only given data samples of generated output, because the information as to which state each byte comes from is "hidden."

Hidden Markov Models are widely used in the field of bioinformatics to model families of proteins or DNA sequences, except that instead of bytes, symbols representing either amino acids (for proteins) or nucleic acids (for DNA) are attached to each state. Hidden Markov Models are also widely used in the field of speech recognition to model how different human speakers pronounce the same word.

The problem of reconstructing a model from samples of its output is called inductive inference. There are many different approaches to inductive inference, each with its own application areas. Both the fields of bioinformatics and speech recognition use inductive inference algorithms to construct HMMs from their respective sample data. However, the problem domain in each of those fields is sufficiently different from our problem domain so as to render their algorithms ineffective when applied to malware. Interestingly enough, the problem domain in bioinformatics is closer to our problem domain than is the problem domain of speech recognition. The issue is that in bioinformatics, the protein sequences do not contain long strings of null operations or wildly different alternatives that accomplish the same function, so techniques based on multiple sequence alignment are effective in their problem domain, while not producing sufficiently accurate results in our problem domain.

## 6.3 Approach

*Stolcke and Omohundro's HMM Induction Framework*

Among the many different algorithms for inferring Hidden Markov Models from sample data, the HMM induction framework of Stolcke and Omohundro seemed to be the most applicable to the work in this IRAD [Stolcke 1993, Stolcke 1994, Stolcke 1996].The relevance of Stolcke and Omohundro's work to our problem domain is not based on their specific application area of speech recognition, but is instead due to the solid mathematical foundation and generalized HMM induction framework that they provide. The key mathematical question in HMM induction is how to know which of the many valid HMMs is the best fit to the sample data. In some sense, the choice of which HMM is arbitrary, because any HMM that is valid with respect to the sample data could conceivably be the appropriate choice. Stolcke and Omohundro answer that question by placing the problem within the framework of Bayesian optimization, which replaces all the heuristics and ad hoc selection techniques present in other approaches with a single mathematical object (i.e., the prior probability distribution) that serves to encapsulate all the arbitrary choices of those other approaches. This has the effect of giving mathematical meaning to those arbitrary choices, and thus provides guidance in how to make them.

Stolcke and Omohundro decompose the prior HMM probability distribution into three components: a global structure component, an individual state transition structure component, and an individual state transition probability component. Additionally, they provide an option to integrate out the state transition probability component, thus forming a prior probability distribution on the HMM structure alone. Thus, in order to apply their framework to our problem, we had to visit each of these many choices to determine the best option for our problem domain.

Once the HMM induction framework has been specialized to a single, precise mathematical problem, the next step is to develop an algorithm to solve that problem. Here, we faced several challenges.

In their papers, Stolcke and Omohundro compare their approach to the better known Baum-Welch algorithm for HMM determination. The key difference turns out to be that Stolcke and Omohundro's HMM induction framework does an excellent job of determining the HMM structure given the sample data, and only secondarily provides values for the probabilities of the HMM transitions, whereas the Baum-Welch does an excellent job of determining the probabilities of the HMM transitions, but only once the appropriate HMM structure is already known. Since in our situation, the HMM structure is precisely the state machine model that we need to implement our malware detector, Stolcke and Omohundro's approach is the better fit.

Phase 1 of the research plan focused on the process of evaluating the HMM options by creating a Python class library of HMM utility functions with the ability to perform general HMM induction experiments. The library of HMM utility functions allowed us to perform the following tasks:

- handle sample data over an arbitrary base alphabet
- form an initial HMM structure from sample data
- calculate Bayesian statistics (both exact and approximate) for any given candidate HMM structure

- perform Stolcke and Omohundro's greedy single step optimization search strategy, including global prior weighting
- compute merged HMM structures for any given state merging

Our intention was to exercise the above listed tasks to validate the code and the evaluation processes on simple examples of polymorphic and metamorphic test cases. This validation process revealed that examples of HMM induction given in Stolcke were not fully automatic but needed intervention to find solutions [Stolcke 1993, Stolcke 1994, Stolcke 1996]. In addition, we discovered that the Stolcke and Omohundro's search algorithm is totally local, that is, it proceeds through the analysis of a the sample data by moving from solution to solution in the search space until a solution is found or a time bound is elapsed. The problem with such a search algorithm is that it can readily become exponentially sized.

Phase 2 of the research plan was designed to specialize Stolcke and Omohundro's general HMM induction framework to the domain of polymorphic and metamorphic malware over the Intel instruction set. To accomplish this task we created a collection of polymorphic and metamorphic malware test cases ranging from very simple to extremely complicated. We then used the validated code base from the first phase of the research to select the best prior HMM probability distribution for this particular problem domain. We found that the artificial test cases were extremely helpful in evaluating the fit of the Stolcke and Omohundro algorithm to the problem space. Because the HMM is known in the simple test cases, it was possible to check the trial algorithm at each decision point. We found that Stolcke and Omohundro's mathematical approximations are valid in the malware problem domain. In addition we found that the Viterbi path (i.e., the most likely sequence of hidden states in Hidden Markov Models) is primary and preserved under state merging of the algorithm. The conclusion from those initial findings is that if the proper state machine is derived, then the results are valuable in malware classification. That having been said, we also found that even in the simple examples used as test cases, the local search strategy experienced combinatorial explosion of potential paths. That reality altered phase 3 of the research plan from bringing in more complex malware in order to tune the algorithm to this class of problems to looking at alternative strategies.

Phase 3 of the research thus became attempts to deal with the special challenges of the malware problem domain to avoid the O(N-squared) combinatorial explosion. That investigation includes exploration of alternative HMM search strategies, such as the bioinformatics global sequencing algorithms to address the limitations of the strictly local search. That approach, while more effective than Stolcke and Omohundro's local search strategy, is still not effective enough to deal with the problems of scale. We also investigated partitioning the test cases to limit the amount of input data as well as augmenting the malware code sequences with instruction set architecture domain knowledge only to find that although that addresses the N-squared complexity of the greedy state merging, it negates much of the benefits of Bayesian inference.

In Phase 4 we planned to implement the tools developed in the research phase. This phase was not initiated based on the early findings that the tools under investigation were not suitable for the class of problems found in the CERT artifact catalog. Since the anticipated results from the algorithm could not make it through the simple test cases, no attempt was made to apply the tools to the more complex problems of the various polymorphic and metamorphic families of malware in an automated or human-augmented semi-automated fashion.

## 6.4 Collaborations

The Principal Investigators for this project were SEI staff members Mark Pleszkoch, Tim Daly, and Cory Cohen.

Dr. Sean Eddy of the Howard Hughes Medical Institute served as a collaborator for the project on the technical aspects of HMM inference.

All collaborators provided their own support for the work.

## 6.5 Evaluation Criteria

The overarching criterion for the complete success of this project is the number of polymorphic and metamorphic malware artifacts that are correctly identified by the HMM-based detectors produced during the fourth phase of this project. Another important criterion is the extent to which the construction of HMM-based detectors is automatic, requiring little or no human intervention or supervision. Due to both the complicated nature of the Allaple polymorphic malware family and the extremely large amount of data from the many Allaple instances, if the Allaple state machine model can be constructed automatically, it promised a significant research breakthrough.

Criteria for partial success include developing a deeper understanding of HMM inference, and of the power and limitations of the Stolcke and Omohundro approach.

## 6.6 Results

Prior research made clear that HMMs are a potentially useful approach for identifying and classifying polymorphic and metamorphic malware families [Pleszkoch 2009]. The aim of this research was to develop and validate an automated approach to generating HMMs applicable to the problem of characterizing the polymorphic and metamorphic families of malware. We attempted to extend HMM analysis tools that were successfully applied in other domains such as speech recognition [Stolckes 1993] and biological sequence analysis [Birney 2001].

As a result of this research we gained a deeper appreciation for the capabilities and limitations of currently well-understood HMM inference techniques. We found that the exponential explosion of a local search strategy combined with successive merging of submodels (HMM induction-by-merging) is a limitation for the application of HMM induction to the class of problems represented by polymorphic and metamorphic malware. The localized greedy search of the Stolcke-Omohundro algorithms will explore a single path to exhaustion or failure. The probability of making a wrong choice at least once in the algorithm and thus pursuing a false path is high. That probability increases as the complexity of the search space increases. This characteristic limits the practical application of this HMM algorithm to this class of problems. We did not discover a work-around to either enhance the search algorithm or reduce the search space's complexity sufficiently to make this particular algorithm useful. We also did not find any other HMM inference approach that would make the creation of a HMM for this set of problems tractable.

While we did not achieve the research breakthrough that we aimed for, we did develop a deeper understanding of HMM inference as well as the capabilities and limitations of the Stolcke and Omohundro approach. The promise of high reward for being able to use HMM for pattern match-

ing of polymorphic and metamorphic malware is worthy of future research on bounding the problem in such a way as to avoid the impact of combinatorial expansion that stymied this effort.

## 6.7 Publications and Presentations


**[Birney 2001]**
Birney, E. "Hidden Markov Models in Biological Sequence Analysis." *IBM Journal of Research and Development. 45*, 3/4 (May/July 2001): 449-454.

**[Cohen 2008]**
Cohen, F. & Havrilla, Jeffrey S. "Malware Clustering Based on Entry Points." *2008 CERT Research Annual Report.* Software Engineering Institute, Carnegie Mellon University, 2009. www.cert.org/research/2008research-report.pdf

**[Pleszkoch 2009]**
Pleszkoch, Mark & Cohen, Cory. *Detection of the Allaple Polymorphic Packer*. Software Engineering Institute, Carnegie Mellon University, 2009.

**[Stolcke 1993]**
Stolcke, Andreas & Omohundro, Stephen. "Hidden Markov Model Induction by Bayesian Model Merging," 11-18. *Proceedings of Advances in Neural Information Processing Systems 5*. Morgan Kaufmann Publishers, 1993.

**[Stolcke 1994]**
Stolcke, Andreas and Omohundro, Stephen. *Best-First Model Merging for Hidden Markov Model Induction* (TR-94-003). International Computer Science Institute, revised April 1994. http://arxiv.org/abs/cmp-lg/9405017

**[Stolcke 1996]**
Stolcke, Andreas & Omohundro, Stephen. *Model Merging for Hidden Markov Model Induction*. NEC Research Institute, 1996. http://omohundro.files.wordpress.com/2009/03/stolcke_omohundro96_model_merging_hmm_induction.pdf

# 7 Advanced Technology for Test and Evaluation (T&E) of Embedded System Functionality and Security

Richard Linger and Tim Daly

## 7.1 Purpose

Software test and evaluation (T&E) can be a substantial problem for organizations seeking to validate functionality and security in new system development, legacy system evolution, and system acquisition. T&E often consumes significant development resources, yet even thoroughly tested software can exhibit errors and vulnerabilities in field use.

The problem is compounded by the need to repeatedly perform T&E as systems are modified and evolved over operational lifetimes. Even small changes can require extensive T&E efforts to ensure that presumably unrelated functionality continues to perform as expected.

## 7.2 Background

Current T&E methods include testing, scanning, and inspection. Software systems can exhibit massive populations of possible execution paths. Even the best testing efforts can do no more than sample these populations, and executions not sampled go into field use untested. Scanning technology involves use of syntactic signatures to be searched for in software, to help reveal errors, vulnerabilities, or malicious content. Signature recognition can be thwarted by simple obfuscation and potential problems for which signatures do not exist cannot be found.

Inspection techniques are labor intensive and subject to human fallibility, and subject-matter knowledge for informing inspection processes may not be readily available for acquired or legacy software.

Testing, scanning, and inspection are important technologies that will continue to play prominent roles. However, given the stakes involved, it seems reasonable to ask if there could be another way to achieve the goals of test and evaluation.

## 7.3 Approach

The objective of this IRAD study is to investigate a new approach to T&E based on emerging technology for computing the behavior of software. The foundation for this approach is CERT-developed Function Extraction (FX) technology [Linger 2007]. FX automates computation of software behavior with mathematical precision to the maximum extent possible. Figure 7-1 depicts a notional illustration of behavior computation for the simple sequence program of three instructions shown in the upper left (machine precision aside). As shown in the trace table and subsequent derivations, the effects of each instruction can be composed in turn to arrive at the net effect of the program. The trace table contains a row for each instruction and a column for each variable that receives a new value, in this case, x and y. In the cells, final values are expressed in terms of initial values, for example, x1 = x0 + y0 means the final value of x produced by that instruction is the sum of the initial values of x and y. The derivations work from final values on exit from the program back to initial values on entry. The resulting computed behavior is depicted in

the lower right. This statement is called a conditional concurrent assignment. In this case, the condition is "true" (the program always executes) and the concurrent assignments are to x and y, namely, the final value of x is the initial value of y and the final value of y is the initial value of x. That is, the program swaps the values of x and y. Note that these assignments, while written sequentially, are concurrent. Both values on the right are assigned simultaneously to both variables on the left. This is the procedure-free definition of the behavior of the program.

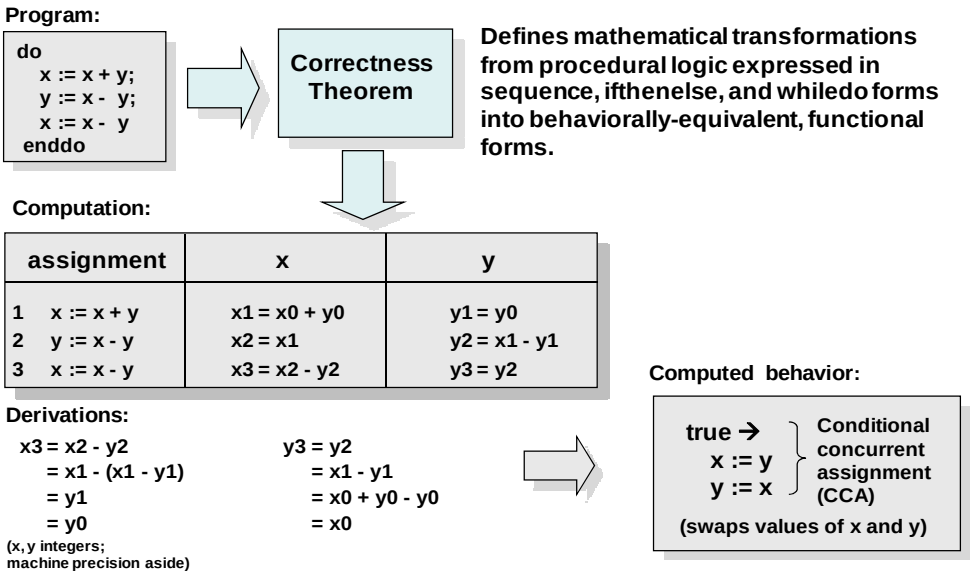



*Figure 7-1: A Simple Behavior Computation*

Behavior computation for branching and looping structures is more complex, and the behavior of large programs will typically contain multiple cases of behavior reflecting variations on input. This example required only one case of behavior. In general, behavior computation produces multiple disjoint cases that cover the entire behavior space, compared to testing which covers the execution space only for those executions tested.

The general process of Function Extraction is depicted in Figure 7-2. First, instructions in an input program are transformed into a functional form that defines their complete effect on the state of the processor, including any effects resulting from the finite precision of operations. These functional semantics are pre-defined in a repository. Next, a Structure Theorem is applied to transform input program logic that may contain confusing jumps and branches (spaghetti logic) into function-equivalent, structured, algebraic form expressed in nested and sequenced fundamental control structures including sequence, ifthenelse, and whiledo [Prowell 1999]. This algebraic structure is a tree of procedural control structures. Next, a Correctness Theorem defines the starting point for transformations of each type of control structure into a function-equivalent, procedure-free definition of its net functional effect, essentially, an as-built specification (Figure 7-1 illustrated this process for a simple sequence structure) [Prowell 1999]. Finally, the computed behavior can be simplified and abstracted through application of pre-defined Semantic Reduction Theorems (SRTs). SRTs have many uses in behavior computation. They define function-equivalent transformations on computed behavior, both for simplification of expressions, and for abstraction to design- and specification-level concepts. SRTs can be developed for functional specifications use-

ful for verification, and for semantic signatures useful for analysis of behavior properties. Application of SRTs in behavior computation is discussed in more detail below.

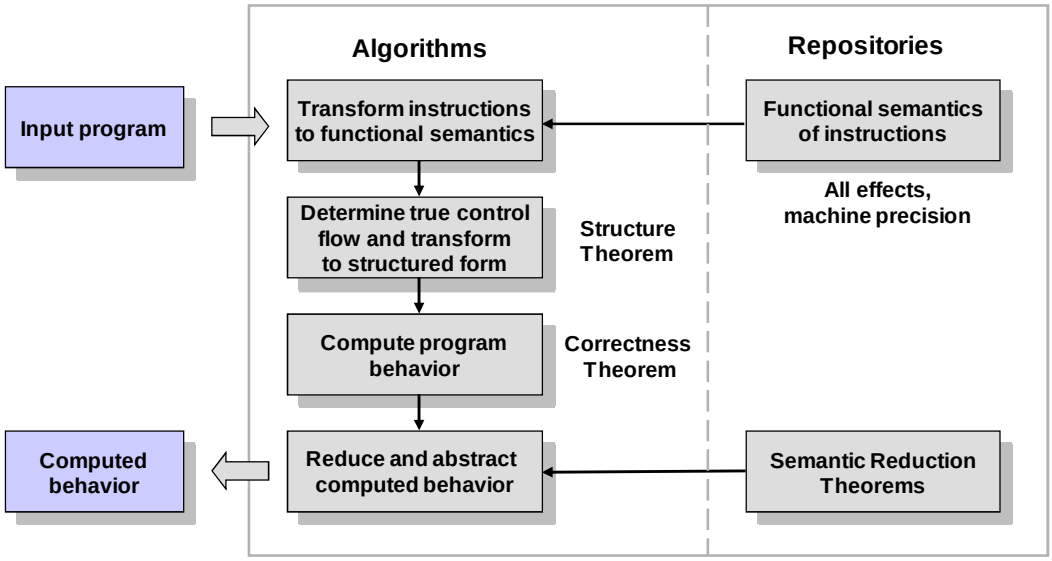



*Figure 7-2: The Software Behavior Computation Process*

Automated behavior computation has been applied to malware analysis. The purpose of this IRAD study is to investigate use of the technology beyond malware, specifically, as a means to address objectives of software test and evaluation [Burns 2009, Linger 2008, Bartholomew 2007]. Behavior computation does not depend on execution, as in testing, nor on signatures, as in scanning. As such, it has the potential to avoid limitations of these T&E methods. A key question for this study is whether the capabilities of behavior computation can be effective in assessing the functionality and security of software. Success could encourage further investigation of how to apply behavior computation to augment or possibly replace certain types of functional testing, particularly at the unit and subsystem levels.

For purposes of this IRAD study, embedded assembly language code from a robot arm control program was selected for analysis. Robot arm computations require matrix operations for calculating spatial rotation and translation maneuvers. Matrix multiplications (cross products of the form A x B → C) are important in these operations, and this code was chosen for the behavior computation experiments. Four operators for use in SRTs were developed to characterize the mathematical process of matrix multiplication:

*Vector memory shape*:
Matrix multiplication operates on vectors of elements. This operator has parameters to define the vector start location, number of elements, and stride, that is, the size of steps to take when reading a vector from memory, in number of bytes.

*Dot product*:
Matrix multiplication can be expressed using the dot products of vectors. The dot product operator has parameters representing the two vectors for which to compute the dot product.

*Matrix memory shape:*
Matrix multiplication requires that the vectors form matrices. A matrix is represented as an operator whose parameters represent the starting address of the matrix in memory and the number of rows and columns in the matrix.

*Matrix multiplication:*
Matrix multiplication must produce a matrix of dot products. The matrix multiplication operator has parameters representing the two matrices to multiply together.

Using these operators, SRTs were developed to first recognize dot products of vectors and then recognize matrix multiplications. For the experiment, three versions of the matrix multiplication code from the robot arm control program were analyzed as described below.

### 7.3.1 Original Code Version

No changes were made to the as-coded assembly language found in the embedded software. The results of the behavior computation are depicted in Figure 7-3. This is a procedure-free conditional concurrent assignment produced by the FX system, of the same general form as shown in the example of Figure 7-1, but accounting for effects of the code on processor registers, memory, and flags.

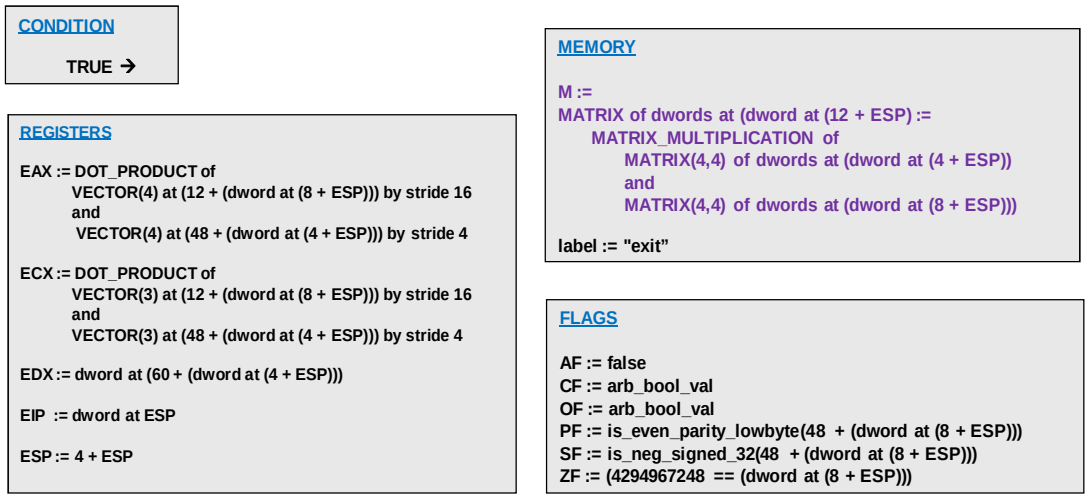



*Figure 7-3: Behavior Computation for As-Coded Matrix Multiplication*

The condition on the behavior is "true" because the code executes unconditionally. Note that memory (M) contains a MATRIX of double words (dword) starting at location 12 + the stack pointer (ESP), specifically, a MATRIX_MULTIPLICATION of a 4x4 MATRIX starting at the location defined by ( 4 + ESP) and a 4x4 MATRIX starting at the location defined by ( 8 + ESP).

Thus, the computed behavior for the original code from the robot arm satisfies the SRTs that define a matrix and a matrix multiplication, both of which depend on SRTs for vector and dot product definition. Given that these SRTs are themselves correct (SRTs can be verified by theorem provers), the code correctly implements a matrix multiplication. The computation also reveals that the registers contain residual values involving dot products and vectors, and the flags contain residual values as well.

### 7.3.2 Incorrect Code Version

Figure 7-4 depicts computed behavior for the matrix multiplication code now containing an error, specifically, values in the target matrix (C matrix) are not initialized to zero as in the correct version.

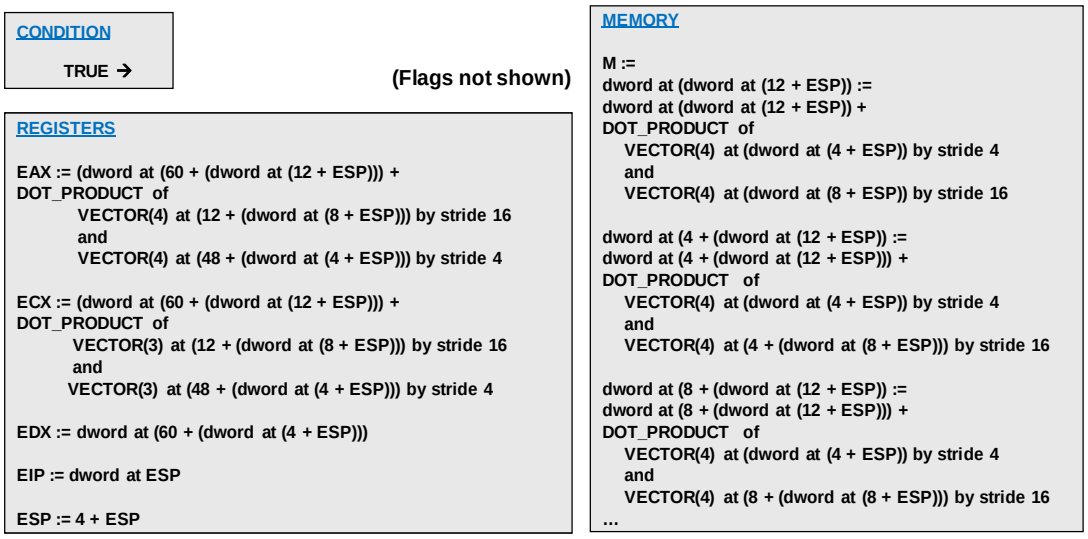



*Figure 7-4: Behavior Computation for Matrix Multiplication Containing an Error*

In this case, the computed behavior reveals that the code no longer computes a matrix multiplication. Memory M now contains a series of double words, each the sum of itself and a dot product of vectors. That is, the code still satisfies SRTs for dot products and vectors, but not the SRT for matrix multiplication. The fact that each of these locations is a sum of itself and a computation that should ultimately contribute to a matrix multiply reveals the problem. The sum is present because the location is not initialized to zero, and any value present on entry will contribute to the final value on exit. Initializing the target matrix will solve the problem.

### 7.3.3 Malware Code Version

Figure 7-5 displays computed behavior for the matrix multiplication obfuscated with a large number of arbitrary jumps, instructions interleaved, and malware inserted with instructions dispersed throughout the code.

In this case, note that the condition is no longer simply “true.” A condition has been revealed involving SRTs named “create file succeeded” and “write file succeeded.” Since this code should not be creating and writing files, something is obviously amiss. The behavior of the code now includes effects on the file system and the operating system. The behavior shows the code is now writing a file beginning at buffer byte 0 and ending at byte 6738. It turns out that is the exact length of the code—the inserted malware is writing the code it is embedded within into the file system of the machine. It is a self replicating virus. Note that the matrix multiplication is itself unaffected by the malware, but that hardly matters at this point. In addition, this behavior computation produced three other cases of behavior (not shown), all of which represented errors in the malicious code that prevented the self-replication.

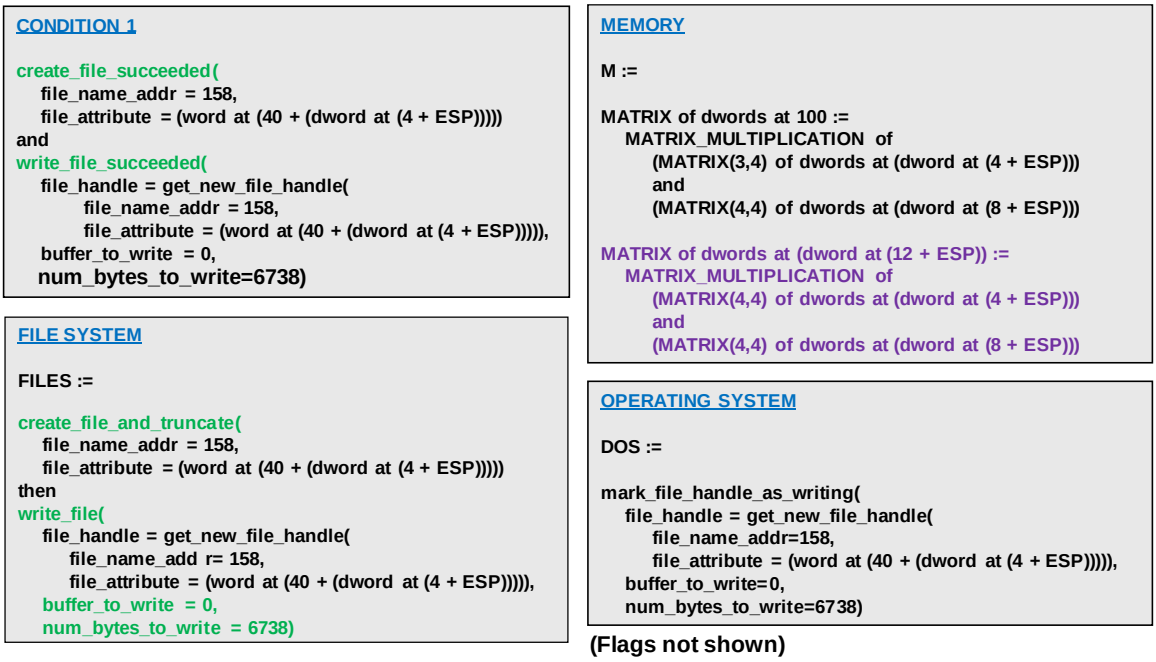



*Figure 7-5: Behavior Computation for Matrix Multiplication with Obfuscation and Embedded Malware*

## 7.4 Collaborations

The Principal Investigators for this project were SEI staff members Richard Linger and Tim Daly. Staff members Kirk Sayre and Mark Pleszkoch also made contributions to the work.

Dr. Alan Hevner of the University of South Florida and Dr. Gwendolyn Walton of Florida Southern College served as collaborators for the project. These researchers have extensive experience in behavior computation technology and strong backgrounds in software test and evaluation. Both collaborators provided their own support for the work.

## 7.5 Evaluation Criteria

The success criteria involved two objectives:

- successful adaptation of the current FX system for the T&E demonstration
- successful computation of behavior for original, incorrect, and malicious versions of the code as an alternate means for test and evaluation of functionality

## 7.6 Results

This IRAD project demonstrated that behavior computation could be employed to reveal the functional behavior of a component of an embedded software system, whether correct, incorrect, or compromised with obfuscation and embedded malware. The FX system was adapted to handle instructions in the robot control program language. Definition of subject-matter SRT specifications, which themselves could be independently checked with theorem proving systems, permitted abstraction and automated checking of the computed behavior for T&E purposes. The behavior computation system and the subject-matter SRTs permit repeated checking as necessary with little additional effort should the code require update or modification during its operational lifetime.

These SRTs are available for analysis of matrix multiplication implementations in whatever context they may appear.

Based on this initial study, behavior computation may have potential as a means to augment or replace certain forms of functional testing. More generally, it may help to achieve T&E objectives earlier in the life cycle of new software development as a check on code as it is written, and later in the life cycle as a check on code as it is evolved for new requirements. It may also prove useful in T&E activities for legacy and acquired software, where documentation and even source code may be unavailable. This latter focus was illustrated in this study, where the existing embedded code was subjected to a reverse engineering process through behavior computation. In either case, definition of subject-matter SRTs permits checking of computed functionality and provides documentation for maintenance and evolution. Beyond this study, an appropriate next step could be to investigate scale up of the process, perhaps through pilot projects with SEI sponsors.

## 7.7 Publications and Presentations

### 7.7.1 References


**[Bartholomew 2007]**
Bartholomew, R., Burns, L., Daly, T., Linger, R., & Prowell, S. "Function Extraction: Automated Behavior Computation for Aerospace Software Verification and Certification." *Proceedings of AIAA Infotech@Aerospace 2007 Conference,* Rohnert Park, CA, May 2007. American Institute of Aeronautics and Astronautics, 2007.

**[Burns 2009]**
Burns, L., & Daly, T. "FXplorer: Exploration of Computed Software Behavior: A New Approach to Understanding and Verification," 49. *Proceedings of 42nd Hawaii International Conference on System Sciences,* Waikoloa, Big Island, Hawaii, January 2009. IEEE Computer Society, 2009.

**[Linger 2008]**
Linger, R., Pleszkoch, M., & Hevner, R. "Introducing Function Extraction into Software Testing." *The DATA BASE for Advances in Information Systems 39,* 3 (August 2008): 41-50.

**[Linger 2007]**
Linger, R., Pleszkoch, M., Burns, L., Hevner, A., & Walton, G. "Next-Generation Software Engineering: Function Extraction for Computation of Software Behavior," 277. *Proceedings of 40th Hawaii International Conference on System Sciences,* Waikoloa, Big Island, Hawaii, January 2007. IEEE Computer Society, 2007.

**[Prowell 1999]**
Prowell, S., Trammell, C., Linger, R., & Poore, J. *Cleanroom Software Engineering: Technology and Process.* Addison Wesley, 1999.

# 8 An Investigation into the Feasibility of Tactical SOA

Soumya Simanta, Dan Plakosh, Joseph Seibel, William Anderson, and Edwin Morris

## 8.1 Purpose

*A Stryker resupply mission to an outpost in a war zone has been hit by an improvised explosive device (IED). Two soldiers were critically injured in the explosion. Two other soldiers were also injured but are able to return incoming fire from surrounding locations.*

*A patrol sets out to reach and support the unit. The patrol must traverse unfamiliar terrain and pass near several villages. They want to keep off main roads to avoid ambush and potential IEDs. Historical information about the area, UAV imagery and other sensor data is available as a set of services that can be discovered and that provide situational awareness about enemy activity.*

*The goal of the patrol is to reach the Stryker and its crew, provide fire support to keep the enemy at bay, and aid the wounded until additional help arrives.*

Just a few years ago, a patrol setting out to relieve the Stryker might have only voice radio contact back to base. Real-time information (e.g., sensor data and images from UAVs flying overhead) and historical information (location of friendly and enemy) were not directly available to the patrol. That situation is rapidly changing. Soldiers on patrol are or will soon be linked back into the network as information gatherers and users with capabilities such as the Tactical Ground Reporting system (TiGR), and as recipients of situational awareness data via increasingly sophisticated, but often special-purposed devices [DARPA 2010a]. In the near future, one can imagine a digital battlefield populated with a variety of sensors and enterprise data sources that assist the warfighter and are accessible via general-purpose handheld devices like commercial smartphones.

Several major research efforts in DoD are underway to achieve this vision, most notably the DARPA Transformative Apps program that is intended to place mobile software applications ("apps") into the hands of warfighters as they are needed by employing commercial smartphone standards and implementations [DARPA 2010b].

One potential approach to facilitating handheld adoption involves the use of service-oriented architectures (SOA) to access data on the handheld device. SOA provides several critical advantages to the military:

- SOA is based on a standard interface format that improves interoperation through the definition of services that are specified and accessed in a language and platform independent manner. By use of SOA, it may be possible to build interface specifications consistent within entire classes of systems (e.g., UAVs, cameras providing real-time video feed, auditory, chemical, and biological sensors) such that the warfighter does not need specialized equipment to access the data. Standardizing the data-access interface also means that the development time to integrate existing capabilities to a new device reduces, therefore reducing the overall time-to-field time for these devices.

- SOA supports the notion of service discovery that allows warfighters to identify and locate critical capabilities that are available based on the warfighter's context (location, time). For example, services may provide historical data, images of leaders in a village located by GPS positioning, and data from auditory sensors located within or around the position.

SOA was initially developed in response to enterprise problems of distribution, interoperation, and reuse of capabilities representing business processes. SOA as commonly applied relies on community standards (e.g., WS-* standards, Representational State Transfer (REST) [W3C 2010, Fielding 2000]) to define interfaces and messages, provide security, and support other functions (e.g., reliability, addressing, discovery) necessary for constructing distributed, loosely coupled capabilities that can be tied together in order to address business needs. Where SOA has made fewer inroads, however, is among tactical systems. Such systems normally execute in resource-constrained environments that are not perceived to be an easy fit for SOA. Within the tactical systems and environments of the DoD enterprise, a variety of architectural paradigms and technologies exist. The paradigms and technologies—and ultimately the systems—tend to interoperate poorly and often provide redundant capabilities. These limitations are exactly those that SOA was originally intended to address—albeit in an enterprise context.

The emergence of commercial handheld technology and resulting market explosion[6] along with the potential for application of this technology to tactical environments presented us an opportunity to consider issues of quality of service and service discovery in tactical environments when applying SOA strategies. Our goals for the IRAD were to

- determine whether SOA approaches employing Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP)-were viable in light of the potential limitations of smartphones. In addition, we sought to identify limitations and changes that are required to make SOA work for tactical situations.
- enhance our expertise in an emerging architectural paradigm that employs lightweight, handheld devices to provide information to "edge users" such as soldiers and first responders, and allows these users to provide information back to base
- employ existing technology, standards, and open source implementations where possible to build a prototype
- determine whether reasonable quality of service for performance and security could be delivered on the handheld using existing service-oriented principles and open source off-the-shelf implementations supporting these principles

## 8.2 Background

Other researchers have not overlooked the value of extending the SOA beyond enterprise environments. Trifca provides direction on applying Representational State Transfer (REST) to model the functions of devices and make them available on the web as services that can be assembled in an ad hoc manner [Trifa 2010]. REST exploits the architectural style of the World Wide Web to

[6] 45.4 million people in the U.S. owned smartphones at the end of February [2010], up 21 percent from the previous three months. (http://www.informationweek.com/news/mobility/smart_phones/showArticle.jhtml?articleID=224201881)

access resources via Uniform Resource Locators (URLs). REST relies on Hypertext Transfer Protocol (HTTP), which presumes a reliable transport mechanism—typically Transmission Control Protocol (TCP). This presumption, while appropriate for connections established among devices residing on reliable networks like the internet, is not appropriate for devices residing on the unreliable ad hoc wireless networks used in tactical situations and crises.

A number of authors have addressed quality of service (QoS) issues for SOA operating in constrained environments. Hafsoe et. al. recommend adaptations to web services for limited bandwidth environments, including use of a combination of publish/subscribe and request/response interaction models, and optimization of information transported to the tactical edge [Hafsoe 2007]. Other recommendations include use of proxies that automatically subscribe to key capabilities on behalf of clients, content filtering, and caching to optimize information flow. The recommendations focus on improving the performance of web services by reducing the volume of network traffic and data transmitted and use of the NATO Standard Military Message Handling Systems. The authors conclude by stating that adapting SOA web services for use in disadvantaged grids is a challenging task and more work remains before SOA can be applied at the tactical level.

Natchetoi and associates propose a lightweight SOA based strategy for mobile devices that allows them to function in disconnected mode and that uses efficient mechanisms for service invocation [Natchetoi 2008]. Srirama and associates have developed the Mobile Web Services Mediation Framework (MWSMF) that acts as an intermediary between consumer/providers operating in relatively unconstrained environments and provider/consumers operating in more constrained, mobile environments like smartphones [Srirama 2007]. MWSMF is intended to enhance quality of service by mapping across differing security mechanisms, compressing messages to enhance performance, providing guaranteed message delivery, handling faults, and supporting transactions. Ahmed et. al. focus on improving the availability of services through a dynamic service replication mechanism that places replicas in zones based on demand [Ahmed 2009].

The Devices Profile for Web Services (DPWS) (approved as a web services standard in 2009) is a standard for messaging, discovery, description, and event triggering on resource-constrained devices [OASIS 2009]. Jammes, Mensch, and Smit describe the use of DPWS for integrating old and building new web service interfaces for resource-constrained devices [Jammes 2007]. Pruter and associates discuss how aspects of DPWS can be adapted for real time capability [Pruter 2008]. However, DPWS is not specifically aimed at tactical environments supported by ad hoc, wireless networks.

Several authors have addressed issues of service discovery. Sheu, Czajkowski, and Hofmann report on a proposed architecture supporting publication and discovery of sensors wrapped with SOA service interfaces in peer-to-peer sensor networks [Sheu 2007]. Halonen and Ojala address a design strategy for providing SOA in mobile ad hoc networks (MANETs) [Halonen 2006]. The strategy involves integrating service discovery into the routing protocol of the MANET such that services are detectable from anywhere within the network without manual configuration. The authors produced a prototype implementation using the Optimized Link State Routing (OLSR) protocol and analyzed functional and performance capabilities. Suri and associates have also developed a SOA solution for MANETs that employs a decentralized, peer-to-peer approach for service discovery [Niranjan 2009]. In addition, this solution supports service migration, whereby a

running service can be stopped, its state saved, moved to another node, and restarted. This improves survivability of services by supporting migration to less loaded nodes at runtime. The authors claim good performance levels for service discovery.

In addition to research in the basic problems in the use of SOA in tactical environments, a growing number of U.S. and allied military efforts are relevant to our work. A few are detailed in Table 8-1.

*Table 8-1: Related Efforts*

| | |
|---|---|
| MITRE Tactical Edge Characterization Framework | The Tactical Edge Characterization Framework identifies characteristics of DoD tactical environments and relates them to SOA [Dandashi 2007]. The Tactical Edge Characterization Framework includes four components:<br>• a common vocabulary for characterizing tactical environments<br>• SOA design patterns that can be applied at the tactical edge<br>• an example identifying how these design patterns could be composed to address a tactical scenario<br>• infrastructure requirements derived from the example |
| CES2TE Focus Team | This team concluded that Core Enterprise Service (CES) solutions were not in use down to the tactical edge. This is at odds with the notion that Global Information Grid (GIG)-enabled, net-centric warfighters will be able to share information with each other quickly and easily [CES2TE 2008]. |
| U.S. Navy CANES | This project is intended to develop a common computing environment tactical network for the Navy. SOA-related goals of CANES include proliferation of net-centric enterprise services (NCES) core services (SOA services) to the tactical edge, modification of existing C4ISR applications to leverage SOA, and deployment of elements of SOA strategies into existing PORs [Anderson 2009]. |
| MIT Lincoln Labs | Conducted live flight tests to measure the performance of three different SOA-based systems when delivering track data in an airborne network [Huang 2007]. |
| Military Communication Institute, Poland | Development of a tactical SOA–based C4I system providing situational awareness. The work uses a mediation service that interacts with service producers and consumers, determines whether acceptable quality of service (QoS) can be delivered, and realizes the delivery of the service [Sliwa 2009]. |
| Raytheon RATS | Entered the Android marketplace a handheld device and is building applications for intelligence collection and analysis such as license plate reading, streaming video camera feeds, and biometric collection [Woyke 2009]. |
| DARPA | Recent Broad Area Announcement (BAA) with the stated goal “to place the right mobile software applications (apps) into the hands of warfighters as the apps are needed.” This BAA will build a marketplace and fund research on middleware services, libraries, and applications targeted at open source mobile platforms [DARPA 2010b]. |

## 8.3 Approach

The top-level goal of the Tactical SOA IRAD was to determine whether the SOA paradigm, along with the current state of SOA technology, is applicable to tactical settings. To achieve this goal, we conducted several experiments and implemented a series of prototypes that demonstrate the viability of using service-orientation (via SOAP-based web services) and mobile handheld technologies (Android mobile computing stack) for identifying available tactical assets and using those assets to gather situational awareness data. The use case implemented provides information

from heterogeneous assets, including unmanned aerial vehicles (UAVs). Specific user features of various prototypes provided on the smartphone include

- display of map data and plotting of geo-location of assets of interest on the map. For example, the actual geo-location of a UAV or a car is overlaid on the map and updated at regular intervals (almost real-time)
- display of real-time video from assets that are capable of capturing and transmitting real-time video (e.g., UAV)
- broadcasting live video feeds from a handheld device (Nexus One) to other handheld devices
- plotting and displaying geo-locations of other handheld devices, for example, those carried by related personnel or deployed on robotic devices
- drawing areas of interest on the map that can then be shared instantly with all other assets that can receive CoT data
- viewing a radar display that shows the relative directional geo-locations of all other assets in the area

Experiments and prototype development was carried out in the SEI Research, Technology, and System Solutions (RTSS) Concept Lab. Prototypes were tested at the United States Special Operations Command-Naval Postgraduate School (USSOCOM-NPS) Field Experimentation Cooperative Capabilities Based Experimentation (CBE) lab at Camp Roberts, California.

## 8.4 Collaborations

The SEI team included Soumya Simanta, Dan Plakosh, Joe Seibel, Bill Anderson, and Ed Morris, all members of the RTSS program at the SEI. The team was provided the opportunity to test prototypes at Camp Roberts by Dr. Ray Buettner, Director of NPS Enterprise Field Experimentation, and worked closely with Dr. Alex Bordetsky, Eugene Bourakov, and Mike Clement, all of NPS, to define and implement field tests.

The central feature of this field experimentation is a quarterly research event, typically at Camp Roberts, where university researchers, military units, government agencies, and industry can explore new concepts in science and technology. Researchers have access to a fleet of manned and unmanned aerial vehicles, ground vehicles, a tactical operations center, restricted air space, and secure networks of different types (e.g., wave relay, mesh, Wi-Fi, cellular).

Access to the Camp Roberts infrastructure and field experiments and the help provided by our NPS colleagues was invaluable and ensured that our prototype was aligned with real user needs. The team also received advice and feedback from active duty U.S. military personnel at Camp Roberts that helped us to focus our experimentation. We particularly appreciate this advice.

The team also appreciates the efforts of Jayson Durham (Space and Naval Warfare Systems Command Pacific) to bring the SEI team into the larger DoD community working on the topics of enterprise lexicon services, sensors, and interoperability.

## 8.5 Evaluation Criteria

The team employed both subjective and objective criteria in analyzing the success of the IRAD. The primary subjective criterion was the degree of interest in our work by active duty military personnel at Camp Roberts. This criterion reflects interest in the functional capability that would be provided by our handheld field experiments as well as the ability to provide reasonable quality of service as judged by military personnel. In addition to this admittedly informal measure, we also gathered objective empirical data[7] regarding the performance of the prototype in several key areas:

- overhead of transforming data (cursor-on-target data) providing provided by tactical assets (e.g., sensors, UAVs etc) into SOAP messages and parsing of these messages by the smartphone
- comparison of SOAP over TCP and User Datagram Protocol (UDP) protocols to provide real time video feeds to a smartphone
- overhead involved in smartphone-to-smartphone SOAP messaging using an intermediary for routing
- overhead of support for network and application level security appropriate to our target environment

## 8.6 Results

### 8.6.1 Initial Experiments

Our initial experiment involved transmitting Motion JPEG (MJPEG) images from a wireless IP camera across an internal wireless network using HTTP/TCP transport to our Nexus One Android smartphone. TCP is well suited for data where reliable transmission of data is essential, and where the network infrastructure itself can support reliable transmission. Where TCP is less useful, however, is in situations where loss, packet ordering, or garbled packets are more common.

We quickly noticed a significant lag between the time a video frame was received from the camera and transmitted across the wireless network and the time the image was actually displayed on the phone. The lag was most significant when the network was heavily loaded—a situation that typically results in lost, out of order, or garbled packet transmission. In this situation, TCP retransmits data in order to maintain reliable delivery. We attributed the lag to these retransmissions and to maintaining guaranteed delivery order of packets. Since real-time video is time-sensitive, retransmitting data and maintaining order introduces delays that result in poor video quality to the end user. In addition, TCP has significant startup latency, and does not provide a broadcast or multicast capability that is useful in tactical situations where the same data must be pushed to multiple end users.

The disadvantages of TCP drove us to re-implement the service using UDP as a transport protocol. UDP is often preferred for time-sensitive applications (such as real-time video streaming) where it is preferable to drop a packet rather than wait for the resend of a packet. UDP does not

[7] Our data collection activity is ongoing, and we are developing plans to gather additional data.

require establishing a connection between client and server, and assumes that error correction is not necessary or is handled elsewhere. In the case of live video feeds in "lossy" networks where it is more important to handle the incoming packet rather than worry about a resend of a now out-dated packet, the application may simple chose to ignore the error. Thus, UDP avoids the over-head associated with guaranteed message delivery and is preferable in situations where "old" data loses meaning (e.g., real-time video feeds).

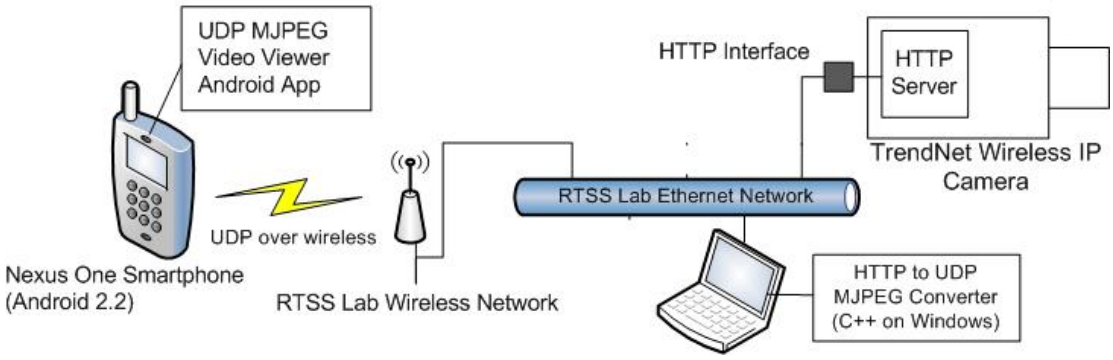



*Figure 8-1: Use of UDP to Transport Video Frames*

This switch to UDP involved developing a HTTP/TCP to UDP converter for transport of MJPEG images to the Nexus One smartphone as depicted in Figure 8-1. The experiment employed a Trendnet wireless IP camera that was connected to a Windows machine running a TCP to UDP MJPEG converter (C++) through a wired Ethernet network. The TCP to UDP converter gets MJPEG image data over TCP and retransmits that same data using UDP to the RTSS lab wireless network. The Android smartphone is connected to the RTSS lab wireless network and can receive these MJPEG frames over UDP and display them using an Android application.

When we switched to UDP, we saw significant improvement in overall video performance. Any disadvantages due to lost or garbled packets were minor. Figure 8-2 indicates inter-arrival times under high network load conditions for MJPEG frames when using the TCP and UDP transport strategies. In general, the UDP strategy demonstrated more consistent arrival and reduced lag between frames compared with TCP that we attribute to receptiveness to frames regardless of loss or out of order delivery conditions. TCP exhibited frequent spikes that we attribute to ensuring reliable delivery. However, as we discuss in subsequent sections, the decision to go outside the mainstream of Web service technology has had many consequences.

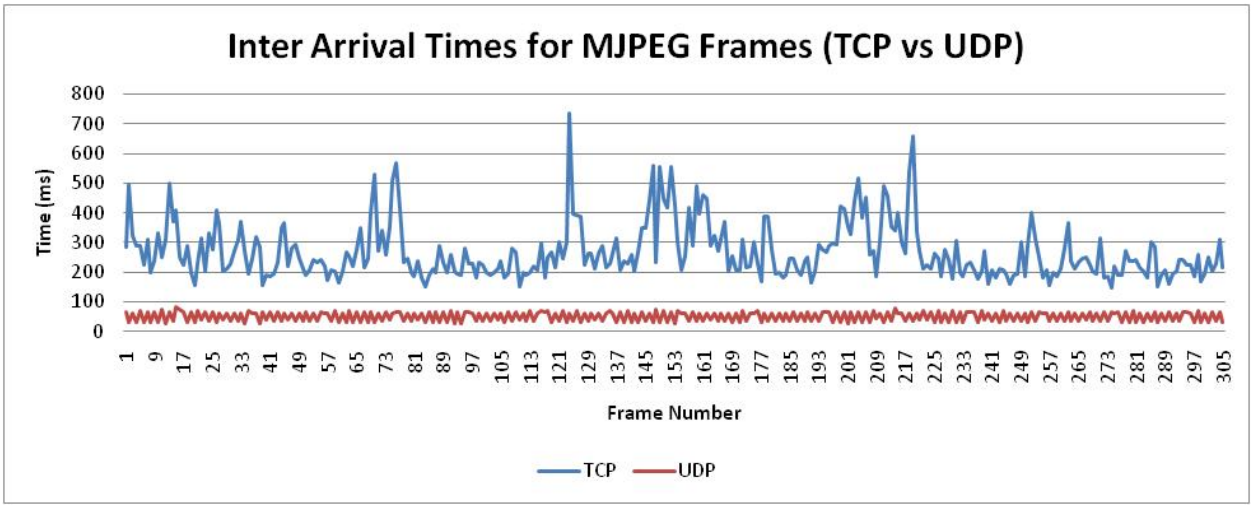



*Figure 8-2: TCP Versus UDP Inter-Arrival Times*

### 8.6.2 Prototypes

Subsequent to defining our UDP strategy in the RTSS Concept Lab experiments, we began developing a series of prototypes for demonstration to active duty military personnel at the NPS Center for Network Innovation and Experimentation (CENETIX) testbed.[8] Prototypes were demonstrated in May and August 2010.

Our initial (May) prototype supported a scenario involving smartphone access to video images being transmitting from a UAV (Figure 8-3). In this scenario, assets (e.g., UAVs, cars) track a hostile vehicle and post messages to the SEI-developed cursor on target (CoT) SOAP service that sends CoT data as SOAP-over-UDP messages. CoT is an XML-based message format developed by MITRE Corp. and used by the U.S. Air Force, Marine Corps, and special operations forces in live, unmanned aircraft operations for UAVs such as the Raven and Predator [Robbins 2007].

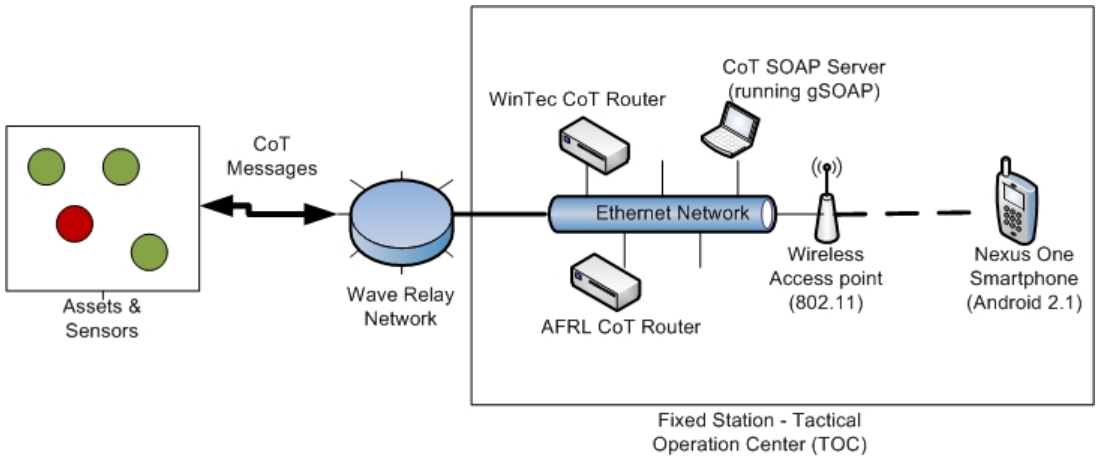



*Figure 8-3: UAV to Smartphone Video*

[8] For information about the testbed, see http://cenetix.nps.edu/cenetix/

In Figure 8-4, images received by Nexus One Android smartphone display positional information (left image) and data from a UAV tracking a car (right image).

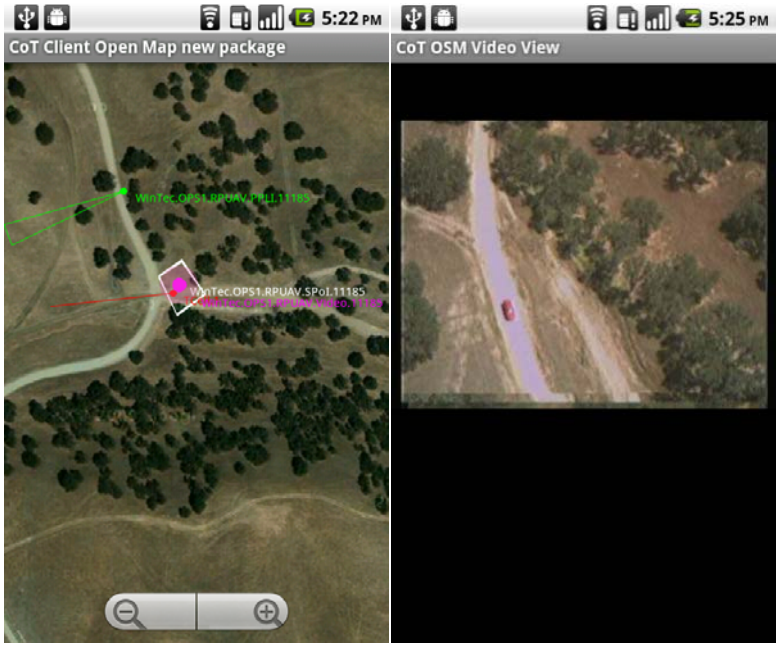



*Figure 8-4: Nexus One Receiving UAV Images*

The SEI team received excellent feedback on this demonstration from active duty military personnel, who identified several additional capabilities they wanted to see on the phone. In response to these requests and our desire to push the technology further, the team developed two additional prototypes that were demonstrated in integrated field experiments at the NPS Field Experiment Cooperative in August 2010. In the first of these experiments, warfighters disguised as civilian tourists employed Android smartphones that were connected to a Wi-Fi access point on a mobile station (car). The mobile station was connected to a wave relay network supporting access to a fixed station tactical operations center (TOC) that was also receiving messages from other assets (Figure 8-5). The smartphones consumed CoT messages from assets such as a UAV, and also transmitted video feeds back to the TOC and to other smartphones. Thus, warfighters in the field carrying smartphones as well as those operating the TOC could see video from the UAV as well as live video from smartphone cameras. In the integrated scenario, warfighters in the field were searching for and transmitting images of electronic devices that had been deployed (presumably by enemy forces) in the environment.

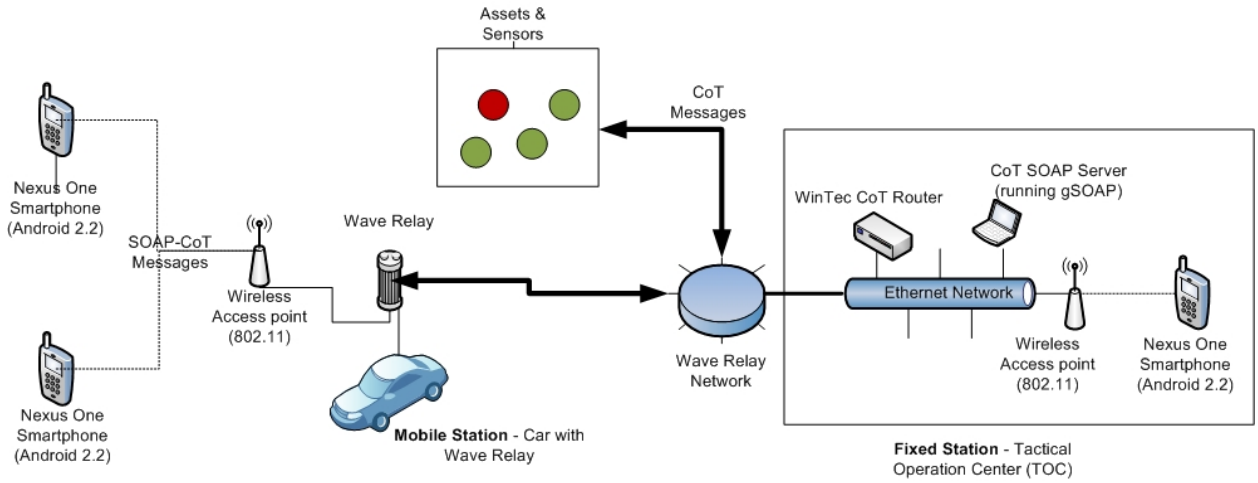



*Figure 8-5: Phone to Phone/TOC Video*

Our final prototype, also demonstrated at Camp Roberts in August, employed an IP camera and Android phones (inside the TOC at Camp Roberts) and captured video frames and converted them into SOAP-over-UDP CoT messages. These SOAP messages were sent to both local service consumers (other phones and CoT display) as well as remote service consumers (phones inside the RTSS lab network in Pittsburgh) (Figure 8-6).

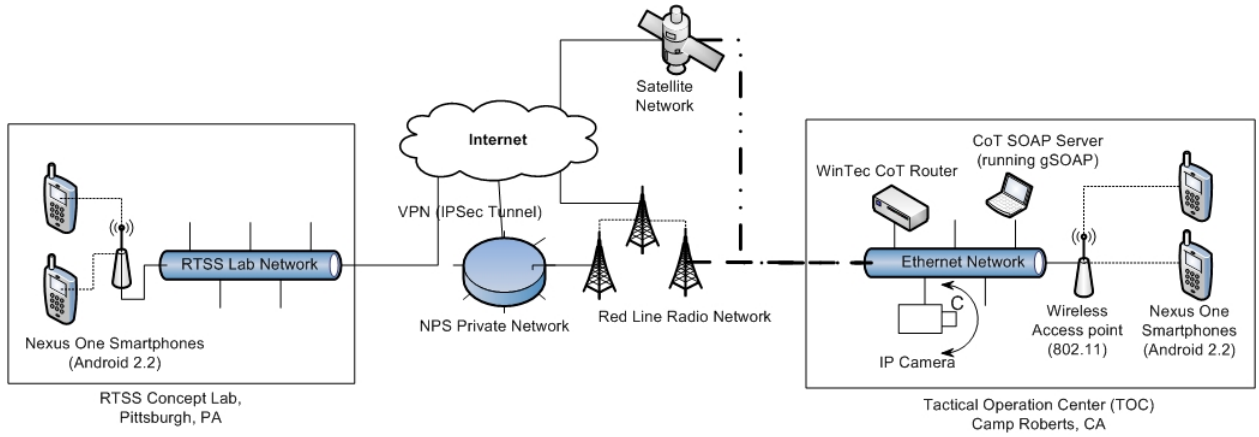



*Figure 8-6: Camp Roberts to Pittsburgh Conferencing*

Remote phones (inside the RTSS lab) were also broadcasting live video feeds as SOAP-over-UDP CoT messages to the TOC in Camp Roberts via a virtual private network (VPN) connection.

Figure 8-7 provides an image displayed in the TOC demonstrating video feed from the smart-phone camera at Camp Roberts (top- left image of three displayed in the upper right corner of a larger map image), from the IP camera at Camp Roberts (top-right), and an image of a doorway from the SEI RTSS Concept Lab.

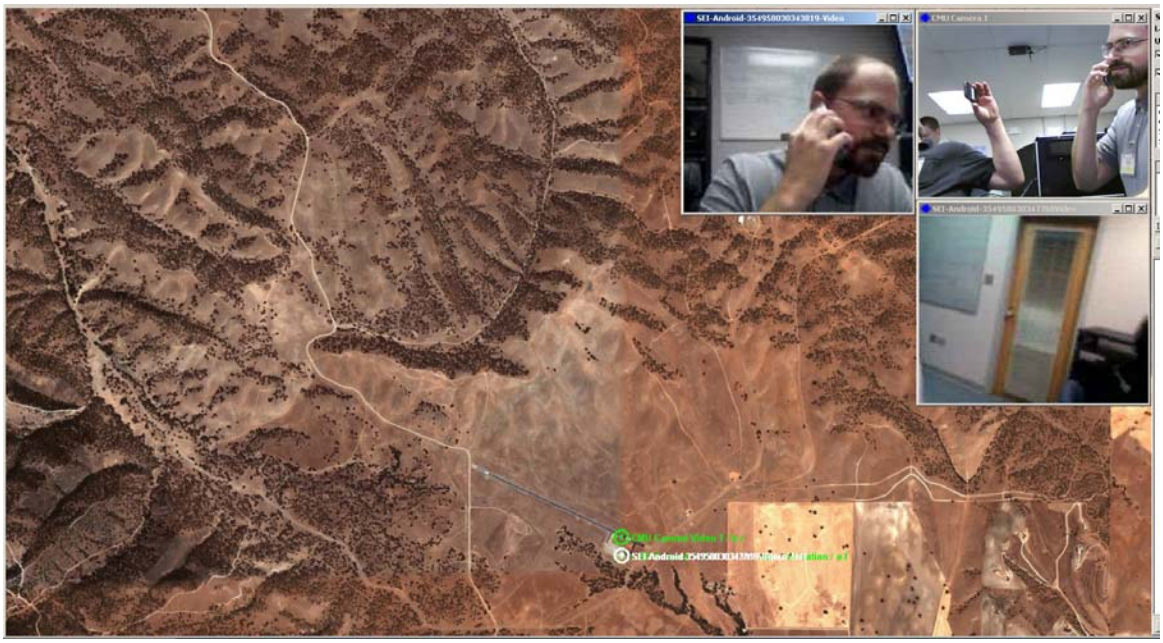


*Figure 8-7: VPN Images*

### 8.6.3 Analysis

By any subjective measure, the Tactical SOA IRAD prototypes were a success. At the May 2010 Tactical Network Topology (TNT) exercise the prototype received excellent feedback from active duty military personnel, who told team members that it was among the best capabilities demonstrated at the event. These personnel also helped us develop a list of features that would make the smartphone even more desirable. This success was repeated at the August 2010 TNT, where the general feeling was that our work was ahead of that of others in developing situational awareness capabilities for smartphones. Subsequently, we have been approached about porting our work to a proprietary cellular network for field trial.

In another subjective measure, we and those to whom we demonstrated the capability were impressed with the power and performance of the Nexus One smartphone. These and similar handhelds are becoming general-purpose computing devices in terms of performance and sophistication of operating systems and libraries. However, several non-software problems remain to be resolved. Two key problems are screen visibility in bright light and susceptibility to radio interference (while using the smartphone on a Wi-Fi network). These are not new problems to the military, and we expect them to be addressed as the basic technology is adapted to military use. We conclude that the strategy of adapting commercial handheld technology to tactical military problems is viable, particularly since the capabilities of these devices are improving at a rapid pace

driven by competition among major industry players such as Google, Apple, Research in Motion, and Nokia.

We also conclude that, at least over Wi-Fi and probably over high bandwidth cellular networks such as 3G and 4G, the use of SOA is viable up to a certain scale. This is despite very significant overhead in parsing SOAP messages discussed in subsequent paragraphs. There are limitations to the use of SOA, however, particularly for relatively low bandwidth networks and—as the number of video streams being sent to a handheld device increases—even in high bandwidth environments.

More objective measures of performance demonstrate the performance improvement associated with using UDP rather than TCP (see Figure 8-2) and the overhead associated with employing SOAP messaging for our web service (Figure 8-8). In Figure 8-8, the blue area represents the percentage of time required to decrypt a message containing an encoded video image on the Nexus One smartphone, red represents the time to parse the SOAP/XML message, and green represents the time to create the bitmap for rendering on the phone's screen. Clearly, the SOAP/XML parsing time dominates the other times, requiring more than eight times the decryption time, and more than five times the bitmap creation times. We have not determined the source of these excessive parsing times, but clearly they will be a problem in some network environments. We intend to investigate alternatives such as public domain binary formats (e.g., Protocol Buffers, Thrift) and the Efficient XML Interchange (EXI) format that was designed to improve performance and reduce bandwidth requirements.

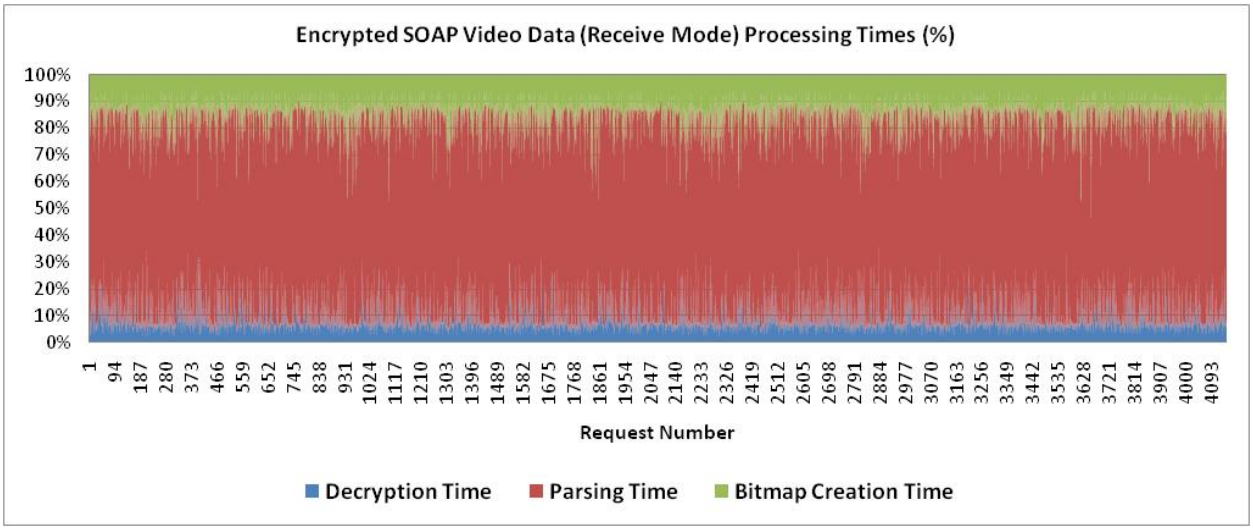



*Figure 8-8: Encrypted SOAP Video Data Processing Times*

Figure 8-8 also indicates that our security strategy employing Advanced Encryption Standard (AES) 256 bit encryption/decryption was handled comfortably by the Nexus One smartphone. Encryption/decryption was accelerated by use of native code for the Android platform and easily able to keep up with the rate of incoming messages. We expect future Android devices to support hardware-based encryption/decryption, which will further improve performance.

Smartphone-to-smartphone messaging through an intermediary was also surprisingly fast, at least with non-video data and video data from only a few feeds. Again, we expect that more video

feeds will ultimately overwhelm the smartphone due to the overhead associated with SOAP/XML parsing.

Our experiments and field tests identified several other engineering issues to consider in future work. These include

- a potential need to reconsider the way CoT data is distributed in order to improve performance by supporting "on demand" messaging
- implementation of a reliability layer on top of UDP to address issues that arise when packets carrying text messages are lost
- ability to split messages across packets to support higher resolution images that will exceed the roughly 64K packet size limit in UDP
- limitations and incompatibilities of SOAP implementations for smartphones and Windows computers

In summary, this work has led us to conclude that a subset of SOA (using SOAP) is practical in tactical environments. In particular, the performance of video feeds transmitting SOAP messages containing CoT data is visually comparable to the same video feed to a standard Windows desktop machine. However, to make the SOA strategy work, we made some non-typical (for SOA) choices including employing UDP rather than the customary TCP to improve video performance, using SOAP rather than REST, and employing a custom security strategy rather than WS-Security. Ultimately, the use of a SOA can potentially reduce development time by supporting service reuse, but only if mature SOA/SOAP implementations are available for the target platforms—not currently the case with kSOAP for Android and gSOAP for Windows.

## 8.7 Publications and Presentations

D. Plakosh; S. Simanta; E. Morris; W. Anderson; & J. Seibel wrote "Web Services for Ad Hoc and Resource-Impoverished Environments" for the Networking and Electronic Commerce Research Conference 2010 (NAEC 2010), held at Riva del Garda, Italy, October 7-10, 2010.

### 8.7.1 Bibliography

Dandashi, Fatma; Higginson, Jeffrey; Hughes, James; Narvaez, Wilson; Sabbouh, Marwan; Semy, Salim; & Yost, Beth. *Tactical Edge Characterization Framework Volume 1: Common Vocabulary for Tactical Environments* (MTR070331). MITRE Corp., 2007.

Dandashi, Fatma; Griggs, Aaron; Higginson, Jeffrey; Hughes, James; Narvaez, Wilson; Sabbouh, Marwan; Semy, Salim; & Yost, Beth. *Tactical Edge Characterization Framework—Vol. 2: Design Patterns for Tactical Environments* (MTR070326). MITRE Corp., 2007.

NATO/RTO Task Group IST-061. *Secure Service Oriented Architectures (SOA) Supporting NEC (Network Enabled Capability)* (RTO-TR-IST-061). http://ftp.rta.nato.int/public/PubFullText/RTO/TR/RTO-TR-IST-061/$$TR-IST-061-ALL.pdf

### 8.7.2 References


**[Ahmed 2009]**
Ahmed, Asaad; Yasumoto, Keiichi; Shibata, Naoki; Kitani, Tomoya; & Ito, Minoru. "DAR: Distributed Adaptive Service Replication for MANETs," 91-97. *Proceedings of 2009 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications.* Marrakech, Morocco, October 2009. IEEE Computer Society, 2009.

**[Anderson 2009]**
Anderson, Sharon. "CANES: Consolidated, Dynamic and Combat-Ready." *Chips 27*, 3 (July-September 2009): 6-8. Space and Naval Warfare Systems Center Atlantic, 2009. www.public.navy.mil/usff/chips/Documents/PDFs/chipsJul09.pdf

**[CES2TE 2008]**
CES2TE Focus Team. *Report on Core Enterprise Services to the Tactical Edge (CES2TE)*. 2008.

**[Dandashi 2007]**
Dandashi, Fatma; Higginson, Jeffrey; Hughes, James; Narvaez, Wilson; Sabbouh, Marwan; Semy, Salim; & Yost, Beth. *Tactical Edge Characterization Framework—Vol. 1: Common Vocabulary for Tactical Environments*. MITRE Corp., 2007. www.mitre.org/work/tech_papers/tech_papers_08/08_0037/08_0037.pdf

**[DARPA 2010a]**
Defense Advanced Research Projects Agency. *Advanced Soldier Sensor Information System and Technology (ASSIST).* http://www.darpa.mil/i2o/programs/assist/assist_tigr.asp

**[DARPA 2010b]**
Defense Advanced Research Projects Agency. *DARPA Transformative Apps: Broad Agency Announcement* (DARPA-BAA-10-41, Amended). April 16, 2010. www.darpa.mil/i2o/solicit/baa/DARPA_TA_BAA-10-41_Mod3.pdf

**[Fielding 2000]**
Fielding, Roy Thomas. "Architectural Styles and the Design of Network-Based Software Architectures." PhD diss., University of California, Irvine, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

**[Hafsoe 2007]**
Hafsoe, T., et al. "Adapting Web Services for Limited Bandwidth Tactical Environments." *12th International Command and Control Research and Technology Symposium (ICCRTS): Coalition Command and Control in the Networked Era*. Newport, RI, June 2007.

**[Halonen 2006]**
Halonen, Tommi & Ojala, Timo. "Cross-Layer Design for Providing Service Oriented Architecture in a Mobile Ad Hoc Network," Article 11. *Proceedings of the 5th International Conference on Mobile and Ubiquitous Multimedia.* Stanford, California, December 2006. Association for Computing Machinery, 2006.

**[Huang 2007]**
Huang, Orton & McGarry, Stephen. *Performance of Some Service-Oriented Architecture-Based Systems in the Airborne Network Environment.* IEEE, 2007.

**[Jammes 2007]**
Jammes, F.; Mensch, A.; & Smit, H. "Service-Oriented Device Communications Using the Devices Profile for Web Services," 947-955. *Proceedings of 21st International Conference on Advanced information Networking and Applications Workshops—Vol. 1.* Niagara Falls, Ontario, Canada, May 2007. IEEE Computer Society, 2007.

**[Natchetoi 2008]**
Natchetoi, Y.; Kaufman, V.; & Shapiro, A. "Service-Oriented Architecture for Mobile Applications," 27-32. *Proceedings of the 1st international Workshop on Software Architectures and Mobility.* Leipzig, Germany, May 2008. Association for Computing Machinery, 2008.

**[Niranjan 2009]**
Suri, Niranjan. "Dynamic Service-Oriented Architectures for Tactical Edge Networks," 3-10. *Proceedings of the 4th Workshop on Emerging Web Service Technology, International Conference Proceedings Series, Vol. 404*. Eindhoven, Netherlands, November 2009. Association for Computing Machinery, 2009.

**[OASIS 2009]**
OASIS. *Devices Profile for Web Services (DPWS)*, 2009.
http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01

**[Pruter 2008]**
Pruter, Steffen; Moritz, Guido; Zeeb, Elmar; Salomon, Ralf; Golatowski, Frank; & Timmermann, Dirk. "Applicability of Web Service Technologies to Reach Real Time Capabilities," 229-233. *Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing.* Orlando, Florida, May 2008. IEEE Computer Society, 2008.

**[Robbins 2007]**
Robbins, Doug. *Unmanned Aircraft Operational Integration Using MITRE's Cursor on Target,* Mitre Corporation, 2007. www.mitre.org/news/the_edge/summer_07/robbins.html

**[Sheu 2007]**
Sheu, Ray-Yuan; Czajkowski, Michael; & Hofmann, Martin. *Adaptive Peer-to-Peer Agent Sensor Networks.* Lockheed Martin Advanced Technology Laboratories, 2007.

**[Sliwa 2009]**
Sliwa, Joanna & Amanowicz, Marek. "A Mediation Service for Web Service Provision in Tactical Disadvantaged Environments." *Proceedings of Military Communications Conference (MILCOM) 2008.* San Diego, California, November 2008. IEEE, 2009.

**[Srirama 2007]**
Srirama, S. N.; Jarke, M.; & Prinz, W. "Mobile Web Services Mediation Framework," 6-11. *Proceedings of the 2nd Workshop on Middleware for Service Oriented Computing.* Newport Beach, CA, November 2007. Association for Computing Machinery, 2007.

**[Trifa 2010]**
Trifa, Vlad. "Web of Things —Toward Open and Sharable Networks of Embedded Devices." Presentation, Developer Conference. Zurich, Switzerland, April 2010. www.slideshare.net/vladounet/web-of-things-towards-open-and-sharable-networks-of-embedded-devices

**[W3C 2010]**
World Wide Web Consortium. *Web of Services*, 2010. www.w3.org/standards/webofservices/

**[Woyke 2009]**
Woyke, Elizabeth. "Raytheon Sends Android To Battlefield," *Forbes.com* (October 2009). www.forbes.com/2009/10/19/android-google-military-technology-wireless-raytheon.html

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| (Leave Blank) | February 2011 | Final |

| 4. TITLE AND SUBTITLE | 5. FUNDING NUMBERS |
|---|---|
| Results of SEI Independent Research and Development Projects | FA8721-05-C-0003 |

6. AUTHOR(S)

William Anderson, Archie Andrews, Nanette Brown, Cory Cohen, Christopher Craig, Tim Daly, Dionisio de Niz, Andres Diaz-Pace, Peter Feiler, David Fisher, David Gluch, Jeffrey Hansen, Jörgen Hansson, John Hudak, Karthik Lakshmanan, Richard Linger, Howard Lipson, Gabriel Moreno, Ed Morris, Onur Mutlu, Robert Nord, Ipek Ozkaya, Dan Plakosh, Mark Pleszkoch, Ragunathan (Raj) Rajkumar, Joe Seibel, Soumya Simanta, Charles Weinstock, and Lutz Wrage

| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) | 8. PERFORMING ORGANIZATION REPORT NUMBER |
|---|---|
| Software Engineering Institute<br>Carnegie Mellon University<br>Pittsburgh, PA 15213 | CMU/SEI-2011-TR-002 |

| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
|---|---|
| HQ ESC/XPK<br>5 Eglin Street<br>Hanscom AFB, MA 01731-2116 | ESC-TR-2011-002 |

11. SUPPLEMENTARY NOTES

| 12A DISTRIBUTION/AVAILABILITY STATEMENT | 12B DISTRIBUTION CODE |
|---|---|
| Unclassified/Unlimited, DTIC, NTIS | |

13. ABSTRACT (MAXIMUM 200 WORDS)

The Software Engineering Institute (SEI) annually undertakes several independent research and development (IRAD) projects. These projects serve to (1) support feasibility studies investigating whether further work by the SEI would be of potential benefit and (2) support further exploratory work to determine whether there is sufficient value in eventually funding the feasibility study work as an SEI initiative. Projects are chosen based on their potential to mature and/or transition software engineering practices, develop information that will help in deciding whether further work is worth funding, and set new directions for SEI work. This report describes the IRAD projects that were conducted during fiscal year 2010 (October 2009 through September 2010).

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| Security Architecture, Software Requirements, Software Architecture, Agile, Multi-core, Cyber-Physical Systems | 98 |

16. PRICE CODE

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |