Software Engineering Institute

Results of SEI Independent Research and Development Projects

Christopher J. Alberts, Bill Anderson, Len Bass, Matthew Bass, Philip Boxer, Lisa Brownsword, Sagar Chaki, Peter H. Feiler, Dave Fisher, Eileen C. Forrester, Suzanne M. Garcia, Aaron Greenhouse, Jorgen Hansson, Jim Herbsleb, James Ivers, Peter Lee, Richard C. Linger, Thomas A. Longstaff, Pratyusa K. Manadhata, B. Craig Meyers, D. Michael Phillips, Carol A. Sledge, James D. Smith II, Kurt Wallnau, Gwendolyn H. Walton, Jeannette Wing, Noam Zeilberger

July 2007

TECHNICAL REPORT
CMU/SEI-2007-TR-006
ESC-TR-2007-006

SEI Director's Office

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.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Abstract

Each year, the Software Engineering Institute (SEI) 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 2006 (October 2005 through September 2006).

1 Introduction

This document briefly describes the results of the independent research and development projects conducted at the Carnegie Mellon® Software Engineering Institute (SEI) during the 2005–06 fiscal year.

1.1 PURPOSE OF THE SEI INDEPENDENT RESEARCH AND DEVELOPMENT PROGRAM

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.

At a DoD meeting in November 2001, the SEI's DoD sponsor approved a set of thrust areas and challenge problems to provide long-range guidance for the SEI research and development program, including its IRAD program. The thrust areas are survivability/security, interoperability, sustainability, software R&D, metrics for acquisition, acquisition management, and commercial off-the-shelf products. The IRAD projects conducted in FY 2006 were based on these thrust areas and challenge problems.

® ` Carnegie Mellon is registered in the U. S. Patent and Trademark Office by Carnegie Mellon University.

1.2 OVERVIEW OF IRAD PROJECTS

The following research projects were undertaken in FY 2006:

- Using Structural and Dynamic Modeling to Support Acquisition, Development, and Sustainable Deployment of Software-Intensive Systems (Bill Anderson, Lisa Brownsword, Dave Fisher, Suzanne M. Garcia, Philip Boxer)
- Understanding Organizational Risk in Architectural Design (Len Bass, Jim Herbsleb, Matthew Bass)
- Certified Binaries for Software Components (Sagar Chaki, James Ivers, Peter Lee, Kurt Wallnau, Noam Zeilberger)
- Technology for Managing Data and Data Quality in Distributed Embedded Real-Time Systems (Jorgen Hansson, Peter H. Feiler, Aaron Greenhouse)
- Technology Foundations for Computational Evaluation of Software Security Attributes (Gwendolyn H. Walton, Thomas A. Longstaff, Richard C. Linger)
- Toward Interoperable Acquisition: The Example of Risk Management (B. Craig Meyers, Christopher J. Alberts, Eileen C. Forrester, Suzanne M. Garcia, D. Michael Phillips, Carol A. Sledge, James D. Smith II)
- An Attack Surface Metric (Pratyusa K. Manadhata, Jeannette Wing)

These projects are described in detail in this technical report.

2 Using Structural and Dynamic Modeling to Support Acquisition, Development, and Sustainable Deployment of Software-Intensive Systems

Bill Anderson, Lisa Brownsword, Dave Fisher, Suzanne M. Garcia,
Philip Boxer

2.1 PURPOSE

Complex modeling techniques such as system dynamics and discrete event modeling have been used in economics, urban planning, and other specialty areas for decades. In the software development arena, simpler structural modeling techniques such as data flow diagrams and structure/function charts have been the prevalent modeling approaches, while techniques like system dynamics and agent-based simulations have had much lower adoption rates. In this IRAD, we surveyed the use of different modeling techniques in software and systems engineering and then selected a few of these—systems thinking, system dynamics, and complex structural modeling—to explore in more depth. We were particularly looking for techniques that will provide novel technical insight into the integration and evolution of systems of systems. We looked at these three approaches from the viewpoint of their anticipated value and transitionability (the factors that create enablers or barriers to their adoption) to the software, systems, and system-of-systems engineering communities.

2.2 BACKGROUND

From the viewpoint of our IRAD, we defined a model as any means of characterizing something that abstracts in some way from reality and simulation as any process that determines the implications of a model through automated interpretation of the model. Thus a model can be viewed as a theory of how a system behaves, while a simulation is a means to determine the implications of that theory for a particular set of initial conditions.

We presumed that systems of systems represent a level of complexity not historically dealt with and that

- Modern systems are or will become systems of systems.
- Specialization (due to its independent development of specialized system components) continues to be a driver of the need for systems of systems.

If modeling and simulation have contributed to understanding similar challenges in other domains and if the cost of doing the modeling and simulation is less than the value derived, we could conclude that use of modeling and simulation is a viable strategy.

2.3 APPROACHES

Our overall approach included

- conducting literature reviews of relevant research
- using multiple modes of interaction (interviews, class instruction, conference attendance, and client engagement) to obtain the requisite skills and knowledge for evaluating the approaches
- recommending a subset of techniques for further feasibility and transitionability analysis and trial application
- directly participating in and observing a significant client engagement
- synthesizing and publishing the results of our research for community review
- recommending next steps for the SEI in relation to research in this area

2.4 COLLABORATIONS

Within the SEI, the project team consisted of Bill Anderson, Lisa Brownsword, David Fisher, and Suzanne Garcia. We also received valuable inputs from Andrew Moore and Dawn Cappelli of CERT.

Outside the SEI, Philip Boxer of Boxer Research Limited co-authored one of our reports and provided invaluable insights into complex Structural Modeling. Drs. Guenther Ruhe and Dietmer Phafl of the University of Calgary hosted our team for a series of meetings and also provided invaluable inputs and peer review. All of these individuals contributed their time without SEI compensation.

2.5 EVALUATION CRITERIA

The a priori success criteria for judging the results of this IRAD were

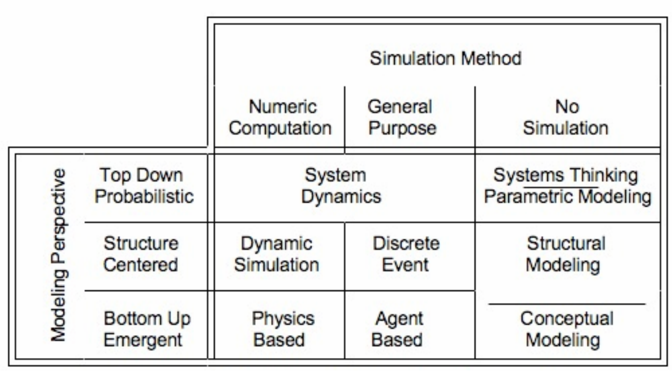
- One or more approaches to structural or dynamic modeling have been identified that have evidence of successful application within a complex, software-intensive system application.
  - We found systems-thinking approaches applicable to many different contexts and observed the use of both system dynamics modeling being used successfully within the SEI and complex structural modeling being used within a case study context.
- Barriers and enablers to the transitionability of structural and dynamic modeling techniques have been identified and characterized.
  - See our results section and the transitionability technical note for details.
- At least one trial participant has been identified to apply one or more of the promising techniques identified by the study.
  - We were able to trial complex structural modeling in one setting and observe results of the use of system dynamic modeling in CERT via interviews with Andy Moore and Dawn Capelli.
  - Three additional potential trial participants have been identified.

We believe we have exceeded the defined evaluation criteria.

2.6 RESULTS

2.6.1 Comparison of Perspectives

Modeling and simulation approaches can be categorized along two major dimensions: (1) the perspective from which the system is being modeled and (2) the simulation method being used. The allocation of nine major modeling and simulation methods along these two dimensions is shown in Figure 2-1.

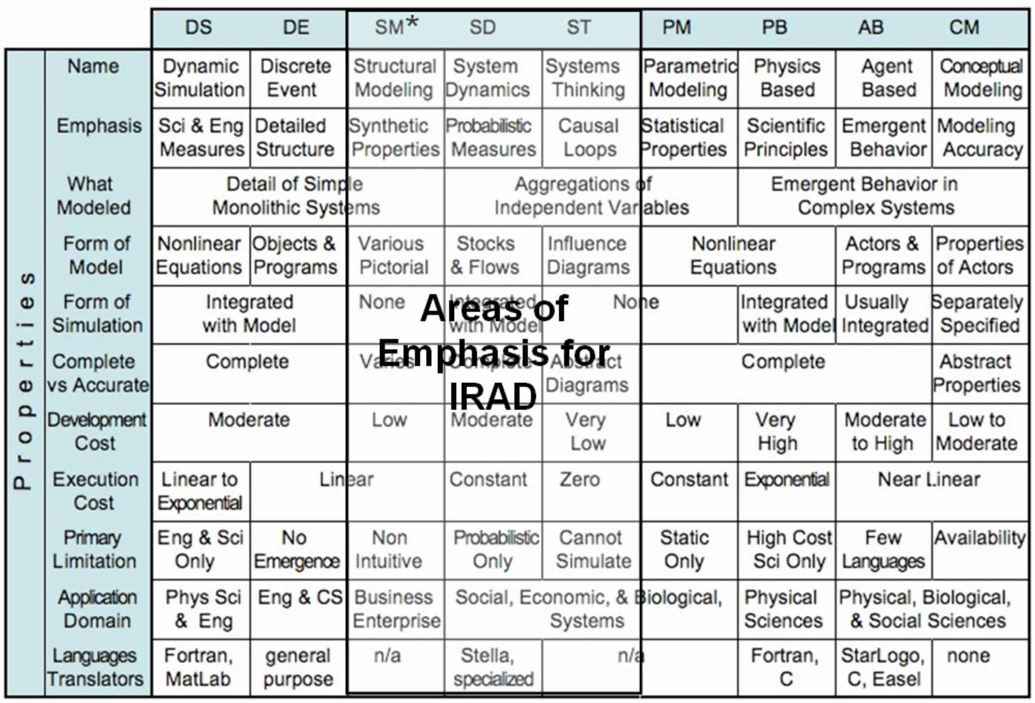


		Simulation Method		
		Numeric Computation	General Purpose	No Simulation
Modeling Perspective	Top Down Probabilistic	System Dynamics		Systems Thinking Parametric Modeling
	Structure Centered	Dynamic Simulation	Discrete Event	Structural Modeling
	Bottom Up Emergent	Physics Based	Agent Based	Conceptual Modeling

Figure 2-1: Comparison of Modeling and Simulation Approaches

2.6.2 Comparison of Properties

To facilitate tradeoff decisions among modeling and simulation approaches, we offer a comparison matrix (Figure 2-2) that compares modeling and simulation approaches according to the properties highlighted in the rows.



		DS	DE	SM*	SD	ST	PM	PB	AB	CM
Properties	Name	Dynamic Simulation	Discrete Event	Structural Modeling	System Dynamics	Systems Thinking	Parametric Modeling	Physics Based	Agent Based	Conceptual Modeling
	Emphasis	Sci & Eng Measures	Detailed Structure	Synthetic Properties	Probabilistic Measures	Causal Loops	Statistical Properties	Scientific Principles	Emergent Behavior	Modeling Accuracy
	What Modeled	Detail of Simple Monolithic Systems			Aggregations of Independent Variables			Emergent Behavior in Complex Systems		
	Form of Model	Nonlinear Equations	Objects & Programs	Various Pictorial	Stocks & Flows	Influence Diagrams	Nonlinear Equations		Actors & Programs	Properties of Actors
	Form of Simulation	Integrated with Model		None	Integrated with Model	None		Integrated with Model	Usually Integrated	Separately Specified
	Complete vs Accurate	Complete		Varies	Complete	Abstract Diagrams	Complete			Abstract Properties
	Development Cost	Moderate		Low	Moderate	Very Low	Low	Very High	Moderate to High	Low to Moderate
	Execution Cost	Linear to Exponential	Linear		Constant	Zero	Constant	Exponential	Near Linear	
	Primary Limitation	Eng & Sci Only	No Emergence	Non Intuitive	Probabilistic Only	Cannot Simulate	Static Only	High Cost Sci Only	Few Languages	Availability
	Application Domain	Phys Sci & Eng	Eng & CS	Business Enterprise	Social, Economic, & Biological, Systems			Physical Sciences	Physical, Biological, & Social Sciences	
	Languages Translators	Fortran, MatLab	general purpose	n/a	Stella, specialized	n/a		Fortran, C	StarLogo, C, Easel	none

Figure 2-2: Property-Based Comparisons of Approaches[1]

2.6.3 Comparison Conclusions

Modeling and simulation are important tools for a broad spectrum of applications and purposes. There are also many approaches to modeling and simulation. The appropriate choice for a given purpose depends on the kind of phenomena being studied and the traditions of the community doing the modeling. The choice should also be influenced by whether simulation is required, the degree of accuracy needed, cost of development and execution, and availability of languages and other support tools. In practice, the split between communities with traditions in continuous mathematical and those with the discrete methods of computer science is paramount, unfortunately to the detriment of both communities. Physics-based and agent-based approaches, for example, can both be used to address the same class of scientific and engineering problems but with differing methods and cost profiles.

There is also potential value in the understanding and insight that several abstract modeling approaches provide without need for simulation. Abstract modeling approaches can be used alone or in combination with executable models and may offer a more transitionable alternative in software-intensive system development settings.

[1] See the technical note *Transitionability of Complex Modeling and Simulation Approaches to Software-Intensive Systems Development* [Anderson 2006] for details on analyses that we performed with these tables.

* Structural Modeling is a general classification and Complex Structural Modeling is a specific example of the class, with some blurring of the boundaries implied in this matrix.

2.6.4 Transition Challenges

We addressed the transitionability of three modeling and simulation approaches: (1) systems thinking, (2) system dynamics modeling, and (3) complex structural modeling.

We define transitionability as the degree to which the characteristics of the technology and its whole product support adoption by the intended user population at the intended level of use. We used the following categories for organizing transition challenges:

- **Inherent adoptability**— Relative Advantage, Complexity, Observability, and Trialability
- **Implementation adoptability** —Modularity, Implementation Complexity, and Technology Drag
- **Adopter characteristics**—Innovators, Early Adopters, Early Majority, Late Majority, and Laggards
- **Transition mechanisms**—includes factors related to the existence/deployment of transition mechanisms related to the Commitment Adoption curve used by the SEI

See the *CMMI Survival Guide* [Garcia 2006] for descriptions of these heuristics.

2.6.4.1 Summary of Transitionability Analysis

The following table summarizes the transitionability of three selected techniques. Following the table are some general statements about the transitionability of complex modeling techniques as a whole.

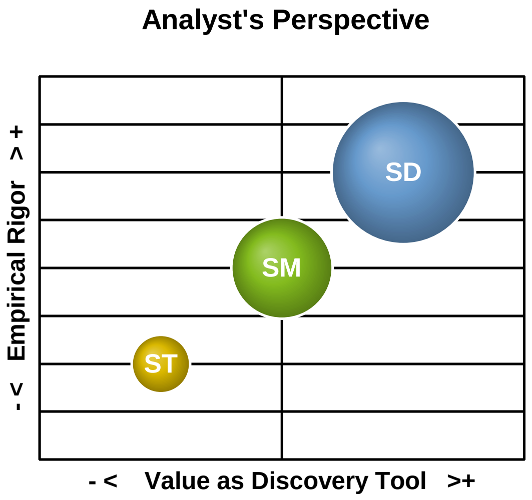
Table 2-1: Transitionability of Modeling and Simulation Approaches Examined

Approach	Inherent Adoptability	Implementation Adoptability	Adopter Characteristics	Transition Mechanisms
System Thinking (ST)	C – Low (+) T – Medium O – High (+) R – High (+)	M – Medium I – Medium TD – Low (+)	• Early Majority (modelers) • Late Majority (users led through process)	• Books • Commercial tool market • Academic and industry training
System Dynamics (SD)	C – High (-) T – Low (-) O – Medium R – Medium	M – Low (-) I – High (-) TD - N/A Steep learning curve	• Early Adopter (modelers) • Early Adopter (users led by)	• Domain focused • Books • Robust commercial tool market • Academic and industry training
Complex Structural Modeling (SM) (as exemplified by PAN)	C – High (-) T – High (+) O – Medium R – Medium	M – Medium I – High (-) TD – Medium	• Innovators (modelers) • Early Adopter (users led by)	• Technical papers • Workshops/consultants • Mix of commercial and proprietary tools used
Key to Inherent Adoptability and Implementation Adoptability Codes C= Complexity; T = Trialability; O= Observability; R = Relative Advantage; M = Modularity; I = Implementation Complexity; TD = Technology Drag; (+) = a positive value for transitionability; (-) = a negative value for transitionability				

2.6.5 Value Considerations

The use of models and simulations as decision support and learning tools provides direct tangible benefit to the *user* community (e.g., risk identification for project managers, complexity analysis for architects, and decision support for design decision makers). The cost/benefit comparisons and learning curves associated with a user community will be different from those required of communities that wish to actively build models and simulations and use them as discovery mechanisms. This second community is broadly labeled *analysts* for our purposes. Analysts use these techniques to develop (the models and simulations), discover (unique attributes of the investigation subject), and teach (the discoveries and how to use the models as user tools). Users leverage the models and simulations provided by the analysts to learn and make decisions about the subject that has been modeled. The participation by both communities in the modeling and simulation (M&S) elicitation also has significant value as a mechanism to generate deeper, shared understanding of the modeled subject.

We compared the techniques on several value scales from both of these perspectives. While they are difficult to quantify in absolute terms, the selected techniques can be compared relatively, as shown in Figure 2-3 and Figure 2-4. The IRAD SR [Anderson 2007a] includes detailed descriptions of these comparisons.



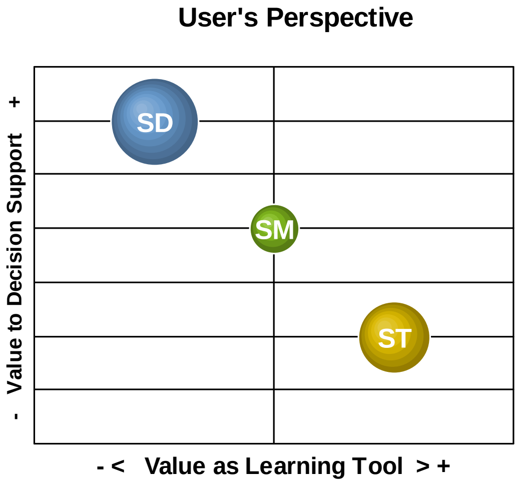


Figure 2-3: Analyst's Perspective on Value

Figure 2-4: User's Perspective on Value

A third perspective is that of software engineering practitioners. Each of the focus techniques advocates system-level analysis and comprehension. In today's era of specialization, we believe this system perspective is increasingly diminished at the risk of overall system quality. The abstraction techniques taught by each of these methodologies generate unique appreciation of and perspectives on complexity, dynamics, and holistic thinking that would benefit every software engineer. To the extent that software engineering is linked with systems engineering, valuable lessons and skills could be developed in the technical staff through general education in the philosophy and techniques advocated in this body of knowledge.

2.6.5.1 Areas of Benefit

We suggest three broad areas within the SEI that could benefit from modeling and simulation: (1) process, (2) education, and (3) systems of systems, or more generally, large complex systems.

- *Process:* We recommend that the CMMI® framework include more elaborations that support complex modeling. M&S should be leveraged to increase awareness (learning) in non-Software Engineering Measurement and Analysis courses (most SEMA courses are already leveraging M&S). Also, we recommend adding more awareness of M&S benefits to appraisal training, especially for high maturity clients.
- *Education:* We recommend leveraging the CERT System Dynamics and System Thinking expertise and experience to develop SEI staff training in these philosophies and techniques. We believe the SEI could position itself to offer outreach to undergraduate programs and could directly influence graduate curricula such as the Master of Software Engineering program at CMU.
- *Systems of Systems:* The Integration of Software-Intensive Systems (ISIS) Initiative is actively exploring M&S through its application of Complex Structural Modeling on the NATO risk probe engagement and Dr. Fisher's student mentoring of agent-based modeling at California State University at Pomona. Initial steps also have been taken with the University of Calgary to explore Visiting Scientist potential during Dr. Ruhe's upcoming sabbatical. These are initial steps toward building SEI capabilities to improve SEI skill levels in modeling the complexity being seen in systems of systems. CERT's insider threat M&S efforts demonstrate M&S value when analyzing complex behavior patterns.

2.6.5.2 Problem Solving Tools

The value entry point is in providing novel insights into complex problems. M&S is a means to this end and should not be presented (at least initially) as an end product. As we gain expertise and case examples, productized M&S applications are envisioned (e.g., training simulators for users to explore complex system behavior or decision support models to assist portfolio management). We believe there is an opportunity for the SEI to become a leader in the creative application of M&S in the software engineering domain, potentially creating opportunities as our expertise and reputation grows.

2.7 PUBLICATIONS AND PRESENTATIONS

[Anderson 2007a]

Anderson, W.; Brownsword, L.; Fisher, D.; & Garcia, S. *Transitionability of Complex Modeling and Simulation Approaches to Software-Intensive Systems Development.* Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, special report forthcoming.

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Anderson, W.; Boxer, P.; & Brownsword, L. *An Examination of a Structural Modeling Risk Probe Technique.* Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, special report forthcoming.

Anderson, W.; Brownsword, L.; Fisher, D.; & Garcia, S. *Using Structural and Dynamic Modeling to Support Acquisition, Development, and Sustainable Deployment of Software Intensive Systems*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, out-brief presentation.

2.8 BIBLIOGRAPHY

[Adams 2005]
Adams, R. J. & Black, L. "System Dynamics Modeling: Integrating Acquisition, Program, and Technical Management." *Proceedings of the 17th Annual Systems & Software Technology Conference (SSTC 2005).* Salt Lake City, UT (USA), April 18–21, 2005.
http://www.sstc-online.org/Proceedings/2005/CDHome.htm

[Alberts 2003]
Alberts, D. S. & Hayes, R. E. *Power to the Edge: Command and Control in the Information Age*. Washington, DC: CCRP Publication Series, 2003.

[Andersson 2002]
Andersson, C.; Karlsson, L.; Nedstam, J.; Host, M.; & Nilsson, B. I. "Understanding Software Processes Through System Dynamics Simulation: A Case Study," 41–48. *Proceedings Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*. Lund, Sweden, April 8–11, 2002. Los Alamitos, CA, USA: IEEE, 2002 (ISBN 0-7695-1549-5).

[Barros 2002]
Barros, M. D. O.; Werner, C. M. L.; & Travassos, G. H. "A System Dynamics Metamodel for Software Process Modeling." *Software Process Improvement and Practice 7*, 3–4 (Sept.-Dec. 2002): 161–172 (ISSN 1077-4866).

[Bérard 2005]
Bérard, C.; Cloutier, L. M.; & Cassivi, L. *Performance Evaluation of Management Information Systems in Clinical Trials: A Systems Dynamics Approach*.
http://www.albany.edu/cpr/sds/conf2005/proceed/papers/CLOUT410.pdf (2005).

[Burke 1997]
Burke, S. *Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization* (CMU/SEI-96-TR-024, ADA327609). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996.
http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.024.html

[Caulfield 1991]
Caulfield, C. W. & Maj, S. P. "A Case for Systems Thinking and System Dynamics," 2793–2798. *2001 IEEE International Conference on Systems, Man and Cybernetics: e-Systems and e-Man for Cybernetics in Cyberspace.* Tucson, AZ (USA), October 7–10, 2001.
http://ieeexplore.ieee.org/iel5/7658/20951/00971932.pdf

[CPSMA 2000]
Commission on Physical Sciences, Mathematics, and Applications. *A Transition Strategy for Enhancing Operational Capabilities*. Washington, DC (USA): The National Academies Press, 2000. http://books.nap.edu/books/0309069254/html

[Forrester 1991]
Forrester, J. W. "System Dynamics and the Lessons of 35 Years." *The Systemic Basis of Policy Making in the 1990s*. Boston, MA (USA): Sloan School of Management, MIT, 1991.

[Garcia 2004]
Garcia, S.; Roberts, J.; & Estrin, L. "Managed Technology Adoption Risk: A Way to Realize Better Return from COTS Investments," 74–83. *Proceedings of the Third International Conference on COTS-Based Software Systems (ICCBSS 04).* Redondo Beach, CA (USA), January 2–4, 2004. Heidelberg, Germany: Springer-Verlag, 2004.

[Garcia 2006]
Garcia, S. M. & Turner, R. *CMMI Survival Guide: Just Enough Process Improvement.* Boston, MA (USA): Addison-Wesley, 2006.

[Gray 2005]
Gray, D. E. & Gibson, D. "Déjà Vu, Again and Again." *Proceedings of the 2005 SEPG (SEPG 2005)*. Seattle, WA (USA), March 7–10, 2005. Pittsburgh, PA (USA): Software Engineering Institute, Carnegie Mellon University, 2005.

[Greene 2005]
Greene, H. & Mendoza, R. "Lessons Learned from Developing the ABCS 6.4 Solution." *Defense AR Journal* (April-July 2005). http://www.dau.mil/pubs/arq/2005arq/arq2005.asp

[Grizzle 2002]
Grizzle, G. A. & Pettijohn, C. "Implementing Performance-Based Program Budgeting: A System-Dynamics Perspective." *Public Administration Review 62*, 1 (January-February 2002), 51–62.

[Guo 2001]
Guo, H.; Liu, L.; Huang, G. H.; Fuller, G. A.; Zou, R. & Yin, Y. Y. "A System Dynamics Approach for Regional Environmental Planning and Management: A Study For The Lake." *Journal of Environmental Management 61*, 1 (January 2001): 93–111.

[Häeberlein 2003]
Häeberlein, T. *A Framework for System Dynamic Models of Software Acquisition Projects.* http://prosim.pdx.edu/prosim2003/paper/prosim03_haeberlein.pdf (2003).

[Homer 2000]
Homer, J.; Ritchie-Dunham, J.; Rabbino, H.; Puente, L. M.; Jorgensen, J.; & Hendricks, K. "Toward a Dynamic Theory of Antibiotic Resistance." *System Dynamics Review 16*, 4 (Winter 2000): 287–319.

[Jackson 2001]
Jackson, M. C. **"**Critical Systems Thinking and Practice." *European Journal of Operational Research 128*, 2 (16 January. 2001): 233–244 (ISSN 0377-2217).

[Karnopp 2000]
Karnopp, D. C.; Margolis, D. L.; & Rosenberg, R. C. *System Dynamics: Modeling and Simulation of Mechatronic Systems*. New York, NY (USA): John Wiley & Sons, 2000.

[Lyneis 1999]
Lyneis, J. M. “System Dynamics for Business Strategy: A Phased Approach.” *System Dynamics Review 15*, 1 (Spring 1999): 37–70.

[Madachy 1996]
Madachy, R. J. “System Dynamics Modeling of an Inspection-Based Process,” 376–386. *Proceedings of the 18th International Conference on Software Engineering*. Berlin, Germany, March 25-29, 1996. Washington, DC (USA): IEEE Computer Society, 1996 (ISBN 0-8186-7246-3).

[McGrath 2001]
McGrath, G. M. & Campbell, B. “Implementing Recommendations as a Result of a System Dynamics Intervention.” *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS-34).* Maui, HI (USA), January 3–6, 2001.
http://www.hicss.hawaii.edu/HICSS_34/PDFs/DTABS02.pdf

[Morris 2004]
Morris, E.; Levine, L.; Meyers, C.; Place, P.; & Plakosh, D. *Systems of Systems Interoperability* (CMU/SEI-2004-TR-004, ADA455619). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004.
http://www.sei.cmu.edu/publications/documents/04.reports/04tr004.html

[Pfahl 2005]
Pfahl, D.; Ruhe, G.; Lebsanft, K.; & Stupperich, M. “Software Process Simulation with System Dynamics—A Tool for Learning and Decision Support,” 57–90. *New Trends in Software Process Modelling*. Edited by S. T Acuña and M. I. Sánchez-Segura. *Vol. 18, Series on Software Engineering and Knowledge Engineering*. Singapore: World Scientific, 2006 (ISBN 981-256-619-8).

[Richmond 2004]
Richmond, B. *An Introduction to Systems Thinking*. Lebanon, NH (USA): isee systems, 2004.

[Senge 1990]
Senge, P. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York, NY (USA): Doubleday, 1990.

[Senge 1994]
Senge, P.; Roberts, C.; Ross, R.; Smith, B.; & Kleiner, A. *The Fifth Discipline Fieldbook: Strategies and Tools for Building a Learning Organization*. New York, NY (USA): Doubleday, 1994.

[Simonovic 2003]
Simonovic, S. P. “Assessment of Water Resources Through System Dynamics Simulation: From Global Issues to Regional Solutions,” 93b. *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36).* Maui, HI (USA), January 6–9, 2003.
http://csdl2.computer.org/comp/proceedings/hicss/2003/1874/03/187430093b.pdf

[Skartveit 2003]
Skartveit, H.-L.; Goodnow, K.; & Viste, M. “Visualized System Dynamics Models as Information and Planning Tools,” 1113–1128. *Proceedings of the Informing Science + Information Technol-*

ogy Education Joint Conference (IS 2003). Pori, Finland, June 24–27, 2003. http://proceedings.informingscience.org/IS2003Proceedings/docs/140Skart.pdf

[Stacey 1992]
Stacey, R. *Managing the Unknowable: Strategic Boundaries between Order and Chaos.* San Francisco, CA (USA): Jossey-Bass, 1992.

[Stallinger 2001]
Stallinger, F. & Grunbacher, P. **“**System Dynamics Modelling and Simulation of Collaborative Requirements Engineering.” *Journal of Systems and Software 59*, 3 (15 December 2001): 311 – 321 (ISSN: 0164-1212).

[Stave 2002]
Stave, K. A. “Using System Dynamics to Improve Public Participation in Environmental Decisions.” *System Dynamics Review 18*, 2 (Summer 2002): 139–167.

[Sterman 2000]
Sterman, J. D. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. New York, NY (USA): Irwin/McGraw-Hill, 2000.

[Sweeney 2000]
Sweeney, L. “*Bathtub Dynamics: Initial Results of a Systems Thinking Inventory*.” *System Dynamics Review 16*, 4 (Winter 2000): 249–286.

[VanDerHejden 2005]
Van Der Hejden, K. *Scenarios: The Art of Strategic Conversation.* West Sussex, UK: John Wiley & Sons, 2005.

[Weaver 2000]
Weaver, W. *Science and Complexity*. http://www.ceptualinstitute.com/genre/weaver/weaver-1947b.htm (2000).

[Weinberg 1991]
Weinberg, G. *Quality Software Management Volume 1: Systems Thinking*. New York, NY (USA): Dorset House Publishing, 1991.

3 Understanding Organizational Risk in Architectural Design

Len Bass, Jim Herbsleb, Matthew Bass

3.1 PURPOSE

The purpose of the IRAD was twofold: first to establish that there is a source of risk in the misalignment between organization structure and processes and the architectural decisions that have been made. Secondly, the purpose was to identify and verify low-cost indicators that can be used to identify the occurrences of this misalignment.

3.2 BACKGROUND

It is well known that architectures (both software and system) become deeply "embedded" in the organizations that design and build software-intensive systems (e.g., [Conway 1968]). Architectures determine key characteristics of organizations, such as work assignments, and have implications for communication patterns, habitual ways that people select and filter information, and organizational problem-solving strategies. Changing the architecture, even in seemingly simple ways, can cause serious misalignments between the architecture and the organization, leading in some cases to complete failure of the firm [Henderson 1990]. Organizations are notoriously difficult to change, and we do not yet have any tools for understanding the kinds of changes we are imposing on them when we perform architectural design. We don't know how to assess the risks, nor do we know how to address the risks once they are identified.

Suppose, for example, that the designer of an architecture who was considering two alternative designs knew that design 1 requires the developers of a particular component to access expertise from a separate organization requiring extensive long distance communication, and design 2 requires the same developers to access expertise from a co-located site. Everything else being equal, design 2 is preferable, since it will reduce organizational risk by lowering the communication overhead required to solve any problem that arises. Organization structure, existing communication patterns, location of people and resources, shared work history, and potentially other factors not yet identified, will influence the risk, time required, component quality, and development efficiency of the design and construction of software components. Organizational factors should be considered during the making of design decisions, just as performance or reliability considerations are considered.

It is also well known that a system's software architecture is a central artifact in the development of software-intensive systems. It is a primary determinant of the system's quality attributes. It is the bridge between business goals served by a system and its implementation. The SEI has invested considerable effort in understanding the principles of the design and analysis of software architecture. However, people must effectively collaborate to realize the benefits of software architecture. Conversely, organizational inhibitors to collaboration can be inimical to realizing the potential benefits of software architecture. Therefore, understanding the relationship between types of organizations and types of architecture is vital to realizing the benefits of software architecture.

The purpose of this IRAD was to investigate the relation between organizational characteristics and software architecture. If two people are working together on the same component, they must communicate about all aspects of the component. If they are working on different components, they must communicate about the ways in which their respective components interact. The two situations may require different types and bandwidths of communication. Depending on the fashion in which two components interact, there may be a requirement for high or low bandwidth communication between the people. The communication bandwidth among people is affected by many different elements, such as co-location, organizational structure, technological support, and knowledge of where different types of expertise reside in an organization. In the modern world, most large systems are constructed across multiple sites, usually involving multiple distinct organizations. Understanding the organizational characteristics that are essential to effectively building a system with a particular architecture will provide fundamental knowledge that can be exploited to more effectively manage and design large software systems.

If the relation between organizational characteristics and software architecture were well understood, it would have application in architecture evaluation and architecture design and in assessing the readiness of an organization for architecture-centric development. During an architecture evaluation, discovering a misalignment between architectural decisions and organizational structure is discovering a risk to the development effort. During an architecture design, discovering a misalignment between proposed architectural decisions and proposed organizational structure allows for the correction of the misalignment or at least for allowing for it in the project management. There might be a tradeoff, for example, between optimizing the performance of a system and optimizing it for the organizational structure of the developing organizations. One question that the Software Architecture Technology initiative is frequently asked is “How ready is organization X for architecture-based development?” Being able to analyze the structure of organization X with respect to common architecture usages is one aspect of being able to answer this question. A fundamental understanding of the relation between organizational characteristics and software architectures could allow us to go beyond assessing readiness in general and determine the range of architectures an existing organization could build.

A complicating element of the understanding of organizational characteristics is that they evolve over time, and desirable characteristics may change, depending on the current stage of the project development. During the initial stages of a project, one might see, for example, particular patterns of communication among the groups working on the infrastructure. During later stages one might see communication between application developers and infrastructure developers.

During this IRAD project, we proceeded on both quantitative and qualitative paths. The quantitative path was based on the Siemens Global Studio project, which exists to provide a test bed for studying global development practices. The qualitative path was based on particular Siemens and other companies projects that were known to the members of the project. The Siemens Global Studio project is heavily instrumented, and through identification of “significant events” involving coordination issues, we were able to isolate the types of data that provide indication of the occurrence of these events. The particular projects we investigated exhibited different types of misalignments. One project had a requirement for extensive coordination over shared resources, but the organizational structure did not support this coordination. The second project had a requirement for extensive coordination over state management to support high availability, but, again, the organizational structure did not support this coordination. The third project had metrics in place for one of the development organizations that led to design decisions that were locally but not globally optimum.

3.3 APPROACH

The fundamental premise of the IRAD is new. That is, that software architecture decisions and organizational structures and processes interact in various fashions. Indeed, one of the major results of the IRAD was to discover instances where major projects were seriously impacted, if not caused to fail, by this interaction.

The next step in the approach is to prepare a catalog of various types of interactions between organization structures and processes and architecture decisions that are problematic. This catalog will enable several activities:

- Coupling the catalog with low-cost monitoring activities, managers will have early indication of the onset of a problem caused by the misalignment of organization structure and processes and the software architecture. This will enable the adjustment of either the processes or the architecture to reduce the severity of the problem.
- The catalog will provide another tool to architecture evaluators. Evaluators will be able to examine the architecture and the organization structure to determine potential risks from misalignment.
- Architecture designers will have another tool to support their design process. They will be able to either match their architecture design to the organizational constraints or inform the management that a misalignment is possible.

3.4 COLLABORATIONS

The SEI participant in this study was Len Bass. Collaborators included Professor James Herbsleb of the School of Computer Science at Carnegie Mellon and one of his graduate students. An additional collaborator was Matthew Bass of Siemens Corporate Research. Matthew Bass’s participation was funded by Siemens.

3.5 EVALUATION CRITERIA

One result was the demonstration that there is a problem. The examples were subjected to peer review through submission to a respected professional conference [Bass 2007].

A second result of the project involves the collection of data to enable the determination of the early indicators of misalignment. The process of collecting data was subjected to peer review

[Mullick 2006], and the results will be subjected to the normal evaluation of empirical results, as well as peer review.

3.6 RESULTS

We have two classes of results. The first class is qualitative in nature and they demonstrate that, in fact, misalignment is a serious problem. The second class is quantitative and they are intended to identify early indicators for misalignment. We have not yet finished the analysis of the quantitative data, but we will describe the data we have been collecting in detail.

3.6.1 Qualitative Results

We describe three systems and their misalignment problems.

System 1

System 1 was an embedded information system with limited memory. The architecture decision was made for multiple processes to share this memory in a dynamic fashion. The organizational decision was that the development of the different process was done by teams in two different countries with no history of working together. The sharing of memory was not identified as a potential risk for the development.

The system failed repeatedly during execution. The failure was traced to the sharing of memory. As a result, a major customer for the system was lost.

Sharing a resource is an activity that requires intensive coordination, yet there was no recognition either by management or by the development teams that special coordination was needed to overcome the distributed development aspects of the system.

System 2

System 2 was a control system for transportation vehicles. The transportation system had multiple vehicles being controlled and multiple control stations. There was a requirement for high availability for the overall system. Development teams were located in Western Europe, Eastern Europe, and the United States. The various teams had no history of working together.

The initial architecture for the system is shown in Figure 3-1. Each local system had its own cache for sensitive data. The concept was that the local data could be consolidated into an overall state necessary for recovery in the event of a failure. The problem was that different aspects of the different caches were distributed. The development teams had no provision for special coordination over the design of the different caches. On the other hand, the basic design had been used successfully by the business unit (although not using these particular development teams).

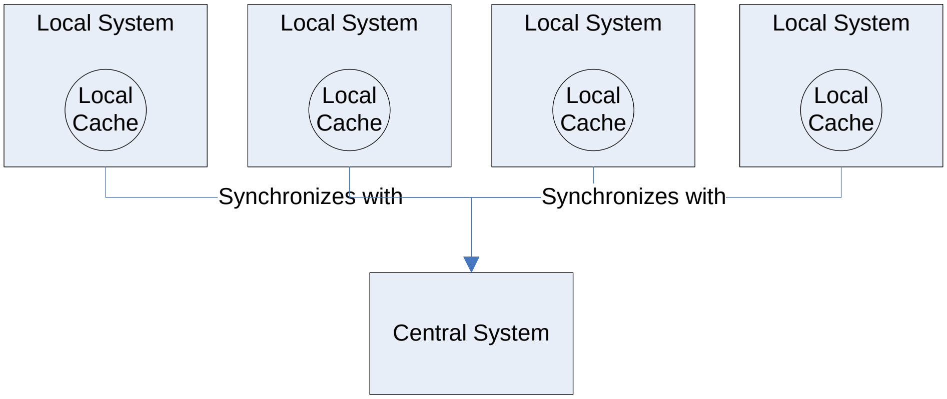


Figure 3-1 Initial System Architecture

The development effort floundered because the assembling of an overall state from the local stations became too difficult. The system was redesigned to have an overall repository in the central system with local copies of the repository in the local systems.

In this example, we see that the delegating of the same type of decisions to distributed teams resulted in inconsistent decisions being made with no special means for coordinating those decisions.

Neither the delegating of the state management decisions to the different development teams nor the distribution of the development teams was by itself incorrect. The basic design had been used previously, and distributed development had occurred previously. It was the combination of the two that caused a problem.

System 3

System 3 is an embedded system. The system was developed as layers. Layer 1 was developed in Europe and layer 2 in Asia. The organization in Asia was measured by three metrics:

1. delivery time
2. customer satisfaction (end customer)
3. defects (in layer 2)

A dispute developed between the developers of Layer 1 and the developers of Layer 2 over the type of interface between the two layers. The developers of Layer 2 favored an interface style that funneled many of the calls through one main application programmer interface. The developers of Layer 1 favored an interface style that had separate application programmer interfaces for different functions.

The virtue of the style favored by the developers of Layer 2 was that it made testing Layer 2 easier, since there were fewer interfaces to test. The virtue of the style favored by the developers of Layer 1 is that it made integration testing easier, since each function was separately identified through the two layers.

The developers of Layer 2 ended the dispute by stating that they had to develop immediately in order to meet their development schedule. This decision caused great difficulty in the integration process.

This is not a coordination problem, as in the prior two examples, but a problem of misplaced metrics. The customer for the developers of Layer 2 should have been identified as the integrators of the system, since they were the recipients of the development. The count of defects should not have been limited to Layer 2 but should have included defects discovered during integration. This is, however, an example of the misalignment between organizational processes (in this case, the metrics) and architectural decisions (the interface style between the two layers).

3.6.2 Quantitative Results

The quantitative results come from the Siemens Global Studio project, which was described in [Mullick 2006]. Since this project was designed as a test bed, a great deal of data is being collected about the development. The information being collected is

- from the Media Wiki (the basic coordination mechanism)
  - who visited what artifact and when
  - who edited what artifact and when
  - who replied to posts by whom
  - what were they talking about (page discussions)
- from Mailman (the e-mail server)
  - who replied to posts by whom
  - what were they talking about
- Social Network Survey
  - who’s talking to whom
  - purpose of conversation and mechanism used
- Additional Survey
  - conformance of code to architecture
  - effectiveness of process
- from Subversion (the configuration management system) – who was working on common files with whom
- From Mantis (issues/bugs) – amount/severity/cost of issues on a particular component
- During code reviews – issue tracking
- meeting minutes – issue tracking
- interviews with architect
  - what issues arose
  - what was stability of interfaces (for architectural dependency analysis)
- From the architecture description – what “parts” are dependent upon what other parts
- From the plan data – who was working on what “parts” and *when*

We have identified a number of "important events" that, potentially, represent examples of misalignments. Our plan for each important event is to perform the following analysis:

- Build a timeline of events.
- Consolidate data on coordination activities related to the event.
- Understand misalignment between coordination activities and technical decisions.
- Examine data collected to see what pattern of data might have identified the problem.
- Test data to determine whether a similar pattern exists elsewhere in the data set and whether it also indicated a problem.

For those events that upon examination actually represent misalignments, we will have identified early indicators of these events from the data collected.

3.7 SUMMARY OF RESULTS

We have shown through our qualitative examples that the misalignment problem is a serious one deserving of attention. Through the quantitative examples we are identifying early indicators. Through both types of examples, we are preparing a catalog of misalignments that satisfies the needs identified in the justification for this IRAD.

3.8 REFERENCES

[Bass 2007]
Bass, M.; Mikulovic, V.; Bass, L.; Herbsleb, J.; & Cataldo, M. "Architectural Misalignment: An Experience Report." *Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA'07)*. New York, NY: IEEE Computer Society Press, 2007 (0-7695-2744-2/07).

[Conway 1968]
Conway, M. E. "How Do Committees Invent?" *Datamation 14,* 4 (1968): 28-31.

[Henderson 1990]
Henderson, R. M. & Clark, K. B. "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms." *Administrative Science Quarterly 35,* 1 (1990): 9-30.

[Mullick 2006]
Mullick, N.; Bass, M.; El Houda, Z.; Paulish, D. J.; Cataldo, M.; Herbsleb, J. D.; Sangwan, R.; & Bass, L. "Siemens Global Studio Project: Experiences Adopting an Integrated GSD Infrastructure." *Proceedings of the International Conference on Global Software Engineering*. Brazil, Oct. 2006. New York, NY: IEEE Computer Society Press, 2006.

4 Certified Binaries for Software Components

Sagar Chaki, James Ivers, Peter Lee, Kurt Wallnau, Noam Zeilberger

4.1 PURPOSE

There is an evident need for mechanisms that enhance our ability to *trust* third-party software. In the current era of plug-and-play, off-the-shelf programs are being increasingly made available as modules or *components* that can be attached to an existing infrastructure. More often than not, such plug-ins are distributed in machine code or *binary* form. In this report we present a framework that can be used to generate trustworthy binaries for software components and to prove that binaries generated elsewhere satisfy specific policies. At the core of our method lies a paradigm called proof-carrying-code (PCC), originally proposed in a seminal paper by Necula and Lee [Necula 1996, Necula 1997]. The essential idea underlying PCC is to construct a proof of the claim that a piece of machine code respects a desired *policy*. The proof is shipped along with the code so that it may be independently verified before the code is deployed.

To date, the application of PCC has been restricted to pure *safety* policies. The progress of PCC has also been hindered by, among other things, the need for manual intervention (e.g., discovering complicated loop invariants) and large proof sizes. Our approach overcomes these limitations of PCC in the context of certifying software components. In particular, we achieve the following three objectives: (1) **Enrich:** Expand the set of PCC policies to include both safety and *liveness*. To this end, we use a state/event-based temporal logic called SE-LTL developed in the context of the Predictable Assembly from Certifiable Components (PACC)[2] project at the SEI. (2) **Automate:** Use iterative refinement in combination with predicate abstraction and model checking to generate appropriate invariants and ranking functions required for certificate and proof construction in a completely *automated* manner. (3) **Compact:** Use state-of-the-art Boolean satisfiability (SAT) technology to generate extremely small proofs. Preliminary investigations [Chaki 2006a] indicate that the use of SAT yields proof sizes that are several orders of magnitude more compact than when using conventional methods.

4.2 BACKGROUND

In the original formulation of PCC, the world is divided into trusted code consumers and untrusted code producers. A code consumer publishes a safety policy. In general, safety policies assert that "something bad never happens," while liveness policies assert "something good will eventually happen." The code producer annotates the code with key invariants and uses a *certifying compiler* to generate object code as well as a verification condition (VC); in essence, the VC is the logical formula that is valid if and only if the object code respects the safety policy. The

[2] http://www.sei.cmu.edu/pacc

certifying compiler also constructs a proof of the VC, which is embedded in the object code; hence “proof carrying.” The code consumer checks that the proof is valid by verifying its construction against a set of sound axioms and inference rules that have been defined on the machine instructions themselves. The verification step is efficient and reduces to a form of type checking. In other words, the proof is valid if, and only if, it is well typed.

PCC does not depend on the correctness of the certifying compiler or on the technologies used to construct proofs of program properties. PCC is also resistant to tampering, including code optimizations. Most attempts at modifying either the object code or the proof of the VC will lead to an ill-typed proof and hence will be detected. Moreover, any undetected tampering is guaranteed to result in code that still respects the published safety policy and hence is harmless as far as the policy is concerned. Last, proof-carrying code is efficient, since the static proof eliminates the need for runtime checks. Still, a number of technical challenges (discussed by Necula et al. [Necula 1996]) arose in this original formulation of PCC. Particularly notable among these are

- **(CH1)** The restriction to safety conditions is problematic if the cost of developing a trustworthy PCC infrastructure is great.
- **(CH2)** The proof generator sometimes requires manual assistance (to compute loop invariants, for example); for practical transition purposes, this is a non-starter.
- **(CH3)** The proofs generated are often quite large, hindering wider use of the PCC paradigm. Despite many recent advances, this continues to be an open problem.

Prior to this work, we conducted two projects that have a direct bearing on the above challenges. First, as part of the PACC project, we developed an expressive linear temporal logic called SE-LTL that can be used to express both safety and liveness claims of component-based software. In this work, we adopt and modify SE-LTL to express certifiable policies, thereby targeting **CH1**. Second, in collaboration with Prof. Peter Lee, an original proponent of and leading expert in PCC, we conducted an Independent Research and Development (IRAD) project [Chaki 2005b] on “Assessing and Demonstrating the Readiness of Proof Carrying Code for Obtaining Objective Trust in Software Components.” As part of this PCC-IRAD, we developed an infrastructure to generate compact certificates for *C programs* (not binaries) against SE-LTL claims in an automated manner. The automation is achieved by combining iterative refinement with predicate abstraction and model checking to generate appropriate invariants and ranking functions that are required for certificate and proof construction. The tightness of proofs is obtained via the use of state-of-the-art Boolean satisfiability (SAT) technology [Chaki 2006a]. In this work, we extend this framework to certify *binaries* generated from component specifications. Thus, we complete the framework from a PCC perspective and also address issues **CH2** and **CH3**. To this end we build on the PACC infrastructure for analyzing specifications of software component assemblies and generating deployable machine code for such assemblies.

4.2.1 Related Work

PCC was proposed by Necula and Lee [Necula 1996, Necula 1997, Necula 1998b] for certifying memory safety policies on binaries. PCC works by hard-coding the desired safety policies within the machine instruction semantics, while our approach works at the source code level and encodes the policy as a separate specification state machine. Foundational PCC [Appel 2001, Hamid 2002] attempts to reduce the trusted computing base of PCC to include only the foundations of mathematical logic. Bernard and Lee [Bernard 2002] propose a new temporal logic to express PCC policies for machine code. Non-SAT-based techniques for minimizing PCC proof sizes [Necula 1998a, Necula 2001] have also been proposed. Certifying model checkers [Kupferman 2004,Namjoshi 2001] emit an independently checkable certificate of correctness when a temporal logic formula is found satisfiable by a *finite state* model. Namjoshi [Namjoshi 2003] has proposed a two-step technique for obtaining proofs of Mu-Calculus specifications on *infinite state* systems. In the first step, a proof is obtained via certifying model checking. In the second step, the proof is *lifted* through an abstraction. Namjoshi's approach is still restricted to certifying source code, while our work aims for low-level binaries. Iterative refinement has been applied successfully by several software model checkers such as SLAM [Ball 2001], BLAST [Henzinger 2002b] and MAGIC [Chaki 2004]. While SLAM and MAGIC do not generate any proof certificates, BLAST implements a method [Henzinger 2002a] for lifting proofs of correctness. However, BLAST's certification is limited to source code and purely safety properties. Assurance about the correctness of binaries can also be achieved by proving the correctness of compilers (which is impracticable) or via translation validation [Pnueli 1998] (which still assumes that the source code is correct). In contrast, our approach requires no such correctness assumptions.

4.3 APPROACH

Our technical approach is best summarized by the architecture described in Figure 4-1. This figure depicts the final infrastructure for certified component binary generation that we developed. The boxes are numbered for ease of reference. The steps involved in generating certified component binaries can be summarized as follows:

Step 1. We begin (Box 1) with a specification *Spec* of a component assembly written in the Construction and Composition Language (CCL) [Wallnau 2003], which has been developed as part of the PACC project. The specification *Spec* contains a description of the component assembly as well as the desired policy *Pol* that the assembly needs to be certified against. CCL is currently implemented as a profile of an executable subset of UML 2.0.

Step 2. *Spec* is automatically *interpreted* [Ivers 2002] into a program, *Prog*, that can be processed by a model checker. *Prog* (Box 2) essentially comprises a C program *Prog* along with finite state machine specifications for library routines invoked by the program. This step was implemented by augmenting the interpretation for the ComFoRT [Ivers 2004] reasoning framework so that *Prog* contains additional information. Specifically, *Prog* contains information that relates its line numbers, variables, and other data structures with those of *Spec*. This information is crucial for the subsequent *reverse-interpretation* of certificates in Step 4.

Step 3. *Prog* is input to *Copper* (Box 3), a state-of-the-art *certifying software model checker*. Copper [Chaki 2005a] was originally developed as part of ComFoRT. To perform this step, Copper was enhanced [Chaki 2006a] with the ability to generate certificates. Copper interfaces with theorem provers (TP) and SAT solvers (SAT) during model checking and certificate generation. The output of Copper is either a counterexample (*CE*) to the desired policy *Pol* or a certificate (*Cert1*) that *Prog* respects *Pol*. Specifically, *Cert1* is a ranking function that associates with each line number of *Prog* a set of pairs *(r,i)* where *r* is a rank and *i* is an invariant. In addition, *Cert1* can be used to generate a valid verification condition for proving that *Prog* respects *Pol*.

Step 4. The certificate *Cert1* certifies *Prog* only against the policy *Pol*. It is reverse-interpreted (arrow 4) into a certificate (*Cert2*) that *Spec* itself also respects *Pol*. The reverse interpretation is enabled by the additional information generated during interpretation to connect *Spec* with *Prog* (Step 2). The certificate *Cert2* is the counterpart of *Cert1* for *Spec*. Thus, it is a ranking function that associates sets of pairs of ranks and invariants with line numbers of *Spec*. It can also be used to generate a valid verification condition for proving that *Spec* respects the policy *Pol*.

Step 5. *Spec* and *Cert2* are now transformed (Box 5) into a certified Pin/C program *Code* (Box 6) that can be compiled and deployed in the Pin runtime environment (RTE) [Hissam 2005]. The uncertified version of this code generation process, which transforms a CCL specification to Pin/C code, had been developed as part of the PACC project. We enhanced the code generation so that it also creates a certificate, using *Cert2*, and embeds it in *Code*. In essence, we transform the certificate, along with the component, from one format (CCL) to another (Pin/C). In practice, this is achieved by embedding calls to special procedures inside *Code*. Recall that *Cert2* is a ranking function that associates each line number of *Spec* with a set of pairs of ranks and invariants. In practice, a rank is just an integer, while an invariant is a C expression. Suppose that *Cert2* associates line *X* of *Spec* with the set $\{(r_1,i_1), \ldots,(r_n,i_n)\}$. Then, just before generating the code for line *X*, we add the following two function calls:

```
BEGIN(); INV((rank == r1 && i1) || … || (rank == rn && in));
```

In the above, `BEGIN` and `INV` are distinguished procedures, while `rank` is a special global variable introduced in the generated code for the purpose of certification.

Step 6. The final step (Box 7) consists of three distinct stages and results in the final product: the certified binary for *Spec*. The first compilation step is conceptually the same as Step 5 but transforms the certified program from Pin/C to binary code (Box 8). In our implementation we use the gcc compiler (which targets the PowerPC instruction set) to achieve this goal. The end result is a certified binary for the component assembly. Part of the certificate is a ranking function that is embedded directly in the binary as calls to `BEGIN` and `INV` and the evaluation of the arguments to these procedures. Recall that these arguments are the ranks and invariants specified by *Cert2*.

The remaining portion of the certificate consists of the proof of a verification condition (*VC*) and is constructed by the remaining two stages of Step 6. The first of these two stages constructs *VC* using the binary and weakest precondition computation. In essence, *VC* is a logical formula that codifies the following two facts: **(Fact 1)** the ranking function embedded in the binary (as calls to `BEGIN` and `INV`) is well formed, and **(Fact 2)** the ranking function implies that the binary respects the policy *Pol*. Once *VC* is constructed, it is proved using a proof-generating automated theorem prover. For our implementation we use a theorem prover based on Boolean satisfiability (SAT). In essence we convert the logical negation of *VC* to a Boolean formula *F*. We then check if *F* is unsatisfiable using the proof-generating SAT solver ZChaff. If this is the case (and hence if *VC* is valid), the resolution proof emitted by ZChaff serves as the proof of the validity of *VC*. The use of SAT enables us to obtain extremely compact proofs [Chaki 2006a] in practice.

Certificate Validation. As mentioned before, one of the key aspects of our approach is that the certified binary (specifically, the certificate itself) can be independently, and objectively, validated without the need to trust most of the artifacts (shown in Figure 4-1) that are involved in the generation of the certified binary. In fact, this process occurs in two stages:

1. First, the verification condition *VC* is reconstructed from the certified binary. This is done in exactly the same manner as during certification (specifically Step 6 described above). Note that the procedure for constructing *VC* is publicly known and is also deterministic. Thus, the *VC* constructed during certificate validation should be exactly the same as the one construction during certification.
2. Finally, we verify that the proof *Proof* associated with the certified binary corresponds to the *VC* that was obtained in the first step. To do this, we again convert the logical negation of *VC* to a Boolean formula *F* and check that *Proof* is a resolution proof of the unsatisfiability of *F*. Once again, the procedure for constructing *F* from *VC* is public and deterministic, ensuring that the two versions of *F* obtained during certification and validation are exactly the same. If *Proof* is indeed found to refute *F*, we declare the certificate to be valid. Otherwise, the certificate is declared invalid.

Trusted Computing Base (TCB). It is instructive to discuss the artifacts that must be trusted in order for our approach to be effective. In essence, the TCB comprises the following: (i) the procedure for constructing the verification condition *VC*, (ii) the procedure for converting the negation of *VC* to a Boolean formula *F,* and (iii) the procedure for checking that the resolution proof associated with the certified binary actually refutes *F.* All of these procedures are computationally inexpensive and can be implemented by relatively small programs. Thus, they are much more trustworthy (and more amenable to some form of formal verification or validation) than the rest of the components of Figure 4-1. Specifically, these components, which are no longer required to be in the TCB, are the interpreter, the certifying model checker, the reverse-interpreter, the code generator, the compiler, and the theorem prover. All of these tools are considerably more complex, and their elimination from the TCB raises considerably the degree of confidence associated with our certification methodology. In fact, a very rough measurement indicates that in our own case, the TCB is over fifteen times smaller in size than the rest of the infrastructure.

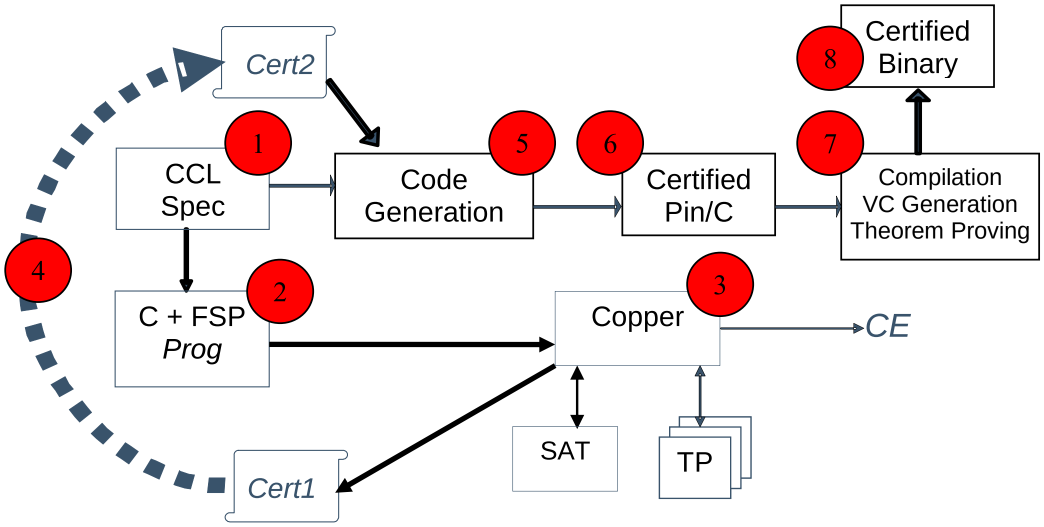


Figure 4-1: Architecture of Developed Framework

4.4 COLLABORATIONS

The IRAD project was carried out as a collaborative effort between the SEI and Carnegie Mellon University. Specifically, the participants from the SEI were Sagar Chaki, James Ivers, and Kurt Wallnau, all members of the Product Line Systems Program. The participants from Carnegie Mellon were Professor Peter Lee, one of the original proponents of PCC, and his student Noam Zeilberger, both of whom are affiliated with the Computer Science Department.

4.5 EVALUATION CRITERIA

- A prototypical infrastructure for automatically generating certified binaries of software components from UML Statechart specifications.
- An improved SEI understanding of the issues related to software trust and component certification documented in a technical report and/or peer-reviewed publications.
- Improved collaborative work between (i) the SEI and the Carnegie Mellon Computer Science Department and (ii) SEI programs and initiatives.
- Improved visibility of the SEI in the world of software and computer science research.

4.6 RESULTS

We implemented a prototype of our technology and experimented with two kinds of examples. First, we created a CCL specification of a component that manipulates an integer variable and the specification that the variable never becomes negative. Our tool was able to successfully prove, and certify at the assembly code level, that the implementation of the component does indeed satisfy the desired claim. The CCL file size was about 2.6 KB, while the generated Pin/C code was about 20 KB. In contrast, the assembly code was about 110 KB, while the proof certificate size was just 7.7 KB. The entire process took about 5 minutes with modest memory requirements.

In order to validate the translation of a certified C program to a certified binary (Step 7 in Figure 4-1), we experimented with Micro-C,[3] a lightweight operating system for embedded real-time applications. The OS source code consists of about 6000 lines of C (97 KB) and uses a semaphore to ensure mutually exclusive access to shared kernel data structures. Using our approach we were able to certify that all kernel routines follow the proper locking order when using the semaphore. In other words, the semaphore is always acquired and released alternately, and thus deadlock is avoided. The total certification time was about one minute, and the certificate size was about 11 KB, or roughly 11% of the operating system source code size. We also experimented with the C implementation of the "tar" program in the Plan 9[4] operating system. Specifically, we certified, using our approach, that a particular buffer will never overflow when the program is executed. The source code was manually annotated in order to generate the appropriate proof certificates. While our experiments show that our approach is viable, we believe that a more robust implementation and more realistic case studies are needed in order to push our technique amongst a wider user base.

A paper [Chaki 2006a] describing the SAT technology we used to obtain concise proof certificates was published and presented at the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) in March 2006. An article describing the adaptation of our IRAD research for certifying the absence of buffer overflows is available as an SEI technical note [Chaki 2006b]. Moreover, an invited talk on research carried out during this IRAD was delivered at the XOOTIC Symposium in September 2006. By enabling our collaboration with Prof. Peter Lee, an original proponent and internationally recognized stalwart in the area of PCC, this IRAD has fostered closer connections between the SEI and the Carnegie Mellon campus. It has also strengthened our knowledge and expertise in the theory and practice of software certification.

4.7 REFERENCES

[Appel 2001]
Appel, A. "Foundational Proof-Carrying Code," 247-256. *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*. June 16-19, 2001. New York, NY: IEEE Computer Society Press, 2001.

[Ball 2001]
Ball, T. & Rajamani, S. "Automatically Validating Temporal Safety Properties of Interfaces." *Proceedings of the 8th International SPIN Workshop on Model Checking of Software (SPIN'2001)*. Berlin, Germany: Springer, LNCS 2057, 2001.

[Bernard 2002]
Bernard, A. & Lee, P. "Temporal Logic for Proof-Carrying Code." *Proceedings of the Conference on Automated Deduction*. Copenhagen, Denmark, July 27-30, 2002.

[3] http://www.micrium.com

[4] http://plan9.bell-labs.com/plan9

[Chaki 2004]
Chaki, S.; Clarke, E.; Groce, A.; Jha, S.; & Veith, H. "Modular Verification of Software Components in C," 388–402. *IEEE Transactions on Software Engineering 30*, 6 (June 2004).

[Chaki 2005a]
Chaki, S.; Ivers, J.; Sharygina N.; & Wallnau, K. "The ComFoRT Reasoning Framework," 164-169. *Proceedings of the 17th Conference on Computer Aided Verification (CAV)*. Berlin, Germany: Springer, LNCS 3576, July 2005.

[Chaki 2005b]
Chaki, S. & Wallnau, K. *Proof-Carrying Code* (Chapter 6, CMU/SEI-2005-TR-020, ADA449433). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr020.html

[Chaki 2006a]
Chaki, S. "SAT-Based Software Certification." *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2006. Berlin, Germany: Springer, LNCS 3920, 2006.

[Chaki 2006b]
Chaki, S. & Hissam, S. *Certifying the Absence of Buffer Overflows* (CMU/SEI-2006-TN-030). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06tn030.html

[Hamid 2002]
Hamid, N.; Shao, Z.; Trifonov, V.; Monnier, S. & Ni, Z. "A Syntactic Approach to Foundational Proof-Carrying Code," 89-100. *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science*. July 22-25, 2002. New York, NY: IEEE Computer Society Press, 2002.

[Henzinger 2002a]
Henzinger, T.; Jhala, R.; Majumdar, R.; Necula, G.; Sutre, G.; & Weimer, W. "Temporal Safety Proofs for Systems Code." *Proceedings of the 14th Conference on Computer Aided Verification (CAV)*. Berlin, Germany: Springer, LNCS 2404, 2002.

[Henzinger 2002b]
Henzinger, T.; Jhala, R.; Majumdar, R.; & Sutre, G. "Lazy Abstraction," 58-70. *ACM SIGPLAN Notices, Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '02)*. Portland, January 16-18, 2002. New York, NY: ACM Press, 2002.

[Hissam 2005]
Hissam, S.; Ivers, J.; Plakosh, D.; & Wallnau, K. *Pin Component Technology (V1.0) and Its C Interface* (CMU/SEI-2005-TN-001, ADA441815). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tn001.html

[Ivers 2002]
Ivers, J.; Sinha, N.; & Wallnau, K. *A Basis for Composition Language CL* (CMU/SEI-2002-TN-026, ADA407797). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. http://www.sei.cmu.edu/publications/documents/02.reports/02tn026.html

[Ivers 2004]
Ivers, J. & Sharygina, N. *Overview of ComFoRT: A Model Checking Reasoning Framework* (CMU/SEI-2004-TN-018, ADA442864). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004.
http://www.sei.cmu.edu/publications/documents/04.reports/04tn018.html

[Kupferman 2004]
Kupferman, O. & Vardi, M. "From Complementation to Certification." *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2004. Berlin, Germany: Springer, LNCS 2988, 2004 (ISBN 978-3-540-21299-7).

[Namjoshi 2001]
Namjoshi, K. "Certifying Model Checkers." *Proceedings of the 13th Conference on Computer Aided Verification (CAV)*. Berlin, Germany: Springer, LNCS 2102, 2001.

[Namjoshi 2003]
Namjoshi, K. "Lifting Temporal Proofs through Abstractions," 174-188. *Proceedings of the 4th International Conference on Verification, Model Checking and Abstract Interpretation*, 2003. Berlin, Germany: Springer, LNCS 2575, 2003.

[Necula 1996]
Necula, G. & Lee, P. "Safe kernel extensions without runtime checking," 229-243. *Proceedings of the 2nd USENIX Symposium on Operating System Design and Implementation (OSDI)*. Seattle, Washington, 1996.

[Necula 1997]
Necula, G. "Proof-Carrying Code," 106-119. *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages 1997*. Paris, France, January 15-17, 1997. New York, NY: ACM Press, 1997.

[Necula 1998a]
Necula, G. & Lee, P. "Efficient Representation and Validation of Proofs," 93-104. *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*. June 21-24, 1998. New York, NY: IEEE Computer Society Press, 1998.

[Necula 1998b]
Necula, G. & Lee, P. "Safe, Untrusted Agents Using Proof-Carrying Code," 61-91. *Proceedings of Mobile Agents and Security*. Berlin, Germany: Springer, LNCS 1419, 1998 (ISBN 3-540-64792-9).

[Necula 2001]
Necula, G. & Rahul, S. "Oracle-Based Checking of Untrusted Software," 142-154. *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages 2001*. New York, NY: ACM Press, 2001.

[Pnueli 1998]
Pnueli, A.; Siegel, M.; & Singerman, E. "Translation Validation," 151-166. *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 1998. Berlin, Germany: Springer, LNCS 1394, 1998 (ISBN 3-540-64356-7).

[Schneider 2000]
Schneider, F. “Enforceable Security Properties.” *ACM Transactions on Information and System Security 3*, 1 (February 2000): 30-50.

[Wallnau 2003]
Wallnau, K. & Ivers, J. *Snapshot of CCL: A Language for Predictable Assembly* (CMU/SEI-2003-TN-025, ADA418453). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tn025.html

5 Technology for Managing Data and Data Quality in Distributed Embedded Real-Time Systems

Jorgen Hansson, Peter H. Feiler, Aaron Greenhouse

5.1 PURPOSE

It is a *fait accompli* that the amount of data being managed by embedded real-time systems has increased significantly, and it continues to increase. This presents new challenges on how to efficiently design a real-time system at a conceptual and architectural level, modeling data requirements, and how to implement the real-time system at a system level to enable efficient manipulation and storing of real-time data. In essence, we are advocating that data needs to be explicitly modeled early in the development process and that a data-centric framework provides a better platform for data-intensive real-time applications, in contrast to task-centric methodologies used today when designing real-time systems.

The purpose of this project has been to (i) investigate existing solutions for managing real-time data and their constraints, (ii) identify technical gaps and shortcomings of distributed real-time systems for enabling efficient management of data in these systems, and (iii) work toward a data modeling framework that can be used at design time of embedded real-time systems to ensure that data quality attributes are satisfied.

Although there are technology advances in managing real-time data, existing technology has not yet been exploited widely or on a large scale. We believe that the primary component missing is that the there is no framework and techniques for modeling data that can be used at design time of real-time systems. By effectively incorporating a framework for managing data effectively in real-time systems, there are significant gains when exploited in large scale across several systems as the development time decreases significantly due to software reuse. The software quality/correctness increases over time as a common and tested framework is being used. Interoperability, in this context referring to simplified exchange of data across systems, can also be improved dramatically.

We have focused on *embedded sensor networks*, as they are becoming increasingly important in net-centric warfare to perform dense sensing of physical environments and in distributed embedded systems. These two applications both represent distributed systems and cover the spectrum of soft to hard real-time performance constraints.

5.2 BACKGROUND

The number of data items that are used in real-time embedded systems has increased over the years. There are several reasons for this depending on the application domain. We observe that for a large class of systems, it is enabled by the availability of cost-effective but still more powerful CPUs and larger amounts of memory, and more advanced functions in the software. Also, the applications require that data is ensured to be of higher quality and/or more data is monitored and collected. An example of this is the increasing need for performing system diagnostics in ECUs (electronic control units) to meet emission regulations.

The emergence of embedded sensor networks (ESNs) introduces a new paradigm in distributed computing platforms, as they are fundamentally different from conventional distributed systems and ad hoc networks [Stankovic 2003]. From the perspective of data quality cognizance, a net-centric system is intrinsically a system of systems comprising a number of heterogeneous, geographically dispersed systems, that is, loosely coupled nodes interconnected by a communication network. Normally they operate under real-time constraints of varying stringency and criticality. ESNs are increasingly being deployed in applications to monitor, collect, process, and communicate data obtained from the environment (e.g., monitoring of weather conditions to improve accuracy in weather forecasting, water leakage systems in nuclear power plants, corrosion in building constructions by the use of smart paint, intrusion in restricted areas). The applications vary in their criticality and also rely on ESNs' ability to monitor and timely detect situations, where the consequences of an undetected situation may range from innocuous to cataclysmal. ESNs provide data services to high-level applications, and they are becoming an important constituent of many net-centric systems. In ESNs the main objective is to minimize communication due to sparse energy resources and dynamically changing topology of networks and to satisfy constraints on timely data dissemination within the network [Akyildiz 2002, Stankovic 2003].

Thus, we can see that there is a significant need for efficient real-time data services, where the application data normally span from low-level control data, typically acquired from sensors, to high-level management and business data. In these applications it is desirable to process user requests within their deadlines using fresh data. As has been pointed out by others, there is no methodical way of handling embedded database systems, and there is a big need for embedded database systems for embedded systems [Ortiz 2000].

Current research and previously developed techniques that have focused on managing data in embedded real-time systems can be categorized as follows:

- Work on algorithms for managing resources (CPU, data, bandwidth) that ensure a performance metric, e.g., data quality (temporal correctness, consistency, freshness, and accuracy). Thus, the focus has been on concurrency control, scheduling, data routing (sensor networks), logging, and recovery.
- Work on real-time database architectures appropriate for real-time systems (BerkeleyDB, BeeHive, COMET, DeeDS, EagleSpeed, Rodain, Starbase).

There are several research issues that need to be resolved that can have a major impact on how data is maintained. Technology advances allow us to understand how we physically should go about building small embedded real-time systems with the support for efficient managing of data. However, two key challenges are

- how to model real-time data and its quality attributes conceptually at design time to ensure that data requirements are satisfied, and
- the efficient exchange of data between small embedded real-time systems and larger information systems, e.g., integration of embedded sensor networks with command-and-control systems that build the global view by collecting and aggregating data from subsystems.

If these challenges can be conquered, the impact is expected to be major, as it would allow for a common information infrastructure and framework for maintaining data and its quality attributes in real-time systems and for integrating the data to higher level systems.

5.3 APPROACH

The approach taken has been to identify the technical gaps and shortcomings of distributed real-time systems (previous work has primarily focused on centralized systems) that hinder their efficient management of data. Specifically, we have characterized the specific requirements and quality attributes of data in embedded sensor networks; embedded sensor networks are increasingly important in DoD applications and part of net-centric warfare scenarios, as the embedded sensor networks exchange data with control-and-command systems. To contribute to a solution, we have focused on developing a framework for modeling quality attributes of the data at the design time of distributed real-time systems, to both aid the designer when it comes managing the data and to enable simple exchange of data with higher layer systems.

Incorporating a framework for managing data effectively using model-based engineering for building real-time systems is conducive to building assurance of the viability of a system architecture in the design phase and the pre-implementation phase, especially when exploited in large scale across several systems.

5.4 COLLABORATIONS

SEI staff involved in this project have been Jorgen Hansson (Performance Critical Systems), Peter Feiler (PCS), Aaron Greenhouse (PCS), and Dennis Smith (ISIS).

Academic collaboration has included Prof. Sang H. Son (University of Virginia), Prof. John A. Stankovic (University of Virginia), Vibha Prasad (University of Virginia), and Dr. Ming Xiong (Bell Laboratories, Network Data and Services Research Department). Dr. Xiong has provided his own support.

5.5 EVALUATIONS

We have proposed that this project be evaluated on our success in achieving the following deliverables, previously outlined in the overall approach:

- Develop a framework for modeling quality attributes of data at the design time of distributed real-time systems.
- Characterize specific data requirements of embedded sensor networks.

Our objective has also been to find an industrial partner for future work and attempt to develop a plan on how issues of data management can incorporated into the Dynamic Systems Program.

5.6 RESULTS

Data quality in a net-centric system can be compromised in a number of ways: confidentiality might be breached due to lack of adequate encryption or incorrectly assigned security clearance for subjects operating on an object; the confidence level of the data might be insufficient in part due to the set size of active sensors or environmental conditions; the accuracy of sensor readings is affected by environmental conditions; correctness of data is low, as it is a function of its temporal coherence, that is, correctness of data decays over time. These attributes together determine the quality of the data forwarded to the subsystems from the net-centric system. Enforcing 100% accuracy, temporal correctness, and confidence is in practice not always possible nor desirable, as the cost becomes too significant with respect to monetary means, computational power, power consumption, quality of sensor, etc. It is more desirable to ensure that data quality is within acceptable tolerance levels of the applications, not jeopardizing their correctness.

We believe that model-driven engineering is conducive and of paramount importance to validating the quality of data and thereby ensuring that high-level systems are provided with data of sufficient quality and consistency driven by the application requirements. Architectural description languages have successfully been applied to prove system properties, often with a propensity toward task- or component-centric perspectives and, thus, generally preclusive of data quality attributes. We have developed a modeling framework for validating, a priori to the implementation phases of the system, the enforcement of the data quality attributes *temporal correctness/freshness*, *data accuracy/precision*, *data confidence*, and *data confidentiality*. We use the architecture and analysis description language AADL as a platform for this research.

To enable modeling and validation of data quality attributes in a model-driven approach using AADL, we have addressed the following technical issues: (i) identifying the appropriate aspects/levels of validating a quality attribute, i.e., what it is we want to validate; (ii) developing adequate analytical methods for validating these aspects (these methods have been implemented in the OSATE tool platform), i.e., how validation is performed; (iii) identify representational issues, i.e., how the properties are represented in an AADL model to enable analysis.

We have investigated how to model and verify that several data quality attributes of a modeled system are enforced. We have developed a framework that models and validates confidentiality (i.e., ensuring that sensitive data is disclosed to and accessed only by authorized users and applications, thus enforcing prevention of unauthorized disclosure of information). The framework can be extended to incorporate integrity (i.e., ensuring prevention of unauthorized modifications of data). It supports modeling and validation according to the well-known Bell-LaPadula model, controlled sanitization of data, and enforcement of sound engineering principles that we have identified that characterize a secure architecture. The framework also supports representation of the Chinese Wall security model and the role-based access model. The developed framework allows validation of several properties of a system: data confidentiality is maintained in an end-to-end flow across a system; software with security requirements is mapped to sufficiently secure hardware platforms (CPUs, memory, communication channels); software components communicate securely with other software components, etc.

Another dimension of successful model-based engineering is the support for incremental modeling. In the context of security and confidentiality, the framework supports the following levels of modeling and validation of various details and strengths: modeling of security requirements of objects, e.g., data, and generating security clearances for the subjects accessing the objects (synthesis); modeling of security requirements and clearances of objects and subjects respectively, validation of confidentiality, and based on the outcome, generating an access matrix/security table specifying allowed operations (the matrix can be used in the final system); or modeling and validation of confidentiality across subjects and objects and the access matrix.

The framework supports modeling and validation of data confidence and data accuracy, ensuring that all users only use data objects that are within the specified limits of confidence and accuracy. Data objects can also be annotated with the temporal validity intervals, capturing for how long a data object is valid.

An important determinant of the viability of a system architecture is the impact that architectural changes have on quality attributes, and similarly, adjusting the level of one quality attribute can impact other quality attributes. A tradeoff and impact analysis using model-based engineering will aid in exploring and analyze the effects. For example, changing security policy, such as increasing the encryption level, has several effects: (i) processing demand increases with the complexity of the encryption, (ii) network bandwidth demand increases with the encryption complexity, the length of the encryption key, and the frequency with which the key is replaced and communicated, (iii) end-to-end latency occurs, as data is now encrypted and decrypted and takes longer to communicate, and (iv) power consumption increases due to (i) and (ii). Another example that illustrates dependencies among the quality attributes is the following. To increase confidence in sensor data, one needs to perform data sampling more frequently (as confidence in a data reading decreases over time), which in turn generates more network traffic; the communication time increases as the network load increases, which adds to the latency of delivery of the data sensor reading, which in turn will lower the confidence in the data. This demonstrates the delicate balance and the need for validation of systems architecture. A challenge for the future is developing a generalized way of expressing dependencies and cost formulas, using a non-modal mechanism, to allow for validation of an architecture with respect to multiple quality attributes under different configurations.

The Eaton company has shown strong interest in collaborating with the SEI on model-based engineering using AADL for validating sensor networks of net-centric computing. A contract for long-term collaboration between the SEI and Eaton has been developed.

5.7 PUBLICATIONS AND PRESENTATIONS

- Jorgen Hansson and Aaron Greenhouse, "Modeling and Validation of Security and Confidentiality," SEI Technical Note (forthcoming).
- Jorgen Hansson, Aaron Greenhouse, and Peter H. Feiler "Modeling and Validation of Data Quality Attributes in Distributed Embedded Systems and Embedded Sensor Networks," SEI Technical Note (forthcoming).
- Invited presentation at the Integrated Air Weapons Systems Conference in Washington, D.C., August 2006. Conference organized by IDGA.
- Invited presentation at the Net-Centric Computing Conference, Paris, June 2006; Conference organized by SAGEM as part of an effort of setting up an international network of excellence.

5.8 BIBLIOGRAPHY

[Akyildiz 2002]
Akyildiz, I. F.; Su, W.; Sankarasubramaniam, Y.; & Cayirci, E. "A Survey on Sensor Networks." *IEEE Communication Magazine 40*, 8 (August 2002): 102-114.

[Beehive]
Stankovic, J. et al. *BeeHive Project*. http://www.cs.virginia.edu/~stankovic/mmdb.html

[BerkeleyDB 2007]
Oracle Berkeley DB Product Family. http://www.oracle.com/database/berkeley-db/index.html (2007).

[Cerpa 2004]
Cerpa, A. & Estrin, D. "ASCENT: Adaptive Self-Configuring Sensor Networks Topologies." *IEEE Transactions on Mobile Computing 3*, 3 (July-Aug. 2004):272--285.

[COMET]
COMET. http://www.ida.liu.se/labs/rtslab/projects/ARTES_EmbeddedDatabases/

[DeeDS]
Andler, S. F.; Hansson, J.; Eriksson, J.; Mellin, J.; Berndtsson, M.; & Eftring, B. "DeeDS: Towards a Distributed Active and Real-Time Database System." *Special Issue on Real Time Data Base Systems, SIGMOD Record 25*, 1 (March 1996).

[Deshpande 2003]
Deshpande, A.; Nath, S.; Gibbons, P. B.; & Seshan, S. "Cache-and-Query for Wide Area Sensor Databases," 503-514. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. New York, NY: ACM Press, 2003.

[EagleSpeed]
Lockheed Martin. *EagleSpeed Real-Time Database Manager*.
http://www.lockheedmartin.com/wms/findPage.do?dsp=fec&ci=11378&rsbci=0&fti=126&ti=0&sc=400

[Intan+ 2003]
Intanagonwiwat, C.; Govindan, R.; Estrin, D.; Heidemann, J.; & Silva, F. "Directed Diffusion for Wireless Sensor Networking." *IEEE Transactions on Networking 11*, 1 (Feb. 2003): 2-16.

[Madden 2005]
Madden, S.; Franklin, M.; Hellerstein, J.; & Hong, W. "TinyDB: An Acquisitional Query Processing System for Sensor Networks." *ACM Transactions on Database Systems (TODS) 30*, 1 (March 2005): 122-173.

[Ortiz 2000]
Ortiz, S. Jr. "Embedded Databases Come Out of Hiding." *IEEE Computer 33*, 3 (March 2003): 16-19.

[Ramamritham 2004]
Ramamritham, K.; Son, S. H.; & DiPippo, L. "Real-Time Databases and Data Services." *Real-Time Systems Journal 28* (Dec. 2004): 179-216.

[Rodain 1999]
Rodain Project. http://www.cs.helsinki.fi/research/rodain/ (1999).

[Shenker 2003]
Shenker, S.; Ratnasamy, S.; Karp, B.; Govindan, R.; & Estrin, D. "Data-Centric Storage in Sensornets." *SIGCOMM Computer Communication Review 33*, 1 (2003): 137-142.

[Stankovic 2003]
Stankovic, J.; Abdelzaher, T. F.; Lu, C.; Sha, L.; & Hou, J. C. "Real-Time Communication and Coordination in Embedded Sensor Networks," 1002-1022. *Proceedings of the IEEE 91*, 7 (July 2003).

[StarBase 1994]
The StarBase Project. http://www.cs.virginia.edu/~vadb/starbase.html (1994).

[STRIP]
The STRIP Project. http://www-db.stanford.edu/strip.html

[Yao 2002]
Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks." *ACM SIGMOD Record 31*, 3 (2002): 9-18.

[Zhao 2004]
Zhao, F. & Guibas, L. *Wireless Sensor Networks – An Information Processing Approach*. San Fransisco, CA: Morgan Kaufmann Publishers, Inc., 2004.

6 Technology Foundations for Computational Evaluation of Software Security Attributes

Gwendolyn H. Walton, Thomas A. Longstaff, Richard C. Linger

6.1 PURPOSE

In the current state of practice, security properties of software systems are typically assessed through labor-intensive evaluations by security experts who accumulate knowledge in bits and pieces from architectures, specifications, designs, code, and test results. Ongoing program maintenance and evolution limit the relevance of even this hard won but static and quickly outdated knowledge. When systems operate in threat environments, security attribute values can change very quickly. To further complicate matters, security strategies must be sufficiently dynamic to keep pace with organizational and technical change.

The purpose of this IRAD project was to define a fundamentally different approach to specification and evaluation of software security attributes. This research recognized and leveraged the fact that the problem of determining the security properties of software comes down in large measure to the question of how programs behave when invoked with stimuli intended to cause harmful outcomes. Because security properties have functional characteristics amenable to computational approaches, the research focused on the question "What can be computed with respect to security attributes?" The results show that security properties of software can in fact be evaluated through computational automation.

6.2 BACKGROUND

Historically, security analysis has required consideration of a variety of artifacts and concerns. Researchers have applied analytical tools such as model checking, concurrent sequential process modeling, and rule-based systems to model, analyze, and verify security protocols. In recent years, several research threads have emerged that address the difficult problems of security metrics and quantitative analysis and evaluation of security attributes. This research to date has typically focused on finding approximate solutions. Many of the studies have been based on assumptions that severely constrain the operational utility of the results. In addition, many security analysis methods require that a separate model of the software be developed to include consideration of the users and/or the environment in which the software is executed. While these approaches provide valuable insights, none of them allow security analysts to state with certainty how the software under consideration behaves under *all* circumstances of use with respect to security attributes of interest**.**

Further complicating the issue, the published definitions for security attributes lack theoretical foundations and cannot support a disciplined approach to analysis of software and system security. For example, various U.S. government publications describe *privacy* as an interest, a freedom, an ability, and a right. A method is needed for security attribute specification and analysis based on (1) the actual behavior of software and (2) security attribute definitions that consider only data and transformations of data.

6.3 APPROACH

The mathematics of functions provides a solid point of departure for computational analysis of security attributes. Desired security attributes can themselves be specified in terms of data and transformation of data, thereby permitting software to be evaluated for conformance or not through comparison of behavioral requirements of security attributes to the functional behavior of the software. The emergence of CERT's new function extraction (FX) technology, unavailable to previous researchers, provides the critical first step for computational security attribute analysis by supporting the derivation of the full functional behavior of programs as a starting point for security analysis.

FX technology can be applied to programs to yield their complete calculated behavior expressed in terms of net effects on data. The function extraction process derives the as-built specification of software, that is, the behavior that has actually been implemented. This derived behavior can be compared to requirements and specifications to determine whether the software is indeed a correct implementation. Thus, the derived behavior can be used to determine whether the software meets security requirements if they have been specified in behavioral terms.

6.3.1 The CSA Approach

While analysts have often characterized many security attributes as "non-functional" properties of programs, it turns out that they are in fact fully functional and thereby subject to FX-based automated analysis. Complete definitions of the required behavior of security attributes of interest can be created based solely on data and transformations of data. These definitions can then be used to analyze the security properties of programs. Thus, as illustrated in Figure 6-1, computational security attribute (CSA) analysis consists of three steps:

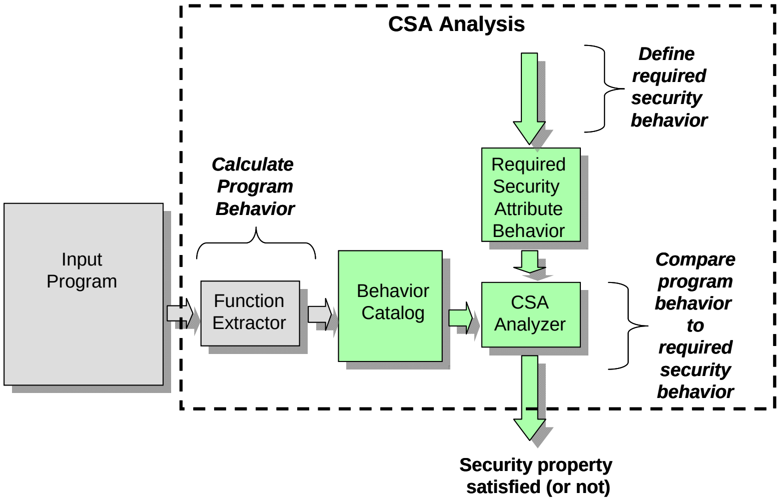

CSA Analysis
Define required security behavior
Calculate Program Behavior
Required Security Attribute Behavior
Input Program
Function Extractor
Behavior Catalog
CSA Analyzer
Compare program behavior to required security behavior
Security property satisfied (or not)

Figure 6-1: The CSA Approach

- **Define required security behavior.** Specify security attributes in terms of required behavior during execution expressed in terms of data and transformations of data.
- **Calculate program behavior.** Apply function extraction to create a behavior catalog that specifies the complete "as built" functional behavior of the code.
- **Compare program behavior to required security behavior.** Compare the computed behavior catalog with required security attribute behavior to verify whether it is correct.

The security attributes investigated in this research are availability, authentication, authorization, confidentiality, integrity, non-repudiation, and privacy. It turns out that the behavioral requirements for each of these attributes can be completely described in terms of data items and constraints on their processing. The processing can be expressed, for example, as logical or quantified expressions or even conditional concurrent assignments, which can be mechanically checked against the calculated behavior of the software of interest for conformance or non-conformance with security attribute requirements. The details of CSA application to analysis of each of these attributes are described in the SEI technical report that was generated as part of the research (see Publications below). An introduction to CSA is provided here.

6.3.2 Use of a Trusted Mechanism

Each security attribute requires the use of one or more trusted mechanisms that are implemented in software components. FX technology can be used to certify a mechanism as trusted. A behavior catalog for a mechanism can be generated to describe all cases of behavior in terms of its data and the transformations it carries out on that data. The behavior catalog can then be analyzed in terms of control data and evidence data to ensure the following:

- The trusted mechanism sets the values of control data, which indicates whether the mechanism executed correctly.
- If control data indicates that the mechanism executed correctly, there exists evidence data to show that the data transformation was performed in a manner that satisfies the defined security specification.

Note that the implementation of a security attribute may include a trusted third party to acquire, authenticate, and adjudicate evidence of transactions. However, for the purposes of behavior specification of security attributes, the specific mechanism and actors are not relevant. All that is needed is a precise specification of the data and the transformations of the data and any constraints concerning those transformations.

6.3.3 Trusted Data Transmission

As illustrated in Figure 6-2, the requirements for trusted data transmission are as follows:

- A trusted data transmission mechanism is used for all data transmissions. If the mechanism is not available or the mechanism fails, the requirement fails.
- No mechanism outside this trusted data transmission mechanism sets the value of the control data that indicates whether the data transmission mechanism executed correctly.

The process for determining whether data transmission security properties are satisfied by the data transmission components of a system consists of the application of the CSA analysis process illustrated in Figure 6-1, where the input is the data transmission components of the system and the output is a determination of whether the data transmission security requirements have been satisfied. Similarly, the process for determining whether the remainder of the system can modify the control data set by the data transmission components consists of application of the CSA analysis process, where the input to the process is the software for the entire system and the output is a determination of whether the control data set by the data transmission components can be modified by any other part of the system.

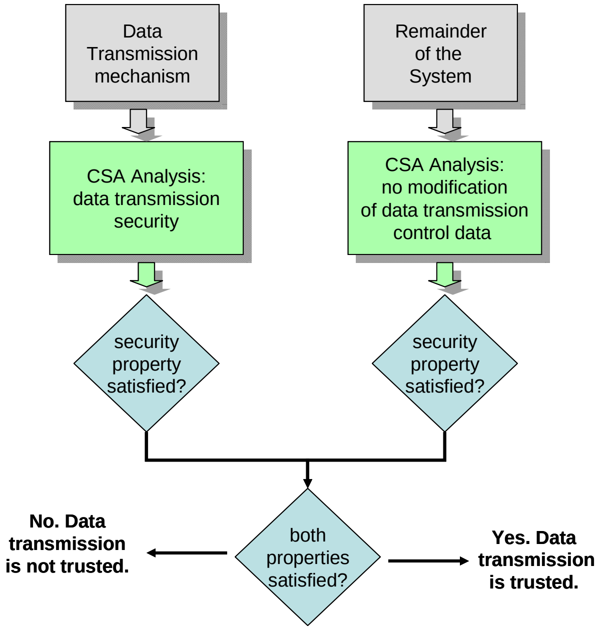

Data Transmission mechanism
Remainder of the System
CSA Analysis: data transmission security
CSA Analysis: no modification of data transmission control data
security property satisfied?
security property satisfied?
both properties satisfied?
No. Data transmission is not trusted.
Yes. Data transmission is trusted.

Figure 6-2: Requirements for Trusted Data Transmission

To verify that a data transmission mechanism is trusted, it is necessary to verify the data that provides the evidence related to data transmission. For example, the specification for the data that provides evidence of valid data transmission may describe the mechanism by which each data message output incorporates a shared (between sender and receiver) data item that can be used to verify that the transformation worked correctly. Assignments to this shared data must not be reversible (i.e., guaranteed encryption.) As another example, suppose the FX behavior catalogs for all of the code have been examined to verify that all data transmissions in the system occur as a result of calls to function YYY. To verify the necessary security properties for data transmission, it is necessary to examine function YYY's behavior catalog to determine the net effect of the data transformations related to any conditions for which invalid data transmission could occur.

6.3.4 The Authentication Security Attribute

If the authentication security attribute is satisfied, the system will allow the requested program to be executed only if the user has previously been determined to be a trusted user. To verify authentication, one must examine the net effects on the control data related to authentication: verify the data that provides evidence that the binding took place, and verify that this evidence data was not changed before completion of any operation that required authentication. As illustrated in Figure 6-3, the requirements for authentication are as follows:

- A trusted data transmission mechanism is always used for every data transmission.
- A trusted identification mechanism is always used to provide proof of a user's identification. Note that the "user" to be identified may be a person, a process, a program, or other entity.
- A trusted binding mechanism is always used to bind user data (user identification, password, or other information to confirm the identification, and the system information that provides the proof of the identification) to an execution environment.
- No other mechanism outside the trusted mechanisms sets the value of any of the control data that indicates whether each of the trusted mechanisms executed correctly and that indicates the status of the bound data.
- If any of the above requirements or mechanisms fails, authentication fails.

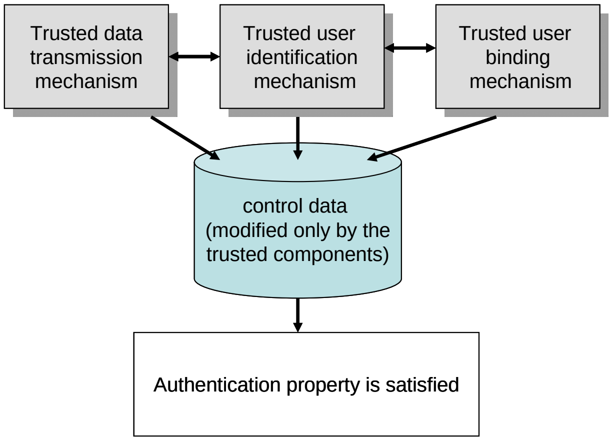

Trusted data transmission mechanism
Trusted user identification mechanism
Trusted user binding mechanism
control data (modified only by the trusted components)
Authentication property is satisfied

Figure 6-3: Requirements for Authentication Property

The mechanism for using CSA to determine whether the data transmission is trusted was discussed earlier. The analysis of the user identification mechanism and the user binding mechanism applies the CSA approach by proceeding along the same lines to determine whether each of these mechanisms is trusted.

6.3.5 Other Security Attributes

The requirements for the authorization security attribute build on the requirements for trusted data transmission and trusted authentication as follows:

- A trusted authentication mechanism is always used to authenticate the user. Note that this requirement includes a requirement for trusted data transmission and authentication.
- A trusted lookup mechanism is always used to determine that the user has the right to complete the specified request.

- No other mechanism outside the trusted mechanisms sets the value of any of the control data that indicates whether each of the trusted mechanisms executed correctly and that identifies the authorized user and the scope of the authorization.
- If any of the above requirements or mechanisms fails, authentication fails.

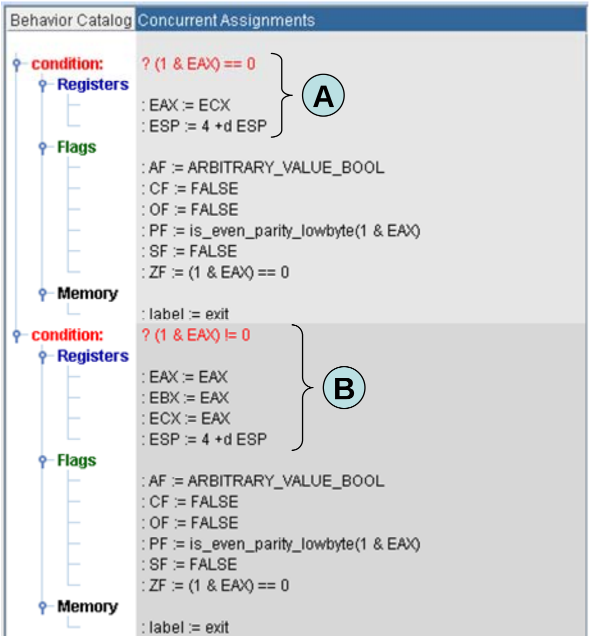
Similarly, the requirements for non-repudiation build on the requirements for trusted data transmission, trusted authentication, and trusted authorization. For the purposes of this discussion we treat data change as a special case of data transmission, where receipt of the data transmission includes making and logging the requested change to the dataset. To verify non-repudiation one must examine the net effects on the control data related to non-repudiation.

The requirements for the other traditional security attributes build on the requirements for trusted mechanisms for data transmission, authentication, authorization, and non-repudiation. The CSA analysis for each of the requirements is performed in a manner similar to the analysis of requirements for authentication. Details are provided in the technical report.

6.3.6 A Miniature Illustration of CSA Using an FX System

Consider the screen image of a behavior catalog, generated by the Function Extraction for Malicious Code (FX/MC) prototype, as shown in Figure 6-4. This system is under development by the SEI's CERT STAR*Lab to compute the behavior of malicious code expressed in the Intel assembly language. FX/MC produces catalogs that represent the net behavior of programs in terms of disjoint conditional concurrent assignment statements. That is, each case of behavior is expressed by a condition and a set of non-procedural assignments that define the final values of registers, flags, and memory locations on exit from an assembly language program.

This notional example illustrates application of the CSA approach to determine whether a security requirement is satisfied. Suppose that the specification of requirements for a security attribute includes a constraint that the value of the second argument to each call to function XXX must not be equal to 4. A constraint such as this, expressed in terms of concrete data operations and values, could be part of the requirements specification for any of the security attributes discussed above.



```
Behavior Catalog   Concurrent Assignments

condition:        ? (1 & EAX) == 0
  Registers
                  : EAX := ECX
                  : ESP := 4 +d ESP
  Flags
                  : AF := ARBITRARY_VALUE_BOOL
                  : CF := FALSE
                  : OF := FALSE
                  : PF := is_even_parity_lowbyte(1 & EAX)
                  : SF := FALSE
                  : ZF := (1 & EAX) == 0
  Memory
                  : label := exit
condition:        ? (1 & EAX) != 0
  Registers
                  : EAX := EAX
                  : EBX := EAX
                  : ECX := EAX
                  : ESP := 4 +d ESP
  Flags
                  : AF := ARBITRARY_VALUE_BOOL
                  : CF := FALSE
                  : OF := FALSE
                  : PF := is_even_parity_lowbyte(1 & EAX)
                  : SF := FALSE
                  : ZF := (1 & EAX) == 0
  Memory
                  : label := exit
```

Figure 6-4: The Behavior Catalog Computed by the FX/MC Prototype

Imagine that the catalog of Figure 6-4 defines the behavior of a fragment of code that executes immediately before the program calls function XXX with the second argument equal to the value stored in register EAX. The computed behavior highlighted as section A is the behavior with respect to register values for the condition *?(1 & EAX) == 0*). This condition means "if the value of register EAX at the beginning of this fragment of code is even." As shown on the figure, if this condition is true, the value of register EAX after executing this fragment of code is equal to the initial value of register ECX, and therefore the value of the second argument to function XXX will be the initial value of register ECX. In contrast, the computed behavior highlighted as section B is the behavior with respect to register values for the condition *?(1 & EAX) != 0*). This condition means "if the initial value of register EAX is odd." As shown on the figure, if this condition is true, the value of register EAX is unchanged and therefore the value of the second argument to function XXX will be the initial value of register EAX. Thus, the security constraint is not satisfied because, under either of these conditions, it is not possible to certify that the value of the second argument will never be equal to 4.

Note that, using FX technology, this analysis can take place in seconds through computational automation, thereby eliminating the need to study and understand the code by manual means.

6.4 COLLABORATIONS

The SEI team for this project was composed of Gwendolyn Walton, Thomas Longstaff, and Rick Linger. Several other researchers in the CERT Survivable Systems Engineering group provided valuable insights throughout the project: Tim Daly, Sven Dietrich, Howard Lipson, Nancy Mead, Mark Pleszkoch, and Stacy Prowell.

6.5 EVALUATION CRITERIA

The objective of the CSA IRAD study was to perform a feasibility study to determine the extent to which a new theory-based technology can be developed to permit security attributes to be evaluated through automatable analysis of the functional behavior of the software. That objective has been achieved, as discussed briefly below and in more detail in the SEI technical report that resulted from the research. The next step in CSA development is to create engineering examples and prototype tools to support demonstrations of this method for analysis and verification of security attributes.

6.6 RESULTS

This research resulted in the definition of a new analysis and verification method that can replace the historical definition of security attributes as subjective, high-level statements with computationally rigorous specifications. The new CSA technology has the potential to transform security attribute analysis and verification. One surprising result was the demonstration that all traditional security attributes are fundamentally based on a small number of behavioral assumptions. This result implies that, to meet security requirements, these fundamental behaviors must first be met in code. Thus, for example, it is not possible to add, say, a confidentiality capability to a network system without addressing the fundamental behavioral property of trusted data transmission and all that follows.

6.7 PUBLICATIONS

Walton, Gwendolyn H.; Longstaff, Thomas A.; & Linger, Richard C. *Technology Foundations for Computational Evaluation of Software Security Attributes* (CMU/SEI-2006-TR-021). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06tr021.html

7 Toward Interoperable Acquisition: The Example of Risk Management

B. Craig Meyers, Christopher J. Alberts, Eileen C. Forrester, Suzanne M. Garcia, D. Michael Phillips, Carol A. Sledge, James D. Smith II

7.1 PURPOSE

The purpose of this IRAD was to investigate the problem of interoperable acquisition using risk management as an example. The term *interoperable acquisition* is defined as

> the set of practices that enables acquisition, development, and operational organizations to more effectively collaborate to field interoperable systems of systems. This is achieved through sharing of relevant information and performing acquisition-related activities that enable the *collective behavior* of these organizations to successfully deliver systems of systems capabilities.

Risk management is an example of a typical acquisition practice. We also make the following definition:

> *Interoperable risk management* is the subset of interoperable acquisition practices that enables the acquisition, development, and operational organizations to identify and share the information that is needed to understand and mitigate risks that are inherent in a system-of-systems context.

There is an increased emphasis within and outside the DoD on network-centric operations based on concepts of information sharing through distributed sensor fusion, collaboration of virtual organizations, and exploitation of power at the edge of an organization. We would argue that these very same principles apply to the acquisition, development, fielding, and sustainment of operational systems. We see the concepts explored in this IRAD as being foundational to any attempt at moving the network-centric perspective to the acquisition community.

7.2 BACKGROUND

The acquisition system within DoD has often been described using the term *stove-pipe*, reflecting a system-centric perspective. This is intended to reflect that acquisitions are principally focused on a particular *system*. This system focus appears in statutes (e.g., Title 10), regulations (e.g., Federal Acquisition Regulations (FARs), and DoD regulations (notably DoDD 5000.1 and DoDI 5000.2).

The term interoperability is traditionally interpreted in an operational sense ("Can we plug the machines together?"). In a previous IRAD project we broadened this concept to include other aspects. For example, we use the term *programmatic interoperability* to mean interoperability between functions related to program management. Programmatic interoperability is but one important facet of interoperable acquisition. The system-centric approach common within DoD is at odds with the approaches needed to achieve interoperable acquisition [Levine 2003, Morris 2004]. A traditional system-centric approach can lead to predictable problems when applied in the context of acquisition related to a system of systems. The realities of a system of systems necessitate changes in the processes used to acquire, develop, field, and sustain operational capabilities.

7.3 APPROACH

The approach in this research was to develop an understanding of the main concepts that play a role in interoperable acquisition. These are shown in Figure 7-1.

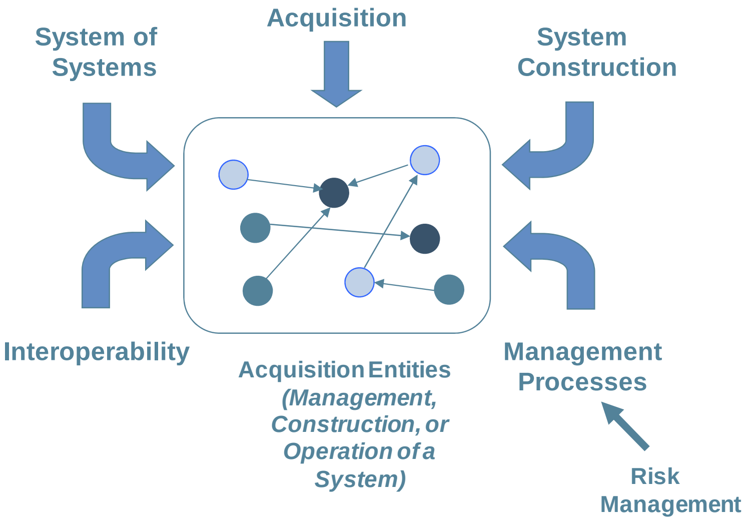


Figure 7-1: Concepts for Interoperable Acquisition

The entities that participate in acquisition are divided into three broad classes, namely, management, construction, and operation. These terms indicate roles to emphasize that successful interoperable acquisition depends on the practices and collaboration of these roles. The rationale for partitioning an acquisition in terms of these entities results from a previous SEI *System of Systems Interoperability* (SOSI) IRAD effort [Levine 2003, Morris 2004].

Given an understanding of the relevant concepts, the approach followed included

- gaining a fuller understanding of key topics, namely, acquisition, process, and risk management in the context of interoperable acquisition
- assessing the identified topics with regard to
  - identifying weaknesses that inhibit the goal of interoperable acquisition
  - identifying possible opportunities to improve practices within a chosen topic

7.4 COLLABORATIONS

The collaborators involved in our team included the following SEI members of the technical staff:

- Christopher J. Alberts
- Eileen C. Forrester
- Suzanne M. Garcia
- B. Craig Meyers (Lead)
- D. Michael Phillips
- Carol A. Sledge
- James D. Smith II

The external collaborators, who provided their own support, who also contributed to this work include

- Lynn Penn, Lockheed Martin
- Cathy Ricketts, PEO-IWS

7.5 EVALUATION CRITERIA

At the outset of this project we set forth the following success criteria:

- *Does this work clearly identify the impediments to multi-program acquisition for interoperable systems-of-systems?* The technical note *Interoperable Acquisition for Systems of Systems: The Challenges* responds to this criterion [Smith 2006].
- *Does this work identify a comprehensive framework for multi-program acquisition such that interoperability can be achieved?* The principles and elements of a framework for interoperable acquisition are discussed from a process perspective in the technical note *Process Considerations for Interoperable Acquisition* [Garcia 2006], which are then considered for schedules in the technical note *Schedule Considerations for Interoperable Acquisition* [Meyers 2006b].
- *Can such a framework be provided for the case of risk management?* Issues related to risk management are addressed in the technical note *Risk Management Considerations for Interoperable Acquisition* [Meyers 2006a].
- *Does this IRAD clearly identify an area of future work for the SEI?* During the final outbrief of this work, a number of possible follow-on activities were identified and discussed with SEI management. These are identified in the following section.

7.6 RESULTS

The results of this IRAD are organized around the key topics that framed the research conducted. Each is discussed below.

Challenges

With regard to identification of challenges, the following findings are noted [Smith 2006]:

There are a number of key obstacles to achieving interoperable acquisition. These include

- a failure to understand how the relations between various aspects of interoperability (e.g., programmatic, constructive, and operational) influence acquisition in a system-of-systems context
- not understanding the importance of sharing relevant information in performing acquisition-related activities to enable collective behavior to successfully deliver system-of-systems interoperability
- “system-centric” versus “system-of-systems” thinking in all aspects of acquisition, development, fielding, sustainment, and operation
- the inappropriate use of centralized management with hierarchical organizational structures

The above findings are based on the experience of team members in working with acquisition programs, as well as several years of working on various aspects of systems of systems.

Risk Management

With regard to risk management, the following findings are noted [Meyers 2006a]:

- The current specifications related to risk management are ineffective to meet the needs of interoperable risk management.

  This finding is supported by an assessment of several current documents often cited by risk management practitioners. In particular, we found that there is not a common vocabulary that is specified sufficiently to achieve interoperability in the acquisition process. For example, the term *probability* is often used in risk management, represented as either a qualitative measure (e.g., high, medium, or low), or a quantitative measure. Yet the *relation* between these measures is not required or specified in any standard we examined.

  Of greater concern is that the documents examined do not include a specification of the behaviors expected to be performed by various entities to achieve interoperability in the risk management process. For example, if multiple organizations are engaged in acquisition and have agreed to collaboratively perform risk management, what behaviors are expected? If an organization identifies a new risk or develops a risk mitigation plan for an existing risk, should this information be shared with other organizations? And what depth of information needs to be shared? We acknowledge the difficulty of specifying trust-related behaviors such as these within a standards framework; however, absence of guidance is a hindrance.
- The current methods used in risk management are insufficient to meet the needs of interoperable acquisition. This finding is based on the experience of our team members, although no broad assessment was performed.
- Fundamental risk management is also inadequately practiced, even though standards and other guidance exist. This finding is also based on the experience of the team.

Possible follow-on work related to risk management includes the following:

- Identify the requirements needed for a risk management standard that addresses interoperable acquisition and work with the community to obtain consensus for such a standard.
- Identify and assess the methods currently being used in the practice of risk management by industry high-performing organizations.
- Continue to evolve the state of the art related to interoperable risk management techniques.
- Develop a strategy to increase risk management capability within the acquisition community—for both fundamental and interoperable applications.

Process

With regard to process, the following findings are noted [Garcia 2006]:

- The guidance in process reference models (e.g., the SEI CMMI framework [Chrissis 2003] and the preliminary report for CMMI-ACQ [Dodson 2006]) that is currently available could provide needed process support for interoperable acquisition, but it does not.

  This finding is based on an analysis of [Dodson 2006]. The document was examined for language in both the normative and informative elements that would support concepts of interoperable acquisition. The current draft of CMMI-ACQ steps back from CMMI-DEV in some ways that could provide useful support for interoperable acquisition (e.g., not dealing with any IPPD content). Throughout CMMI models, the practices for data management are not adequate to support interoperable acquisition. Further details on this analysis are provided in [Garcia 2006].
- The current practice of acquisition process management, based on the team's experience (supported by reports from outside the SEI such as the recent *Defense Acquisition Performance Assessment* (DAPA) report [Kadish 2006]), is insufficient to support interoperable acquisition.

  This finding is based on the experience of team members and is amply supported by work outside the SEI, notably the recent DAPA report [Kadish 2006].

Possible follow-on work in regard to improving processes used to support interoperable acquisition includes the following:

- Develop an interpretive guide for using CMMI-DEV and the proposed CMMI-ACQ in an interoperable acquisition context.
- Consider addition of a *Generic Practice* for data management, at least to CMMI-ACQ if not CMMI-DEV as well.
- Participate in and contribute to work that has started exploring multi-organizational maturity models and multi-organization process improvement.
- Provide stronger support mechanisms within the acquisition community for the use of existing integrated product and process development practice.
- Identify ways to deconflict the contrasting goals and rewards of system-centric and acquisition in a system-of-systems context.

Schedule

Concerns about schedule are paramount in the acquisition of a system and are amplified in a system of systems context [Smith 2006]. We examined the role of schedule in terms of interoperable acquisition from the perspective of a *Gedanken* red team[5] [Meyers 2006b]. That is, given information typical of a schedule, what are the questions that might be expected to be posed by a red team? Many questions were developed in the following areas: basic information, organization and dependencies, shared information, approval, risks, and dealing with change.

The general conclusion from this experiment was that a *schedule is a window into a program.* That is, discussion of schedule can vary far and wide. There is information expected to be provided concerning a schedule; however, that information is often oriented toward a schedule for a particular system.[6] Of relevance to an interoperable acquisition, there is additional information that one would like, such as

- information relevant to a milestone, such as exit criteria of any product expected to be provided at a specified milestone
- estimate of variance of a milestone
- confidence of any schedule estimates
- sharing of risk data that could have an impact on schedule
- expected behaviors regarding sharing of schedule information
- approval status of a milestone and the role of collaborative decision making in that process
- status of supplementary information associated with a schedule[7]

The examination of schedule considerations supports our finding regarding the lack of sufficient specification and practice to provide the necessary support to interoperable acquisition cited earlier. This is further illustrated by the connection between schedule and risk management considerations. One would expect that there should indeed be such a connection; in particular, the data that needs to be shared and the expected behaviors of participants need to be identified—and adhered to—to gain broader success.

Possible follow-on work in this area would include identification of the data related to schedule that needs to be shared. More importantly, concern must be given to the expected behaviors of participants that constitute the subject of schedule management in a system of systems environment.

[5] The term *Gedanken* red team refers to a thought experiment involving a red team. Originally used in the sense of a *Gedanken* experiment in physics, it refers to an imagined scenario that is used to help gain an understanding of the domain of the experiment. The methodology is *a priori*, as opposed to empirical.

[6] We are speaking of the Data Item Description for an *Integrated Master Schedule*, DI-MGMT-86150, March 30, 2005.

[7] The well-known DODI 5000.1 requires much information for certain types of milestones, such as certification of Clinger-Cohen Act compliance, technology development strategy, market research, and independent cost estimates. Such information is required according to statute (e.g., Title 10) and DoD regulations.

7.7 PUBLICATIONS

The following documents have been created as a result of this effort:

- Garcia, Suzanne. M.; Forrester, Eileen; Alberts, Christopher. J.; & Meyers, B. Craig. *Process Considerations for Interoperable Acquisition*. SEI technical note, forthcoming.
- Meyers, B. Craig. *Risk Management Considerations for Interoperable Acquisition* (CMU/SEI-2006-TN-032). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06tn032.html
- Meyers, B. Craig, & Sledge, Carol A. *Schedule Considerations for Interoperable Acquisition* (CMU/SEI-2006-TN-035). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06tn035.html
- Meyers, B. Craig & Smith, James D. II. *Programmatic Interoperability* (CMU/SEI-2007-TN-012). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2007.[8]
- Smith, James D. II & Phillips, D. Michael. *Interoperable Acquisition for Systems of Systems: The Challenges* (CMU/SEI-2006-TN-034). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06tn034.html

Several project members also conducted a mini-workshop with a customer regarding interoperable acquisition in the context of cost and budget estimates. Results of this work will appear in a future publication.

7.8 SUMMARY

This IRAD effort has examined a number of topics we believe are critical to success for interoperable acquisition. Special concern was given to risk management, process, and schedule considerations. Several barriers were identified that inhibit successful acquisition in a systems of systems environment. Proposed follow-on work could provide some of the enablers needed to address particular barriers; however, there are important aspects of this problem that are outside the scope of the SEI.

We close on a philosophical point related to one of our initial assertions that has been confirmed by our experience on the IRAD project. The increased emphasis on network-centric operations is critical for the acquisition community as well as the operational community. Toward that end, the concept of relevance—desired, though yet to be realized—is that of *network-centric acquisition*.

[8] This work was partially supported by the IRAD effort.

7.9 REFERENCES

[Chrissis 2003]
Chrissis, Mary Beth; Konrad, Mike; & Shrum, Sandy. *CMMI: Guidelines for Process Integration and Product Improvement*, 2nd edition. Boston MA: Addison Wesley, 2003.

[Dodson 2006]
Dodson, Kathryn M.; Hofmann, Hubert F., Ramani, Gowri; Yedlin, Deborah K.; Fisher, Matthew J.; & Kost, Keith. *Adapting CMMI for Acquisition Organizations: A Preliminary Report* (CMU/SEI-2006-SR-005). Pittsburgh PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06sr005.html

[Garcia 2006]
Garcia, Suzanne M.; Forrester, Eileen C.; Alberts, Christopher J.; & Meyers, B. Craig. *Process Considerations for Interoperable Acquisition*. SEI technical note, forthcoming.

[Kadish 2006]
Kadish, Ronald et al. *Defense Acquisition Performance Report*. Department of Defense. January 2006.

[Levine 2003]
Levine, Linda, Meyers, B,. Craig, Morris, Ed ; Place, Patrick R. H.; & Plakosh, Daniel. *Proceedings of the System of Systems interoperability Workshop* (CMU/SEI-2003-TN-016, ADA416429). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.
http://www.sei.cmu.edu/publications/documents/03.reports/03tn016.html

[Meyers 2006a]
Meyers, B. Craig. *Risk Management Considerations for Interoperable Acquisition* (CMU/SEI-2006-TN-032). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06tn032.html

[Meyers 2006b]
Meyers, B. Craig, & Sledge, Carol A. *Schedule Considerations for Interoperable Acquisition* (CMU/SEI-2006-TN-035). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/publications/documents/06.reports/06tn035.html

[Morris 2004]
Morris, Ed; Levine, Linda L; Meyers, B. Craig; Place, Patrick R. H; & Plakosh, Daniel. *System of Systems Interoperability (SOSI): Final Report* (CMU/SEI-2004-TR-004, ADA455619). Pittsburgh PA: Software Engineering Institute, Carnegie Mellon University, 2004.
http://www.sei.cmu.edu/publications/documents/04.reports/04tr004.html

[Smith 2006]
Smith, James D. II & Phillips, D. Mike. *Interoperable Acquisition for Systems of Systems: The Challenges* (CMU/SEI-2006-TN-034). Pittsburgh PA: Software Engineering Institute, Carnegie Mellon University, 2006.
http://www.sei.cmu.edu/publications/documents/06.reports/06tn034.html

8 An Attack Surface Metric

Pratyusa K. Manadhata, Jeannette Wing

8.1 PURPOSE

Measurement of security, both qualitatively and quantitatively, has been a long-standing challenge to the research community and is of practical import to the software industry today [CRA 2003, McGraw 2003, Vaughn 2003]. The software industry has responded to demands for improvement in software security by increasing effort into creating “more secure” products and services. How can industry determine whether this effort is paying off, and how can consumers determine whether industry’s effort has made a difference? Our work is motivated by the question faced by both industry and consumers today: How can we quantify a software system’s security?

We propose to use the measure of a system’s *attack surface* as an indication of the system’s security. While it is very difficult to devise metrics that definitively measure the security of software, prior work has shown that a system’s *attack surface measurement* serves as a reliable proxy for security. Howard et al. have measured the attack surfaces of seven different versions of Windows [Howard 2003], and we have measured the attack surfaces of four different versions of Linux [Manadhata 2004]. The results of both the Linux and Windows measurements confirm perceived beliefs about the relative security of the different versions. The measurement methods, however, are based on intuition and are hard to replicate. Our current work is focused on defining a metric to *systematically* measure a system’s attack surface.

We envision our *attack surface metric* to be useful to both industry and consumers. Software designers and developers can use our *attack surface metric* as a tool in the software development process; they can measure their system’s attack surface periodically during the software development phase and compare the results with previous measurements. They should strive toward reducing their system’s attack surface from one version to another to mitigate the security risk of their system. Software consumers can also use our metric to compare and differentiate between alternative and competing software systems. For example, system administrators can compare the attack surface measurements of different available Web servers in choosing one for their organization.

8.2 APPROACH

Intuitively, a system’s attack surface is the set of ways in which an adversary can enter the system and potentially cause damage. We know from the past that many attacks on a system, such as exploiting a buffer overflow, take place by sending data from the system’s operating environment into the system. Similarly, many other attacks on a system, such as symlink attacks, take place because the system sends data into its environment. In both these types of attacks, an attacker connects to a system using the system's *channels* (e.g., sockets), invokes the system's *methods* (e.g., API), and sends *data items* (e.g., input strings) into the system or receives data items from the system. An attacker can also send data indirectly into a system by using data items that are persistent. Specific examples of persistent data items are files, cookies, registry entries, and database records. An attacker can send data into a system by writing to a file that the system later reads. Similarly, an attacker can receive data indirectly from the system by using shared persistent data items. Hence an attacker uses a system's methods, channels, and data items present in the system's environment to attack the system. We collectively refer to a system's methods, channels, and data items as the system's *resources*.

Given the above observation, a system's attack surface is defined in terms of the system's resources. Not all resources, however, are part of the attack surface, and not all resources contribute equally to the measure of a system's attack surface. In order to measure a system's attack surface, we need to identify the relevant resources that are part of the system's attack surface and determine the contribution of each such resource to the system's attack surface. We have introduced a formal *entry point and exit point framework* to identify the relevant resources that contribute to a system's attack surface; we also have introduced the informal notions of *damage potential* and *effort* to estimate a resource's contribution to a system's attack surface [Manadhata 2005].

8.2.1 Attack Surface Definition

A system's attack surface is the subset of resources that an attacker can use to attack the system. An attacker attacks a system either by sending data into the system or by receiving data from the system; hence any resource that the attacker can use either to send data into the system or to receive data from the system is part of the system's attack surface. Intuitively, the more resources available to an attacker, the more ways a system can be attacked, hence the more insecure it is. We use the entry point and exit point framework to identify the resources that are part of a system's attack surface.

Entry Points

The methods of a system that receive data items from the system's environment are the system's entry points. For example, a method that receives input from a user or a method that reads a configuration file is an entry point. A method m of a system s receives data items *directly* if either (a) a user or a system in s's environment invokes m and passes data items as input to m, or (b) m reads from a persistent data item, or (c) m invokes the API of a system in s's environment and receives data items as the result returned. A method m receives data items *indirectly* if either (a) a method m_1 of s receives a data item d directly, and either m_1 passes d as input to m or m receives d as result returned from m_1, or (b) a method m_2 of s receives a data item d indirectly, and either m_2 passes d as input to m or m receives d as result returned from m_2. A method m of s is a *direct entry point* if m receives data items directly, and is an *indirect entry point* if m receives data items indirectly.

Exit Points

The methods of a system that send data items to the system's environment are the system's exit points. For example, a method that writes into a log file is an exit point. A method m of a system s sends data items *directly* if either (a) a user or a system in s's environment invokes m and receives data items as results return from m, or (b) m writes to a persistent data item, or (c) m invokes the API of a system in s's environment and passes data items as input to the API. A method m sends data items *indirectly* if either (a) m passes a data item d as input to a method m_1 and m_1 passes d either directly or indirectly to s's environment, or (b) a method m_2 receives a data item d as result returned from m and m_2 passes d either directly or indirectly to s's environment. A method m of s is a *direct exit point* if m sends data items directly, and is an *indirect exit point* if m sends data items indirectly.

Channels

An attacker uses a system's channels to connect to the system and attack the system. Hence a system's channels act as another basis for attacks. Specific examples of channels are TCP/UDP sockets and pipes.

Untrusted Data

An attacker uses persistent data items either to send data indirectly into the system or receive data indirectly from the system. A system might read from a file after an attacker writes into the file. Similarly, the attacker might read from a file after the system writes into the file. Hence the persistent data items act as another basis for attacks on a system. An *untrusted data item* of a system s is a persistent data item d such that a direct entry point of s reads from d or a direct exit point of s writes into d.

Attack Surface

By definition, the relevant resources that contribute to the attack surface are the set of entry points and exit points, the set of channels, and the set of untrusted data items. Hence a system's *attack surface* is the triple $\langle M, C, I \rangle$, where M is the set of entry points and exit points, C is the set of channels, and I is the set of untrusted data items of the system.

8.2.2 Attack Surface Measurement

A naive way of measuring a system's attack surface is to count the number of resources that contribute to the attack surface. This naive method that gives equal weight to all resources is misleading, since all resources are not equally likely to be used by an attacker. For example, a method, m_1, running as `root` is more likely to be used in an attack than a method, m_2, running as `non-root`; hence m_1's contribution to the attack surface is larger than m_2's.

We estimate a resource's contribution to a system's attack surface as a *damage potential-effort ratio*, where *damage potential* is the level of damage the attacker can cause to the system in using the resource in an attack and *effort* is the effort the attacker spends to acquire the necessary access rights in order to be able to use the resource in an attack. The higher the damage potential, the higher the contribution; the higher the effort, the lower the contribution.

Damage Potential-Effort Ratio

We use informal means to estimate damage potential and effort in terms of the attributes of a resource. The estimates depend on the kind of the resource, i.e., method, channel, or data item.

An attacker gains the same privilege as a method by using a method in an attack. For example, the attacker gains `root` privilege by exploiting a buffer overflow in a method running as `root`. Hence we estimate a method's damage potential in terms of the method's *privilege*. The attacker uses a system's channels to connect to a system and send (receive) data to (from) a system. A channel's *protocol* imposes restrictions on the data exchange allowed using the channel, e.g., a `TCP socket` allows raw bytes to be exchanged, whereas an `RPC endpoint` does not. Hence we estimate a channel's damage potential in terms of the channel's protocol. The attacker uses persistent data items to send (receive) data indirectly into (from) a system. A persistent data item's *type* imposes restrictions on the data exchange, e.g., a `file` can contain executable code, whereas a `registry entry` cannot. Hence we estimate a data item's damage potential in terms of the data item's type.

The attacker can use a resource in an attack if the attacker has the required *access rights*. The attacker spends effort to acquire these access rights. Hence for the three kinds of resources, i.e., method, channel, and data, we estimate the effort the attacker needs to spend to use a resource in an attack in terms of the resource's access rights.

We assign numbers to the values of the attributes to compute a numeric damage potential-effort ratio. We impose a total ordering among the attributes of the resources and assign numeric values in accordance to the total ordering. For example, we assume a method running as `root` has a higher damage potential than a method running as `authenticated` user; hence `root` > `authenticated user` in the total ordering, and we assigned a higher number to `root` than `authenticated user`. The exact choice of the numeric values is subjective and depends on a system and its environment. We assign the numeric values based on our knowledge of the system and its environment.

Attack Surface Measurement Method

We measure a system's attack surface along three dimensions by estimating the total contribution of the methods, the total contribution of the channels, and the total contribution of the data items to the system's attack surface.

1. Given a system s and its environment, we identify a set, M, of entry points and exit points, a set, C, of channels, and a set, I, of untrusted data items of s.
2. We estimate the damage potential-effort ratio, $der_m(m)$, of each method $m \in M$, the damage potential-effort ratio, $der_c(c)$, of each channel $c \in C$, and the damage potential-effort ratio, $der_d(d)$, of each data item $d \in I$.

3. The measure of s ’s attack surface is the triple

$$\langle \sum_{m \in M} der_m(m), \sum_{c \in C} der_c(c), \sum_{d \in I} der_d(d) \rangle$$

We have demonstrated the use of our attack surface metric by measuring the attack surfaces of two open source IMAP servers: Cyrus 2.2.10 and Courier-IMAP 4.0.1.

8.2.3 Current Status

A key challenge in security metric research lies in devising appropriate techniques for validating a security metric. Our current work is focused on developing both formal and empirical validation techniques for our attack surface metric. Using I/O automata [Lynch 1989], we have formally established a relation between a system’s attack surface and the number of *executions* allowed by the system that an adversary can use to attack the system; we have shown that if a system, A, has a larger attack surface compared to a system, B, then A allows a larger number of such executions compared to B. We are also exploring three different empirical validation techniques. First, we plan to establish a correlation between a system’s attack surface and the number of vulnerability bulletins released for the system; if a system, A, has a larger attack surface compared to a system, B, then we expect to see a larger number of bulletins for A compared to B. In collaboration with the Idaho National Laboratory (INL), we have measured the attack surfaces of two open source FTP servers: ProFTP 1.2.10 and Wu-FTP 2.6.2. We have also counted the number of times these two FTP servers are mentioned in CERT bulletins, MITRE CVEs, and the Bugtraq vulnerabilities database. The vulnerability bulletin counts are as expected; Wu-FTP has a larger attack surface compared to ProFTP, and the number of bulletins for Wu-FTP is more than the number of bulletins for ProFTP [Manadhata 2006]. Second, we are using machine learning techniques to show that the six attributes (method privilege and access rights, channel protocol and access rights, and data item type and access rights) used in our measurement method are good indicators of a resource’s damage potential and effort. Third, we are analyzing the data collected from honeypots to establish a correlation between a system’s attack surface and the number of observed attacks on the system.

8.3 RELATED WORK

Prior research on measurement of security has largely taken an attacker-centric approach, i.e., relying on the knowledge of the attacker’s capabilities and resources in order to assess a system’s security. In contrast, we take a system-centric approach. We measure a system’s attack surface in terms of the system’s inherent attributes without making any assumptions about the attacker’s capabilities or resources.

Alves-Foss et al. [Alves-Foss 1995] use the System Vulnerability Index (SVI)—obtained by evaluating factors such as system characteristics, potentially neglectful acts, and potentially malevolent acts—as a measure of a system’s vulnerability. The problem with the approach is that we might not be able to quantify the factors that determine a system’s SVI. For example, all the “physical security vulnerabilities” of a system are not known and hence cannot be estimated.

Littlewood et al. [Littlewood 1993] measure the operational security of a system by quantitatively estimating the intuitive notion of “the system’s ability to resist attacks.” Our attack surface metric quantitatively estimates the notion of “a system’s attractiveness to an attacker” based on the system’s design and operating environment.

Voas et al. [Voas 1996] propose the minimum-time-to-intrusion (MTTI) metric based on the predicted period of time before any simulated intrusion can take place. Similar to our attack surface metric, the MTTI metric is a relative metric that allows the users to compare different versions of the same system. MTTI, however, is computed using fault injection, and the MTTI value depends on the threat classes being simulated. In contrast, our attack surface metric does not depend on any particular threat class.

Ortalo et al. [Ortalo 1999] model a system’s known vulnerabilities as a privilege graph [Dacier 1994] and combine simple assumptions about the attacker’s behavior with the privilege graphs to obtain the attack state graphs. They analyze the attack state graphs using Markov techniques to obtain probabilistic measures of operational security. Our attack surface metric does not rely on the knowledge of either the vulnerabilities present in the system or the attacker’s behavior. For example, Ortalo et al. assign a weight to each arc in the privilege graph corresponding to the effort needed for the attacker to perform the privilege transfer corresponding to the arc. The weight assigned depends on factors such as available attack tools, time needed to perform the attack, and computing power available to the attacker. In contrast, we estimate the effort spent by an attacker to use a resource in terms of the resource’s access rights.

Schneier [Schneier 1999] uses attack trees to model the different ways in which a system can be attacked. Given an attacker goal, Schneier constructs an attack tree to identify the different ways in which the goal can be satisfied and to determine the cost to the attacker in satisfying the goal. Construction of an attack tree requires the knowledge of the system vulnerabilities, as well as the knowledge of the attacker’s behavior. Our attack surface metric does not rely on the knowledge of either the vulnerabilities present in the system or the attacker’s behavior.

8.4 COLLABORATIONS

Clyde Chittister was the SEI MTS involved in this project. We also collaborated with Mark A. Flynn and Miles A. McQueen of INL; the INL researchers provided their own support. We also had many fruitful discussions with Mike Howard and Jon Pincus of Microsoft.

8.5 EVALUATION CRITERIA

We have established the following success criteria for our project:

- developing a formal framework for systematically measuring a system’s attack surface,
- devising appropriate validation techniques for attack surface measurement, and
- demonstrating the suitability of our measurement method by measuring the attack surfaces of real-world systems.

8.6 RESULTS

We summarize the significant results of our project in the following paragraphs.

- We have introduced the formal entry point and exit point framework to identify the resources that contribute to a system's attack surface. We also have introduced the informal notions of damage potential and effort to estimate a resource's contribution to a system's attack surface. We have outlined a method to measure a system's attack surface.
- We have demonstrated our method by measuring the attack surfaces of two open source IMAP servers and two open source FTP servers.
- We have formally established a relationship between a system's attack surface and the number of possible attacks on the system. We are also exploring three empirical techniques for validating a system's attack surface measurement.

8.7 PUBLICATIONS AND PRESENTATIONS

8.7.1 Publications

Manadhata, P. K. & Wing, J. M. "An Attack Surface Metric." *First Workshop on Security Metrics*, Vancouver, BC, August 2006.

Manadhata, P. K.; Wing, J. M.; Flynn, M. A.; & McQueen, M. A. "Measuring the Attack Surfaces of Two FTP Daemons." *Second Workshop on Quality of Protection*, Alexandria, VA, October 2006.

8.7.2 Presentations

Wing, J. M. "Attack Surface Measurement." *CMU CyLab Partners Conference*, Pittsburgh, PA, April 2006.

Manadhata, P. K. "An Attack Surface Metric." *ARO CyLab Research Program Review*, Pittsburgh, PA, May 2006.

Manadhata, P. K. "An Attack Surface Metric." *First Workshop on Security Metrics*, Vancouver, BC, August 2006.

8.8 REFERENCES

[Alves-Foss 1995]
Alves-Foss, J. & Barbosa, S. "Assessing Computer Security Vulnerability." *ACM SIGOPS Operating Systems Review 29*, 3 (1995): 3–13.

[CRA 2003]
Computing Research Association. CRA Conference on "Grand Research Challenges in Information Security Assurance." Warrenton, VA, November 2003.
http://www.cra.org/Activities/grand.challenges/security/home.html

[Dacier 1994]
Dacier, M. & Deswarte, Y. "Privilege Graph: An Extension to the Typed Access Matrix Model," 317–334. *Proceedings of the European Symposium on Research in Computer Security*, 1994. Berlin, Germany: Springer-Verlag LNCS 875, 1994 (ISBN 3-540-58618-0).

[Howard 2003]
Howard, M.; Pincus, J.; & Wing, J. M. "Measuring Relative Attack Surfaces." *Proceedings of the International Workshop on Advanced Developments in Software and Systems Security*. Taipei, Taiwan, December 5–7, 2003.

[Littlewood 1993]
Littlewood, B.; Brocklehurst, S.; Fenton, N.; Mellor, P.; Page, S.; Wright, D.; Dobson, J.; McDermid, J.; & Gollman, D. "Towards Operational Measures of Computer Security." *Journal of Computer Security 2*, 2/3 (1993): 211–230.

[Lynch 1989]
Lynch. N. & Tuttle, M. "An Introduction to Input/Output Automata." *CWI-Quarterly 2*, 3 (September 1989): 219–246.

[Manadhata 2004]
Manadhata, P. & Wing, J. M. "Measuring a System's Attack Surface." In technical report CMU-CS-04-102. Pittsburgh, PA: Carnegie Mellon University, 2004.

[Manadhata 2005]
Manadhata, P. & Wing, J. M. "An Attack Surface Metric." In technical report CMU-CS-05-155. Pittsburgh, PA: Carnegie Mellon University, 2005.

[Manadhata 2006]
Manadhata, P.; Wing, J. M.; Flynn, M. A.; & McQueen, M. A. "Measuring the Attack Surfaces of Two FTP Daemons." *Proceedings of the Second Quality of Protection Workshop*. Alexandria, VA, October 30, 2006. New York, NY: ACM Press, 2006 (ISBN 1-59593-553-3).

[McGraw 2003]
McGraw, G. "From the Ground Up: The DIMACS Software Security Workshop." *IEEE Security and Privacy 1*, 2 (Mar.-Apr. 2003): 59–66.

[Ortalo 1999]
Ortalo, R.; Deswarte, Y.; & Kaâniche, M. "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security." *IEEE Transactions on Software Engineering 25*, 5 (Sep./Oct. 1999): 633–650.

[Schneier 1999]
Schneier, B. "Attack Trees: Modeling Security Threats." *Dr. Dobb's Journal*, December 1999.

[Vaughn 2003]
Vaughn, R. B.; Henning, R. R.; & Siraj, A. "Information Assurance Measures and Metrics - State of Practice and Proposed Taxonomy." *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36)*. Hawaii, January 6–9, 2003. New York, NY: IEEE Computer Society Press, 2003 (ISBN 0-7695-1874-5).

[Voas 1996]
Voas, J.; Ghosh, A.; McGraw, G.; Charron, F.; & Miller, K. "Defining an Adaptive Software Security Metric from a Dynamic Software Failure Tolerance Measure," 250–263. *Proceedings of the Eleventh Annual Conference on Computer Assurance*. New York, NY: IEEE Computer Society Press, 1996.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY
(Leave Blank)

2. REPORT DATE
July 2007

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Results of SEI Independent Research and Development Projects

5. FUNDING NUMBERS
FA8721-05-C-0003

6. AUTHOR(S)
Christopher J. Alberts, Bill Anderson, Len Bass, Matthew Bass, Philip Boxer, Lisa Brownsword, Sagar Chaki, Eileen C. Forrester, Suzanne M. Garcia, Peter H. Feiler, Dave Fisher, Aaron Greenhouse, Jorgen Hansson, Jim Herbsleb, James Ivers, Peter Lee, Richard C. Linger, Thomas A. Longstaff, Pratyusa K. Manadhata, B. Craig Meyers, D. Michael Phillips, Carol A. Sledge, James D. Smith II, Kurt Wallnau, Gwendolyn H. Walton, Jeannette Wing, Noam Zeilberger

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER
CMU/SEI-2007-TR-006

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
ESC-TR-2007-006

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified/Unlimited, DTIC, NTIS

12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)
Each year, the Software Engineering Institute (SEI) 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 2006 (October 2005 through September 2006).

14. SUBJECT TERMS
structural modeling, dynamic modeling, software acquisition, software-intensive systems, systems thinking, risk analysis, software architecture, organizational change, plug-in, binaries, software component, proof-carrying code, proofs, embedded system, real-time system, sensor networks, software security attributes, function extraction, interoperable acquisition, risk management, attack surface, attack surface measurement

15. NUMBER OF PAGES
80

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
UL