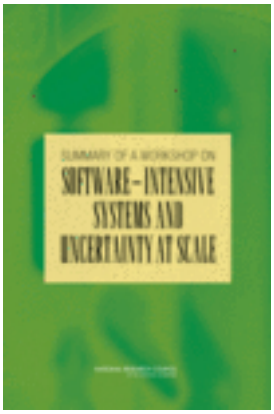

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale

Joan D. Winston and Lynette I. Millett, Editors,
Committee on Advancing Software-Intensive Systems Producibility, National Research Council

ISBN: 0-309-10845-4, 78 pages, 6 x 9, (2007)

This free PDF was downloaded from:
http://www.nap.edu/catalog/11936.html

Visit the National Academies Press online, the authoritative source for all books from the National Academy of Sciences, the National Academy of Engineering, the Institute of Medicine, and the National Research Council:

- Download hundreds of free books in PDF
- Read thousands of books online, free
- Sign up to be notified when new books are published
- Purchase printed books
- Purchase PDFs
- Explore with our innovative research tools

Thank you for downloading this free PDF. If you have comments, questions or just want more information about the books published by the National Academies Press, you may contact our customer service department toll-free at 888-624-8373, visit us online, or send an email to comments@nap.edu.

This free book plus thousands more books are available at http://www.nap.edu.

Copyright © National Academy of Sciences. Permission is granted for this material to be shared for noncommercial, educational purposes, provided that this notice appears on the reproduced materials, the Web address of the online, full authoritative version is retained, and copies are not altered. To disseminate otherwise or to republish requires written permission from the National Academies Press.

THE NATIONAL ACADEMIES
Advisers to the Nation on Science, Engineering, and Medicine

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

SUMMARY OF A WORKSHOP ON

SOFTWARE-INTENSIVE SYSTEMS AND UNCERTAINTY AT SCALE

Joan D. Winston and Lynette I. Millett, Editors

Committee on Advancing Software-Intensive Systems Producibility

Computer Science and Telecommunications Board

Division on Engineering and Physical Sciences

NATIONAL RESEARCH COUNCIL
OF THE NATIONAL ACADEMIES

THE NATIONAL ACADEMIES PRESS
Washington, D.C.
www.nap.edu

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

THE NATIONAL ACADEMIES PRESS 500 Fifth Street, N.W. Washington, DC 20001

NOTICE: The project that is the subject of this report was approved by the Governing Board of the National Research Council, whose members are drawn from the councils of the National Academy of Sciences, the National Academy of Engineering, and the Institute of Medicine. The members of the committee responsible for the report were chosen for their special competences and with regard for appropriate balance.

Support for this project was provided by the Office of the Secretary of Defense, Department of Defense, with assistance from the National Science Foundation under sponsor award number CNS-0541636 and by the Office of Naval Research under sponsor award number N00014-04-1-0736. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the agencies and organizations that provided support for the project.

International Standard Book Number-13: 978-0-309-10844-7
International Standard Book Number-10: 0-309-10844-6

Additional copies of this report are available from

The National Academies Press
500 Fifth Street, N.W., Lockbox 285
Washington, DC 20055
800/624-6242
202/334-3313 (in the Washington metropolitan area)
http://www.nap.edu

Copyright 2007 by the National Academy of Sciences. All rights reserved.

Printed in the United States of America

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

THE NATIONAL ACADEMIES

Advisers to the Nation on Science, Engineering, and Medicine

The **National Academy of Sciences** is a private, nonprofit, self-perpetuating society of distinguished scholars engaged in scientific and engineering research, dedicated to the furtherance of science and technology and to their use for the general welfare. Upon the authority of the charter granted to it by the Congress in 1863, the Academy has a mandate that requires it to advise the federal government on scientific and technical matters. Dr. Ralph J. Cicerone is president of the National Academy of Sciences.

The **National Academy of Engineering** was established in 1964, under the charter of the National Academy of Sciences, as a parallel organization of outstanding engineers. It is autonomous in its administration and in the selection of its members, sharing with the National Academy of Sciences the responsibility for advising the federal government. The National Academy of Engineering also sponsors engineering programs aimed at meeting national needs, encourages education and research, and recognizes the superior achievements of engineers. Dr. Charles M. Vest is president of the National Academy of Engineering.

The **Institute of Medicine** was established in 1970 by the National Academy of Sciences to secure the services of eminent members of appropriate professions in the examination of policy matters pertaining to the health of the public. The Institute acts under the responsibility given to the National Academy of Sciences by its congressional charter to be an adviser to the federal government and, upon its own initiative, to identify issues of medical care, research, and education. Dr. Harvey V. Fineberg is president of the Institute of Medicine.

The **National Research Council** was organized by the National Academy of Sciences in 1916 to associate the broad community of science and technology with the Academy's purposes of furthering knowledge and advising the federal government. Functioning in accordance with general policies determined by the Academy, the Council has become the principal operating agency of both the National Academy of Sciences and the National Academy of Engineering in providing services to the government, the public, and the scientific and engineering communities. The Council is administered jointly by both Academies and the Institute of Medicine. Dr. Ralph J. Cicerone and Dr. Charles M. Vest are chair and vice chair, respectively, of the National Research Council.

www.national-academies.org

Copyright © National Academy of Sciences. All rights reserved.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

COMMITTEE ON ADVANCING SOFTWARE-INTENSIVE SYSTEMS PRODUCIBILITY

WILLIAM L. SCHERLIS, Carnegie Mellon University, *Chair*
ROBERT F. BEHLER, The MITRE Corporation
BARRY W. BOEHM, University of Southern California
LORI A. CLARKE, University of Massachusetts, Amherst
MICHAEL A. CUSUMANO, Massachusetts Institute of Technology
MARY ANN DAVIDSON, Oracle Corporation
LARRY DRUFFEL, Independent Consultant
RUSSELL FREW, Lockheed Martin
JAMES LARUS, Microsoft Corporation
GREG MORRISETT, Harvard University
WALKER ROYCE, IBM
DOUGLAS C. SCHMIDT, Vanderbilt University
JOHN P. STENBIT, Independent Consultant
KEVIN J. SULLIVAN, University of Virginia

Staff

LYNETTE I. MILLETT, Study Director and Senior Program Officer
JOAN D. WINSTON, Program Officer
MARGARET MARSH HUYNH, Senior Program Assistant

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

COMPUTER SCIENCE AND TELECOMMUNICATIONS BOARD

JOSEPH F. TRAUB, Columbia University, *Chair*
ERIC BENHAMOU, Benhamou Global Ventures, LLC
FREDERICK R. CHANG, University of Texas, Austin
WILLIAM DALLY, Stanford University
MARK E. DEAN, IBM Almaden Research Center
DEBORAH ESTRIN, University of California, Los Angeles
JOAN FEIGENBAUM, Yale University
KEVIN KAHN, Intel Corporation
JAMES KAJIYA, Microsoft Corporation
MICHAEL KATZ, University of California, Berkeley
RANDY H. KATZ, University of California, Berkeley
SARA KIESLER, Carnegie Mellon University
TERESA H. MENG, Stanford University
PRABHAKAR RAGHAVAN, Yahoo! Research
FRED B. SCHNEIDER, Cornell University
ALFRED Z. SPECTOR, Independent Consultant, Pelham, New York
WILLIAM STEAD, Vanderbilt University
ANDREW J. VITERBI, Viterbi Group, LLC
PETER WEINBERGER, Google, Inc.

Staff

JON EISENBERG, Director
KRISTEN BATCH, Associate Program Officer
RADHIKA CHARI, Administrative Coordinator
RENEE HAWKINS, Financial Associate
MARGARET MARSH HUYNH, Senior Program Assistant
HERBERT S. LIN, Senior Scientist
LYNETTE I. MILLETT, Senior Program Officer
DAVID PADGHAM, Associate Program Officer
JANICE M. SABUDA, Senior Program Assistant
TED SCHMITT, Consultant
BRANDYE WILLIAMS, Program Assistant
JOAN D. WINSTON, Program Officer

For more information on CSTB, see its Web site at <http://www.cstb.org>, write to CSTB, National Research Council, 500 Fifth Street, N.W., Washington, DC 20001, call (202) 334-2605, or e-mail CSTB at cstb@nas.edu.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Preface

Pursuant to a request by the Department of Defense, the National Research Council (NRC) convened a study committee under the auspices of the Computer Science and Telecommunications Board (CSTB) to assess the nature of the U.S. national investment in software research and, in particular, to consider ways to enhance the knowledge and human resource base needed to design, produce, and employ software-intensive systems for tomorrow's weapons and operations systems. Many organizations are facing the combination of increasing system scale and increasing complexity of software-intensive systems. However, the compelling need to interconnect them to realize DoD's vision of "net-centric warfare" exacerbates the challenges of uncertainty at scale for DoD.

Several recent reports that highlight these challenges[1] suggest that the scale and complexity of software-intensive systems introduce fundamental new challenges and require augmentation of existing approaches by software practices and technologies that more explicitly address these challenges. Challenges of uncertainty and scale are faced in large-scale enterprise systems of all kinds but are particularly demanding in defense systems owing to the relative lack of precedent in both requirements and engineering designs and also to the need for high levels of quality, secu-

[1]See, for example, Software Engineering Institute (2006), *Ultra-large Scale Systems: The Software Challenge of the Future*, which noted that current abstractions fail for the levels of complexity that systems require today. Also see Defense Science Board (2000), *Task Force on Defense Software Report*, which noted, among other things, that strengthening the technology base to rapidly adapt to fluid circumstances is important and that the complexity of DoD software applications is increasing more than linearly.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

rity, and safety in an environment with well-resourced adversaries. This suggests that defense, while sharing many particular kinds of requirements with large-scale enterprises and infrastructures, is nonetheless a demand leader with respect to many of these requirements, outpacing most enterprise and commercial projects.

As part of its study, this committee organized a public workshop on January 17, 2007, to examine *uncertainty at scale* in current and future software-intensive systems. Workshop sessions examined the challenges related to engineering uncertainty, system complexity, and scale from a range of perspectives. Session speakers were given roughly 25 minutes to provide their views on issues identified in the workshop agenda (see Appendix A). There was substantial discussion and interaction among the session speakers and moderators, the committee, and others present.

The purpose of the workshop was to inform the committee as it conducts its study. This report summarizes the workshop discussions, including speaker presentations and discussions with committee members and others present. It is not a compilation of quotations from particular individuals nor is it a complete synthesis of conclusions or results. Although the summary was prepared by the committee based on presentations and discussion at the workshop, the comments do not necessarily reflect the views of the committee nor do they represent findings and recommendations of the NRC. Moreover, the summary points for the sessions are a digest of both presentations and discussion. They should not be taken as remarks made solely by the scheduled session speakers because the discussions included remarks by the others in attendance.

The committee's broader consideration of advancing software-intensive system producibility will appear in its final report, to be issued near the end of the study. That report will provide recommendations to the responsible agency, the executive branch, and legislative officials—and to the broader software community—about how to improve software development and achieve future goals.

The committee thanks all the workshop participants for their thoughtful presentations and discussion. It also thanks the Computer Science and Telecommunications Board staff, particularly study director Lynette Millett and program officer Joan Winston, who have ably managed this project, coordinated the team, and contributed greatly to the development of this report, and to Margaret Huynh, who has facilitated our meetings and other project activities.

William Scherlis, *Chair*
Committee on Advancing
Software-Intensive Systems Producibility

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Acknowledgment of Reviewers

This report has been reviewed in draft form by individuals chosen for their diverse perspectives and technical expertise, in accordance with procedures approved by the National Research Council's (NRC's) Report Review Committee. The purpose of this independent review is to provide candid and critical comments that will assist the institution in making its published report as sound as possible and to ensure that the report meets institutional standards for objectivity, evidence, and responsiveness to the study charge. The review comments and draft manuscript remain confidential to protect the integrity of the deliberative process. We wish to thank the following individuals for their review of this report:

David Notkin, University of Washington,
Alfred Spector, Independent Consultant,
John Vu, Boeing Corporation,
Peter Weinberger, Google, Inc., and
Jeannette Wing, Carnegie Mellon University.

Although the reviewers listed above have provided many constructive comments and suggestions, they were not asked to endorse the conclusions or recommendations, nor did they see the final draft of the report before its release. The review of this report was coordinated by Susan Graham of the University of California, Berkeley. Appointed by the NRC,

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

she was responsible for making certain that an independent examination of this report was carried out in accordance with institutional procedures and that all review comments were carefully considered. Responsibility for the final content of this report rests entirely with the authoring committee and the institution.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Contents

Copyright © National Academy of Sciences. All rights reserved.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

1

Introduction and Overview

This report summarizes the workshop on uncertainty at scale in the context of software-intensive systems producibility, held January 17, 2007, in Washington, D.C., under the auspices of the National Research Council's (NRC's) Committee on Advancing Software-Intensive Systems Producibility. The workshop was structured to gather inputs and insights from the commercial and military software and system development communities looking at the current and future challenges that surround engineering uncertainty, system complexity, and scale. The purpose of the workshop was to inform the committee as it conducts its study and to stimulate discussion of these issues in the community. This workshop summary, which does not contain findings or recommendations from the committee, is presented in the spirit of continuing that discussion.

As they develop unprecedented systems, software engineers must address many kinds of uncertainty in the course of requirements assimilation, architecture, design, development, deployment, and ongoing use of large-scale software systems. For defense systems, the operating environment, coalition-driven requirements, adversary capabilities, security and safety requirements, and so on also pose significant uncertainty. The increasing scale and complexity of systems, along with the compelling need to interconnect them, make the challenges of uncertainty at scale increasingly formidable and of particular significance for the Department of Defense (DoD). Exacerbating the situation, requirements for DoD software-intensive systems are emerging and/or not static, and the DoD has not enough internal software engineering expertise. In the context of

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

this workshop, "scale" refers not just to gross scale (e.g., the numbers of instructions or lines of code), but also to the extent of complexity, interconnection, and interdependence. "Uncertainty" refers to the full range of engineering risks, including the rates at which requirements and features and the operating and technical environments are changing. The briefings for this workshop examined these and other challenges regarding engineering uncertainty, system complexity, and scale from a range of perspectives.

Several recent reports that highlight these challenges[1] suggest that the scale and complexity of software-intensive systems introduce fundamental new challenges and require augmentation of existing approaches by software practices and technologies that more explicitly address the challenges. Challenges of uncertainty and scale are faced in large-scale enterprise systems of all kinds but are particularly demanding in defense systems. Technical factors contributing to heightened challenges include the relative lack of precedent in both requirements and engineering designs, as well as the need for high levels of quality, security, and safety in an environment with well-resourced adversaries. Institutional factors include complexities resulting from the particulars of DoD management and procurement. As a result, defense requirements, although having some commonality with those of other large-scale enterprises and infrastructures, may outpace those of most enterprise and commercial projects.

Workshop participants explored the various dimensions of these challenges, focusing particularly on software engineering and the management of uncertainties in requirements, operating environment, and implementation. They were invited to examine the range of problems inherent in building large-scale systems and to explore both the current state of software engineering knowledge regarding potential incremental solutions to problems of scale as well as areas where fundamental research is needed to bridge the gap between current practice and the revolutionary challenges offered by future systems. Discussions at the workshop took place with an eye to emerging defense needs but in a way that was receptive to lessons to be gleaned from commercial and enterprise-level efforts. Case studies and examples provided at the workshop were explored to assess promising ideas and directions and to identify the fundamental research problems that remain.

Questions posed to participants included these: Are there precedented

[1]See, for example, Software Engineering Institute (2006), *Ultra-large Scale Systems: The Software Challenge of the Future*, which noted that current abstractions fail for the levels of complexity that systems require today. Also see Defense Science Board (2000), *Task Force on Defense Software Report*, which noted, among other things, that strengthening the technology base to rapidly adapt to fluid circumstances is important and that the complexity of DoD software applications is increasing more than linearly.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

architectural concepts from existing systems that can provide a basis for making decisions about the architecture for systems where constant evolution and change are the norm? Which best and emerging practices extant today are likely to be beneficial? Which technological approaches would support large-scale decentralized design processes? Which specific existing approaches show the most potential? What problems will industry likely solve on its own? What problems require the stimulus of R&D investment? Participants were also asked to consider areas of research that are likely to yield fruitful outcomes and how DoD, in particular, can both stimulate and assimilate innovative ideas, whether they be advanced practices from industry or ideas from the research laboratory.

Each succeeding section describes some of the main themes arising from a workshop session. The themes are not conclusions or findings of the committee; they are ideas extracted from the discussions during each session, drawn not only from the presentations of the speakers but also from the discussions among all the participants (committee, speakers, and attendees) that seem to form the gist of the session. Several items should be kept in mind when reading this report. The workshop focused on a particular subset of areas that the committee believed would provide a basis for its work during the next year. The committee plans to gather input on other topics, including through another public workshop; feedback and additional input from readers of this report are welcome. Also, in the interests of timely dissemination, the committee elected to defer elaboration and analysis of the workshop discussion and instead to offer here a more succinct summary focused on reporting the issues discussed. Accordingly, this report does not provide a freestanding overview of the current state of challenges in software development or large-scale systems producibility. Moreover, for readability and to promote understanding, background material on some of the topics raised has been interspersed with the record of presentations and discussions. Presenters' views varied, depending on their own experiences and expertise; some speakers provided detailed information while others offered higher-level, more abstract presentations. For these reasons and because some of the discussion amounted to brainstorming, this summary may contain internal inconsistencies that reflect the wide range of views offered by the speakers and other participants.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

2

Summary of Workshop Discussions

SESSION 1: PROCESS, ARCHITECTURE, AND THE GRAND SCALE

Panelists: John Vu, Boeing, and Rick Selby, Northrop Grumman Corporation
Moderator: Michael Cusumano

Panelist presentations and general discussions at this session were intended to explore the following questions from the perspectives of software development for government and commercial aerospace systems:

- What are the characteristics of successful approaches to architecture and design for large-scale systems and families of systems?
- Which architecture ideas can apply when systems must evolve rapidly?
- What kinds of management and measurement approaches could guide program managers and developers?

Synergies Across Software Technologies and Business Practices Enable Successful Large-Scale Systems

Context matters in trying to determine the characteristics of successful approaches—different customer relationships, goals and needs, pacing of projects, and degree of precedent all require different practices. For example, different best practices may apply depending on what sort of

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

system or application is under development. Examples discussed include commercial software products, IT and Internet financial services, airplanes, and government aerospace systems.

- *Different systems and software engineering organizations have different customers and strategies.* They may produce a variety of deliverables, such as a piece of software, an integrated hardware-software environment, or very large, complicated, interconnected, hardware-software networked systems.
- *Different systems and software engineering organizations have different goals and needs.* Product purposes vary—user empowerment, business operations, and mission capabilities. Projects can last from a month to 10 or 12 years. The project team can be one person or literally thousands. The customer agreement can be a license, service-level agreement, or contract. There can be a million customers or just one—for example, the government. The managerial focus can be on features and time to market; cycle time, workflow, and uptime; or reliability, contract milestones, and interdependencies; and so on.
- *While some best practices, such as requirements and design reviews and incremental and spiral life cycles, are broadly applicable, specific practice usually varies.* Although risk management is broadly applicable, commercial, financial, and government system developers may adopt different kinds of risk management. While government aerospace systems developers may spend months or years doing extensive system modeling, this may not be possible in other organizations or for other types of products. Commercial software organizations may focus on daily builds (that is, each day compiling and possibly testing the entire program or system incorporating any new changes or fixes); for aerospace systems, the focus may be on weekly or 60-day builds. Other generally applicable best practices that vary by market and organization include parallel small teams, design reuse, domain-specific languages, opportunity management, trade-off studies, and portability layers. These differences are driven by the different kinds of risks that drive engineering decisions in these sectors.
- *Government aerospace systems developers, along with other very large software-development enterprises, employ some distinctive best practices.* These include independent testing teams and, for some aspects of the systems under consideration, deterministic, simple designs. These practices are driven by a combination of engineering, risk-management, and contractual considerations.

In a very large[1] organization, synergy across software technologies and business practices can contribute to success. Participants explored

[1]Very large in this case means over 100,000 employees throughout a supply chain doing systems engineering, systems development, and systems management; managing multiple product lines; and building systems with millions of lines of code.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

the particular case of moderately precedented systems[2] and major components with control-loop architectures. For systems of this kind there are technology and business practice synergies that have worked well. Here are some examples noted by speakers:

- *Decomposition of large systems to manage risk.* With projects that typically take between 6 and as many as 24 months to deliver, incremental decomposition of the system can reduce risk, provide better visibility into the process, and deliver capability over time. Decomposition accelerates system integration and testing.
- *Table-based design, oriented to a system engineering view in terms of states and transitions, both nominal and off-nominal.* This enables the use of clear, table-driven approaches to address nominal modes, failure modes, transition phases, and different operations at different parts of the system operations.
- *Use of built-in, domain-specific (macro) languages in a layered architecture.* The built-in, command-sequencing macro language defines table-driven executable specifications. This permits a relatively stable infrastructure and a run-time system with low-level, highly deterministic designs yet extensible functionality. It also allows automated testing of the systems.
- *Use of precedented and well-defined architectures for the task management structure that incorporates a simple task structure, deterministic processing, and predictable timelines.* For example, a typical three-task management structure might have high-rate (32 ms) tasks, minor-cycle (128 ms) tasks, and background tasks. The minor cycle reads and executes commands, formats telemetry, handles fault protection, and so forth. The high-rate cycle handles message traffic between the processors. The background cycle adds capability that takes a longer processing time. This is a reusable processing architecture that has been used for over 30 years in spacecraft and is aimed at the construction of highly reliable, deterministic systems.
- *Gaining advantages from lack of fault proneness in reused components by achieving high levels of code, design, and requirement reuse.* One example of code reuse was this: Across 25 NASA ground systems, 32 percent of software components were either reused or modified from previous systems (for spacecraft, reuse was said to be as high as 80 percent). Designs and requirements can also be reused. Typically, there is a large backward

[2]Precedent refers to the extent to which we have experience with systems of a similar kind. More specifically, there can be precedent with respect to requirements, architecture and design, infrastructure choices, and so on. Building on precedent leads to routinization, reduced engineering risk, and better predictability (lower variance) of engineering outcomes.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

compatibility set of requirements, and these requirements can be reused. Requirements reuse is very common and very successful even though the design and implementation might each be achieved differently. Design reuse might involve allocation of function across processors in terms of how particular algorithms are structured and implemented. The functions might be implemented differently in the new system, for example, in components rather than custom code or in different programming languages. This is an example of true design reuse rather than code reuse.

In addition to these synergies, it was suggested that other types of analyses could also contribute to successful projects. Data-driven statistical analyses can help to identify trends, outliers, and process improvements to reduce or mitigate defects. For example, higher rates of component interactions tend to be correlated with more faults, as well as more fault-correction effort. Risk analyses prioritize risks according to cost, schedule, and technical safety impacts. Charts that show project risk mitigation over time and desired milestones help to define specific tasks and completion criteria. It was suggested that each individual risk almost becomes a microcosm of its own project, with schedules and milestones and progressive mitigation of that risk to add value.

One approach to addressing the challenge of scale is to divide and conquer. Of course, arriving at an architectural design that supports decomposition is a prerequisite for this approach, which can apply across many kinds of systems development efforts. Suggestions included the following:

- *Divide the organization into parallel teams.* Divide very large 1,000-person teams into parallel teams; establish a project rhythm of design cycles and incremental releases. This division of effort is often based on a system architectural structure that supports separate development paths (an example of what is known as Conway's law—that software system structures tend to reflect the structures of the organizations they are developed by). Indeed, without agreement on architectural fundamentals—the key internal interfaces and invariants in a system—division of effort can be a risky step.
- *Innovate and synchronize.* Bring the parallel teams together, whether the task is a compilation or a component delivery and interface integration. Then stabilize, usually through some testing and usage period.
- *Encourage coarse-grain reuse.* There is a lot of focus on very fine-grain reuse, which tends to involve details about interfaces and dependencies; there is also significant coarse-grain opportunity to bring together both legacy systems and new systems. A coarse-grain approach makes possible the accommodation of systems at different levels of maturity.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Examples of success in coarse-grain reuse are major system frameworks (such as e-commerce frameworks), service-based architectures, and layered architectures.

- *Automate.* Automation is needed in the build process, in testing, and in metrics.

Uncertainty is inherent in the development of software-intensive systems and must be reassessed constantly, because there are always unknowns and incomplete information. Waiting for complete information is costly, and it can take significant time to acquire the information—if it is possible to acquire it at all. Schedules and budgets are always limited and almost never sufficient. The goal, it was argued, should be to work effectively and efficiently within the resources that are available and discharge risks in an order that is appropriate to the goals of the system and the nature of its operating environment: Establish the baseline design, apply systematic risk management, and then apply opportunity management, constantly evaluating the steps needed and making decisions about how to implement them. Thus, it was suggested that appropriate incentives and analogous penalty mechanisms at the individual level and at the organization or supplier level can change behavior quickly. The goal is thus for the incentive structure to create an opportunity to achieve very efficient balance through a "self-managing organization." In a self-managing organization, it was suggested, the leader has the vision and is an evangelist rather than a micromanager, allowing others to manage using systematic incentive structures.

Some ways to enable software technology and business practices for large-scale systems were suggested:

- Creating strategies, architectures, and techniques for the development and management of systems and software, taking into account multiple customers and markets and a broad spectrum of projects, from small scale through large.
- Disseminating validated experiences and data for best practices and the circumstances when they apply (for example, titles like "Case Studies of Model Projects").
- Aligning big V waterfall-like systems engineering life-cycle models with incremental/spiral software engineering life-cycle models.[3]

[3]The V model is a V-shaped, graphical representation of the systems development life cycle that defines the results to be achieved in a project, describes approaches for developing these results, and links early stages (on the left side of the V) with evaluation and outcomes (on the right side). For example, requirements link to operational testing and detailed design links to unit testing.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

- Facilitating objective interoperability mechanisms and benchmarks for enabling information exchange.
- Lowering entry barriers for research groups and nontraditional suppliers to participate in large-scale system projects (Grand Challenges, etc.).
- Encouraging advanced degree programs in systems and software engineering.
- Defining research and technology roadmaps for systems and software engineering.
- Collaborating with foreign software developers.

Process, Architecture, and Very Large-Scale Systems

Remarks during this portion of the session were aimed at thinking outside the box about what the state of the art in architectures might look like in the future for very large-scale, complex systems that exhibit unpredictable behavior. The primary context under discussion was large-scale commercial aircraft development—the Boeing 777 has a few million lines of code, for example, and the new 787 has several million and climbing.

It was argued that very large-scale, highly complex systems and families of systems require new thinking, new approaches, and new processes to architect, design, build, acquire, and operate. It was noted that these new systems are going from millions of lines of code to tens of millions of lines of code (perhaps in 10 years to billions of lines of code and beyond); from hundreds of platforms (servers) to thousands, all interconnected by heterogeneous networks; from hundreds of vendors (and subcontractors) to thousands, all providing code; and from a well-defined user community to dynamic communities of interdependent users in changing environments. It was suggested that the issue for the future—10 or 20 years from now—is how to deal with the potential billion lines of code and tens of thousands of vendors in the very diverse, open-architecture-environment global products of the future, assembled from around the world. According to the forward-looking vision presented by speakers, these systems may have the following characteristics:

- Very large-scale systems would integrate multiple systems, each of them autonomous, having distinctive characteristics, and performing its own functions independently to meet distinct objectives.
- Each system would have some degree of intelligence, with the objectives of enabling it to modify its relationship to other component systems (in terms of functionality) and allowing it to respond to changes, perhaps unforeseen, in the environment. When multiple systems are joined

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

together, the significant emergent capabilities of the resulting system as a whole would enable common goals and objectives.

- Each very large-scale system would share information among the various systems in order to address more complex problems.
- As more systems are integrated into a very large-scale system, the channels connecting all systems through which information flows would become more robust and continue to grow and expand throughout the life cycle of the very large-scale system.

It was argued that a key benefit of a very large-scale system is the interoperability between operational systems that allows decision makers to make better, more informed decisions more quickly and accurately. From a strategic perspective, a very large-scale system is an environment where operational systems have the ability to share information among geographically distributed systems with appropriate security and to act on the shared information to achieve their desired business goals and objectives. From an operational perspective, a very large-scale system is an environment where each operational subsystem performs its own functions autonomously, consistent with the overall strategy of the very large-scale system.

The notion of continuous builds or continuous integration was also discussed. Software approaches that depend on continuous integration—that is, where changes are integrated very frequently—require processes for change management and integration management. These processes are incremental and build continuously from the bottom up to support evolution and integration, instead of from the top down, using a plan-driven, structured design. They separate data and functions for faster updates and better security. To implement these processes, decentralized organizations and an evolving concept of operations are required to adapt quickly to changing environments.

The overall architectural framework for large-scale systems described by some participants in this session consists of five elements:

- *Governance.* These describe the rules, regulations, and change management that control the total system.
- *Operational.* These describe how each operational system can be assembled from preexisting or new components (building blocks) to operate in their own new environment so they can adapt to change.
- *Interaction*. These describe the communication (information pipeline) and interaction between operational systems that may affect the very large system and how the very large system will react to the inputs from the operational systems.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

- *Integration and change management.* These describe the processes for managing change and the integration of systems that enable emergent capabilities.
- *Technical.* These depict the technology components that are necessary to support these systems.

It was suggested that large-scale systems of that future that will cope with scale and uncertainty would be built from the bottom up by starting with autonomous building blocks to enable the rapid assembly and integration of these components to effectively evolve the very large-scale system. The architectural framework would ensure that each building block would be aligned to the total system. Building blocks would be assembled by analyzing a problem domain through the lens of an operational environment or mission for the purpose of creating the characteristics and functionality that would satisfy the stakeholders' requirements. In this mission-focused approach, all stakeholders and modes of operations should be clearly identified; different user viewpoints and needs should be gathered for the proposed system; and stakeholders must state which needs are essential, which are desirable, and which are optional. Prioritization of stakeholders' needs is the basis for the development of such systems; vague and conflicting needs, wants, and opinions should be clarified and resolved; and consensus should be built before assembling the system.

At the operational level, the system would be separated from current rigid organization structures (people, processes, technology) and would evolve into a dynamic concept of operation by assembling separate building blocks to meet operational goals. The system manager should ask: What problem will the system solve? What is the proposed system used for? Should the existing system be improved and updated by adding more functionality or should it be replaced? What is the business case? To realize this future, participants suggested that research is needed in several areas, including these:

- Governance (rules and regulations for evolving systems).
- Interaction and communication among systems (including the possibility of negative interactions between individual components and the integrity, security, and functioning of the rest of the system).
- Integration and change management.
- User's perspective and user-controlled evolution.
- Technologies supporting evolution.
- Management and acquisition processes.
- An architectural structure that enables emergence.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

- Processes for decentralized organizations structured to meet operational goals.

SESSION 2: DOD SOFTWARE CHALLENGES FOR FUTURE SYSTEMS

Panelists: Kristen Baldwin, Office of the Under Secretary of Defense for Acquisitions, Technology and Logistics, and Patrick Lardieri, Lockheed Martin
Moderator: Douglas Schmidt

Panelist presentations and general discussions during this session were intended to explore two questions, from two perspectives: that of the government and that of the government contractor:

- How are challenges for software in DoD systems, particularly cyber-physical systems, being met in the current environment?
- What advancements in R&D, technology, and practices are needed to achieve success as demands on software-intensive system development capability increase, particularly with respect to scale, complexity, and the increasingly rapid evolution in requirements (and threats)?

DoD Software Engineering and System Assurance

An overview of various activities relating to DoD software engineering was given. Highlights from the presentation and discussion follow. The recent Acquisition & Technology reorganization is aimed at positioning systems engineering within the DoD, consistent with a renewed emphasis on software. The director of Systems and Software Engineering now reports directly to the Under Secretary of Defense for Acquisition and Technology. The mission of Systems and Software Engineering, which addresses evolving system—and software—engineering challenges, is as follows:

- Shape acquisition solutions and promote early technical planning.
- Promote the application of sound systems and software engineering, developmental test and evaluation, operational test and evaluation to determine operational suitability and effectiveness, and related technical disciplines across DoD's acquisition community and programs.
- Raise awareness of the importance of effective systems engineering and raise program planning and execution to the state of the practice.
- Establish policy, guidance, best practices, education, and training in collaboration with the academic, industrial, and government communities.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

• Provide technical insight to program managers and leadership to support decision making.

DoD's Software Center of Excellence is made up of a community of participants including industry, DoD-wide partnerships, national partnerships, and university and international alliances. It will focus on supporting acquisition; improving the state of the practice of software engineering; providing leadership, outreach, and advocacy for the systems engineering communities; and fostering resources that can meet DoD goals. These are elements of DoD's strategy for software, which aims to promote world-class leadership for DoD software engineering.

Findings from some 40 recent program reviews were discussed. These reviews identified seven systemic issues and issue clusters that had contributed to DoD's poor execution of its software program, which were highlighted in the session discussion. The first issue is that software requirements are not well defined, traceable, and testable. A second issue cluster involves immature architectures; integration of commercial-off-the-shelf (COTS) products; interoperability; and obsolescence (the need to refresh electronics and hardware). The third cluster involves software development processes that are not institutionalized, have missing or incomplete planning documents, and inconsistent reuse strategies. A fourth issue is software testing and evaluation that lacks rigor and breadth. The fifth issue is lack of realism in compressed or overlapping schedules. The sixth issue is that lessons learned are not incorporated into successive builds—they are not cumulative. Finally, software risks and metrics are not well defined or well managed.

To address these issues, DoD is pursuing an approach that includes the following elements:

• *Identification of software issues and needs through a software industrial base assessment,*[4] *a National Defense Industrial Association (NDIA) workshop on top software issues, and a defense software strategy summit.* The industrial base assessment, performed by CSIS, found that the lack of comprehensive, accurate, timely, and comparable data about software projects within DoD limits the ability to undertake any bottom-up analysis or enterprise-wide assessments about the demand for software. Although the CSIS analysis suggests that the overall pool of software developers is adequate, the CSIS assessment found an imbalance in the supply of and demand for the specialized, upper echelons of software developer/management cadres. These senior cadres can be grown, but it takes time (10 or more years) and

[4]Center for Strategic and International Studies (CSIS), Defense-Industrial Initiatives Group, 2006. *Software Industrial Base Assessment: Phase I Report*, October 4.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

a concerted strategy. In the meantime, management/architecture/systems engineering tools might help improve the effectiveness of the senior cadres. Defense business system/COTS software modification also places stress on limited pools of key technical and management talent. Moreover, the true cost and risk of software maintenance deferral are not fully understood.

- *Creation of opportunities and partnerships through an established network of government software points of contact; chartering the NDIA Software Committee; information exchanges with government, academia, and industry, and planning a systems and software technology conference.* Top issues emerging from the NDIA Defense Software Strategy Summit in October 2006 included establishment and management of software requirements, the lack of a software voice in key system decisions, inadequate life-cycle planning and management, the high cost and ineffectiveness of traditional software verification methods, the dearth of software management expertise, inadequate technology and methods for assurance, and the need for better techniques for COTS assessment and integration.
- *Execution of focused initiatives such as Capability Maturity Model Integration (CMMI) support for integrity and acquisition, a CMMI guidebook, a handbook on engineering for system assurance, a systems engineering guide for systems of systems (SoSs), the provision of software support to acquisition programs, and a vision for acquisition reform.* SoSs to be used for defense require special considerations for scale (a single integrated architecture is not feasible), ownership and management (individual systems may have different owners), legacy (given budget considerations, current systems will be around for a long time), changing operations (SoS configurations will face changing and unpredictable operational demands), criticality (systems are integrated via software), and role of the network (SoSs will be network-based, but budget and legacy challenges may make implementation uneven). To address a complex SoS, an initial (incremental) version of the DoD's SoS systems engineering guide is being piloted; future versions will address enterprise and net-centric considerations, management, testing, and sustaining engineering.

The issue of system assurance—reducing the vulnerability of systems to malicious tampering or access—was noted as a fundamental consideration, to the point that cybertrust considerations can be a fundamental driver of requirements, architecture and design, implementation practice, and quality assurance.[5] Because current assurance, safety, and protection

[5]A separate National Research Council study committee is exploring the issue of cybersecurity research and development broadly, and its report, *Toward a Safer and More Secure Cyberspace,* will be published in final form in late 2007. See http://cstb.org/project_cybersecurity for more information.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

initiatives are not aligned, a comprehensive strategy for system assurance initiatives is being developed, including standards activities and guidance to put new methods into practice.

One additional challenge for DoD is that, given its shortage of software resources and critical dependence on software, it cannot afford to have stovepipes in its community. To that end, the DoD Software Center of Excellence is intended to be a focal point for the community. Areas to be explored include, for example, agile methods, software estimation (a harder problem to address for unprecedented systems than for precedented ones), and software testing.

Challenges in Developing DoD Cyber-physical Systems

This session explored challenges in building cyber-physical systems for DoD—systems that integrate physical processes and computer processes in a real-time distributed fashion—from the perspective of a large contractor responsible for a wide range of systems and IT services.

Cyber-physical systems are increasingly systems- and software-intensive—for example, in 1960, only 8 percent of the F-4 fighter capability was provided by software; in 2000, 85 percent of the F-22 fighter capability was provided by software. Such systems are distributed, real-time systems with millions of lines of code, driven by multiple sensors reporting at a variety of timescales and by multiple weapon system and machinery control protocols. Current examples of cyber-physical systems include the Joint Strike Fighter (JSF) and Future Combat Systems (FCS); examples of forthcoming technologies would be teams of autonomous robots or teams of small, fast surface ships. Characteristics of these systems exemplify the challenges of uncertainty and scale; they include

- Large scale—tens of thousands of functional and performance requirements, 10 million lines of code, and 100 to 1,000 software configuration items;
- Simultaneous conflicting performance requirements—real-time processing, bounded failure recovery, security;
- Implementation diversity—programming languages, operating systems, middleware, complex deployment architectures, 20-40 year system life cycles, stringent certification standards; and
- Complex deployment architectures—systems of systems; mixed wired, wireless and satellite networks; multi-tiered servers; personal digital assistants (PDAs) and workstations; and multiple system configurations.

These systems are challenging, complex, and costly. Accordingly, system design challenges are frequently simplified by deferring or eliminat-

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

ing capability to bound costs and delivery dates. Nevertheless, cost overruns and schedule delays are common. The GAO reported that in fiscal year 2003 (FY03) the DoD spent $21 billion on research, development, testing, and evaluation (RDT&E) for new warfighting systems; about 40 percent of that may have been spent on reworking software to remedy quality-related issues.[6] For the F/A-22, the GAO reported that Air Force officials do not understand avionics software instability well enough to predict when they will be able to resolve its problems.[7] Because of the complex interrelationships between parts of these cyber-physical systems and the high degree of interactive complexity, piecewise deployment of partial systems is not helpful. An example given was a situation regarding the JSF, where changing an instruction memory layout to accommodate built-in test processing unexpectedly damaged the system's ability to meet timing requirements. It was suggested that as a result of experiences such as the one with the Aegis Combat System, where Aegis Baseline 6, Phase I, deployment was delayed for months because of integration problems between two independently designed cyber-physical systems, certi-

[6]Government Accountability Office (GAO), 2004, "Defense acquisitions: Stronger management practices are needed to improve DOD's software-intensive weapon acquisitions," Report to the Committee on Armed Services, U.S. Senate, GAO-04-393, March.

[7]Government Accountability Office (GAO), 2003, "Tactical aircraft: Status of the F/A-22," Statement of Alan Li, Director, Acquisition and Sourcing Management, Testimony Before the Subcommittee on Tactical Air and Land Forces, Committee on Armed Services, House of Representatives (GAO-03-603T), February. See also: Government Accountability Office (GAO), 2005, "Tactical aircraft: F/A-22 and JSF acquisition plans and implications for tactical aircraft modernization," Statement of Michael Sullivan, Director, Acquisition and Sourcing Management Issues, Testimony Before the Subcommittee on AirLand, Committee on Armed Services, U.S. Senate (GAO-05-519T), April 6, which concluded as follows:

> The original business case elements—needs and resources—set at the outset of the program are no longer valid, and a new business case is needed to justify future investments for aircraft quantities and modernization efforts. The F/A-22's acquisition approach was not knowledge-based or evolutionary. It attempted to develop revolutionary capability in a single step, causing significant technology and design uncertainties and, eventually, significant cost overruns and schedule delays;

and Government Accountability Office (GAO), 2007, "Tactical aircraft: DOD needs a joint and integrated investment strategy," Report to the Chairman, Subcommittee on Air and Land Forces, Committee on Armed Services, House of Representatives (GAO-07-415), April, which concluded as follows:

> We have previously recommended that DOD develop a new business case for the F-22A program before further investments in new aircraft or modernization are made. DOD has not concurred with this recommendation, stating that an internal study of tactical aircraft has justified the current quantities planned for the F-22A. Because of the frequently changing OSD-approved requirements for the F-22A, repeated cost overruns, significant remaining investments, and delays in the program we continue to believe a new business case is required and that the assumptions used in the internal OSD study be validated by an independent source.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

fication communities have become extremely conservative and require a static configuration for certification.[8]

Despite software's centrality and criticality in DoD cyber-physical systems and in warfighting in general, participants suggested that it is underemphasized in high-level management reviews. For example, the Quadrennial Defense Review calls for more complex systems for advanced warfighting capabilities but mentions software only twice.

Some inherent scientific and research challenges underlying engineering and engineering management of DoD cyber-physical systems cited by workshop participants include these:

- The management of knowledge fragmentation—fragmentation among people and teams, geographic areas, organizations, and temporal boundaries;
- Design challenges—the many problems that cannot be clearly defined without specifying the solution and for which every solution is a one-time-only operation (these are sometime referred to as "wicked problems"); and
- Team collaboration complexity—thousands of requirements, huge teams (hundreds or thousands of engineers), with frequent turnover and highly variable ranges of skill.

With respect to knowledge fragmentation (that is, knowledge split across individual minds, knowledge split across different phases of the development cycle and the life cycle, knowledge split across different artifacts, and knowledge split across various components of an organization), system engineering today is a concurrent top-down process. There is ad hoc coordination among engineers (domain engineers, system engineers, software engineers) at different levels and loose semantic coupling between design and specifications. There are some problems where it is difficult to say what to do without specifying how and thereby committing to an implementation; participants noted that current tools do not generally help manage the tremendous interdependence between the specification of the problem and the realization of a solution. Solutions are not necessarily right or wrong, and designers have to iterate rapidly, switching repeatedly between thinking about problem and solution concerns, along the lines of Fred Brooks' description of throw-away prototyping.[9] The process is slow, it is error prone because interaction is ad hoc,

[8]Note that the previous discussion noted the desirability of not having a static configuration in early stages of development.

[9]Frederick P. Brooks, 1995, *The Mythical Man-Month: Essays in Software Engineering*, Addison-Wesley Professional, New York.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

it uses imprecise English prose, and automated checking is relegated to the lowest level where formal specifications exist. Matters become even worse as the program or system grows in size and complexity.

Large teams managing complex systems must grapple with the issues of large scale in a complex collaborative environment. Interactive complexity has two dimensions: coupling (tight or loose) and interactions (complex or linear). Systems with high interactive complexity—for example, nuclear power plants and chemical plants—possess numerous hidden interactions that can lead to systems accidents and hazards. Interactive complexity can complicate reuse. Well-known cyber-physical system accidents cited by participants included the Ariane 5, which reportedly reused a module developed for Ariane 4. That module assumed that the horizontal velocity component would not overflow a 16-bit variable. This was true for Ariane 4 but not for Ariane 5, leading to self-destruction roughly 40 seconds after the launch.[10]

Cyber-physical systems typically have high interactive complexity. New systems have more resource sharing, which leads to hidden dependencies. There is limited design time support to understand or reduce interactive complexity. Modeling and analytic techniques are difficult to employ and often are underutilized. Simulations may not capture the system that is actually built; diagrams are not sufficient to convey all consequences of decisions. Thus, present cyber-physical systems rely on human ingenuity at design time and extensive system testing to manage interactive complexity. They also rely on particular knowledge of and experience with specific vendor-sourced components in the "technology stack." For this reason, the structure of the stack tends to resist change, impeding architectural progress and increased complexity in these systems. The resulting long and costly development efforts are expected to run into system accidents.

Elements of a research agenda for cyber-physical systems that perform predictably were discussed. One goal of such research would be to find ways to manage the uncertainty that arises from the highly-coupled nature and interactive complexity of system design at very large scale. Two areas were focused on during discussions at this session:

- *Platform technology*. One example of a platform technology would be generation of custom run-time infrastructures. Current run-time infrastructure is deployed in general-purpose layers that are not designed for specific applications. It is a significant challenge to configure controls

[10]J.C. Lyons, 1996, "Ariane 5 Flight 501 failure: Report by the inquiry board," Paris, July 19. Downloaded from http://www.ima.umn.edu/~arnold/disasters/ariane5rep.html on March 15, 2007.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

across the layers to achieve performance requirements; analysis is difficult because of many hidden dependencies and because of complex interfaces and capabilities. The generation of custom run-time infrastructure (e.g., WebSprocket) reduces system complexity. Another such technology would be certifiable dynamic resource management services. Current certification processes are based on extensive analysis and testing (hundreds of man-years) of fixed system configurations. Furthermore, these are human-intensive evaluation processes with limited technological support that occur over the design, development, and production lifecycle. There is no way to achieve the same level of assurance for untested system configurations that may be generated by an adaptive system in the run-time environment—and these are the kinds of systems that are likely to be deployed in the future.

- *System design tools.* Model-centric system design would allow evaluation of the design before final implementation by developing prototyping systems and using forms of static verification. Domain-specific modeling languages enable unambiguous system specifications. Model generation tools could be used to make models the center of the development process, synchronized with software artifacts. Tools that enable automated characterization of the behavior of third-party and COTS applications would be helpful. And, program transformation tools could be used to make the legacy code base and COTS software compatible with new platforms.

In addition to platform technologies and design tools, cultural elements are also needed to address the challenges of cyber-physical system development. Speakers noted some aspects of these elements:

- Independent, neutral-party benchmarking and evaluation—speakers believed that there is currently insufficient funding for this type of work.
- Development challenges that are realistic and at scale, allowing credible evaluation of technology solutions (measure technologies, not just artifacts).
- System design education as part of the undergraduate curriculum.

SESSION 3: AGILITY AT SCALE

Panelists: Kent Beck and Cynthia Andres, Three Rivers Institute
Moderator: Douglas Schmidt

This session addressed the application and applicability of extreme programming's "agile techniques" to very large, complex systems from

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

the perspectives of technology, development practices, and social psychology. For this session, both speakers and workshop participants were asked the following question:

- How can the engineering and management values that the "agile" community has identified be achieved for larger-scale systems and for systems that must evolve in response to rapidly changing requirements?

Values and Sponsorship

Participants noted that extreme programming (XP)—an agile software development methodology—was one of the first methodologies to be explicit about the value system behind its approach and about what is fundamental to this perspective on software development.[11] Different development approaches have their own underlying values. Speakers argued that over the 10 years or so since the coining of extreme programming, the key to success seems to be sponsorship—senior-level commitment to adopting XP ideas within an organization. Trying extreme programming can be disruptive, stirring up internal tension and controversy. Effective sponsors advocate among their peers and mitigate these effects. Senior-level sponsors also can help acquire resources to foster teamwork and communication.

When trying extreme programming, it was suggested that people tend to focus initially on the more visible and explicit changes to practices, such as pair programming, weekly releases, sitting together in open rooms, or using a test first approach. If a fundamental value shift is taking place, practices will change accordingly. However, under pressure, people tend to revert back to their old ways. Without support at higher levels for changes in approach and underlying values and without sustaining that support through periods of organizational discomfort during the transition, simply trying to put new or different practices in place is not very effective. One speaker noted that senior-level commitment and sponsorship are therefore key to changing values and conveying these changes to larger groups and organizations.

Human Issues in Software Development

One speaker noted that many of the challenges in software development are human issues: People are the developers and people write the

[11] For a brief summary of some of the underlying values in agile software development, see "The Agile Manifesto," available online at http://www.agilemanifesto.org/. These values include a focus on individuals over process, working software over documentation, and responsiveness to change over following a particular plan.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

software. Limitations to what can be done with software are often limitations of human imagination and how much complexity can be managed in one person's mind. Innovation requires fresh ideas, and if all parties are thinking similarly, not as many ideas are generated. Many problems have multiple solutions—a key is to sort out which solutions are sufficient and doable. It was suggested that one way to promote innovation is to encourage diversity: Small projects with diverse groups can be effective in that fostering interaction and coordination across disciplines often results in a stronger, richer set of ideas to choose from.

It was suggested that having interesting problems to work on is a nonmonetary motivator for many software and computer science practitioners. A good example of nonmonetary incentives is open source technology. Participants also noted that marketing innovation, intellectual curiosity, and creativity as an organizational goals are important. The perception or image of the work can be crucial to attracting new hires, who may know that the organization's work involves a lot of processes, requires great care, and takes a long time, but not necessarily that it is also interesting and creative work.

Trust, Communication, and Risk

Speakers argued that much of the effort in extreme programming comes down to finding better ways of building trust. Examples were given of ways to begin conversations and to putpeople in contact with one another in order to establish trust. These include the techniques of Appreciative Inquiry (talking about what works), World Café (acquire the collective knowledge, insight, and synergies of a group in a fairly short time), and Open Space (people talk about the concerns that they have and the issues that matter to them in breakout sessions whose highlights are reported to the rest of the group).[12] Some of the technical aspects of extreme programming are useful ways for programmers to demonstrate their trustworthiness. It was suggested that enhancing communication—in part, by using these communication techniques—could be useful to DoD. Tools to make developers' testing activities more visible, not just to themselves but also to teammates, contribute to accountability and transparency in development and increase communication as well.

There are technical things that can be done to reduce risk in projects. Some risk-reduction principles persist throughout all of extreme programming (XP). However, the bulk of the XP experience is not at the scale that

[12]See the Appreciative Inquiry Commons (http://appreciativeinquiry.case.edu/), the World Café (http://www.theworldcafe.com/), and Open Space (http://www.openspaceworld.org/).

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

DoD systems manifest. Therefore, one issue raised at the meeting was whether and how the agile programming/extreme programming experience can scale. Three risk reduction techniques were mentioned:

- *Reduce the amount of work that is half done.* Half-done work is an inherent risk. The feedback cycle has not been closed. No value has yet been received from the effort that has been expended; the mix of done and undone work occupies and distracts people. By gradually reducing the inventory of half-done work, a project can be made to run more smoothly with lower overall risk.
- *Find ways to defer the specification of requirement detail.* If a project experiences requirement churn, the chances are that too much detail has been specified too soon. There is a clear case to be made for much more carefully specifying the goals of a project up front but not the means for accomplishing those goals.
- *Testing sooner*. The longer a bug lives, the more expensive it becomes. One way of addressing that situation and improving the overall effectiveness of development is finding ways to validate software sooner, such as by developer testing. Integration is part of that testing.

Several research topics were discussed at this session:

- *Techniques for communication.* Examine how teams actually communicate and how they could communicate more effectively.[13]
- *Encouragement of multidisciplinary work and collaboration.*
- *Learning how to value simplicity.* In complex systems, fewer components in the architecture means fewer possible unpredictable interactions.
- *Empirical research in software.* One example of the results of such research was noted—namely, the appearance of the power law distributions for object usage in software. That is, many objects are only used once, some are used multiple times, and very few are used very frequently. Exploring the implications of this and other phenomena may provide insight into development methodologies and how to manage complexity and scale.
- *Testing and integration techniques*. In a complicated deployment environment, finding better ways to get more assurance sooner in the

[13]One example is some research under way at the University of Sheffield. Psychologists watch teams using XP methodologies and then report on the psychological effects of using XP, as opposed to other metrics such as defect rates. Results suggest that people are happier doing things this way. See http://www.shef.ac.uk/dcs/research/groups/vt/research/observatory.html for more information.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

cycle and more frequently should improve software development as a whole.

SESSION 4: QUALITY AND ASSURANCE WITH SCALE AND UNCERTAINTY

Panelists: Joe Jarzombek, Department of Homeland Security; Kris Britton, National Security Agency; Mary Ann Davidson, Oracle Corporation; Gary McGraw, Cigital
Moderator: William Scherlis

Panelist presentations and general discussions in this session were intended to address the following questions, from government and industry perspectives:

- What are the particular challenges for achieving particular assurances for software quality and cybersecurity attributes as scale and interconnection increase?
- What are emerging best practices and technologies?
- What kinds of new technologies and practices could assist? This includes especially interventions that can be made as part of the development process rather than after the fact. Interventions could include practices, processes, tools, and so on.
- How should cost-effectiveness be assessed?
- What are the prospects for certification, both at a comprehensive level and with respect to particular critical quality attributes?

The presentations began by describing the goals and activities of two federal programs in software assurance and went on to explore present and future approaches.

Software Assurance Considerations and the DHS Software Assurance Program

The U.S. Department of Homeland Security (DHS) has a strategic program (discussed in more detail later in this section) to promote integrity, security, and reliability in software.[14] This program for software assur-

[14]The definition of software assurance that DHS uses comes out of the Committee on National Security Systems: namely, it is the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life cycle, and that the software functions in the intended manner. More generally, "assurance" is about confidence—that is, it is a human judgment, not an objective test/verification/analytic result but rather a judgment based on those results.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

ance emphasizes security; the risk exposures associated with reliance on software leave a lot of room for improvements. In industry as well as government there is increased concern about security. Security is difficult to measure. It is difficult to quantify security or assess relative progress in improving it. Participants noted the need for more comprehensive diagnostic capabilities and standards on which to base assurance claims. Two suggestions were made:

- The software assurance tool industry has not been keeping pace with changes in software systems—tools that provide point solutions are available, but much of the software industry cannot apply them. As testing processes become more complex, costly, and time consuming, the testing focus frequently narrows to functional testing.
- Tools are not interoperable. This leads to more standards but, paradoxically, less standardization. Less standardization, in turn, leads to decreased confidence and lower levels of assurance.

One remedy for this situation would have the following elements:

- Government, in collaboration with industry and academia, works to raise expectations on product assurance. This would help to advance more comprehensive diagnostic capabilities, methodologies, and tools to mitigate risks.
- Acquisition managers and users start to factor information about suppliers' software development processes and the risks posed by the supply chain into their decision making and acquisition/delivery processes. Information about evaluated products would become available, and products in use could be securely configured.
- Suppliers begin to deliver quality products with requisite integrity and make assurance claims about their IT and the software's safety, security, and dependability. To do this, they would need to have and use relevant standards, qualified tools, independent third-party certifiers, and a qualified workforce.

It was suggested that software is an industry that demands only minimal levels of responsible practice compared to some other industries and that this is part of the challenge. But raising the level of responsible practice could increase sales to customers that demand high-assurance products.

From the perspective of the DHS, critical infrastructure around the United States is often not owned or operated by U.S. interests. As cyberspace and physical space become increasingly intertwined and software-controlled or -enabled, these interconnections and controls are often

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

implemented using the Internet. This presents a target-rich environment especially given the asymmetries at work: According to one speaker, extrapolating from data on average defect rates, a deployed software package of a million lines of code will have 6,000 defects. Even if only 1 percent of those defects introduce security-related vulnerabilities, then there are 60 different vulnerabilities for an adversary to exploit. In an era riddled with asymmetric cyberattacks, claims about system reliability, integrity, and safety must also address the built-in security of the enabling software. Security is an enabler for reliability, integrity, and safety. Cyber-related disruptions have an economic and business impact because they can lead to the loss of money and time, delayed or cancelled products, and loss of sensitive information, reputation, even life. From a CEO/CIO perspective, disruptions and security flaws can mean having to redeploy staff to deal with problems and increase IT security, reduced end user productivity, delayed products, and unanticipated patch management costs. Results from a survey of CIOs in 2006 by the CIO Executive Council indicate that reliable and vulnerability-free software is a top priority. In that same survey respondents expressed "low to medium confidence" that software is "free from" flaws, vulnerabilities, and malicious code. The majority of these CIOs would like vendors to certify and test software using qualified tools.

Speakers noted that the second national software summit had identified major gaps in requirements for tools and technologies, as well as major shortcomings in the state-of-the-art and the state-of-the practice for developing error-free software. A national software strategy was recommended in order to enhance the nation's capability to routinely develop trustworthy software products and ensure the continued competitiveness of the U.S. software industry. This strategy focused on improving software trustworthiness, educating and fielding the software workforce, re-energizing software R&D, and encouraging innovation in the U.S. industry.[15] In addition to the gaps and shortcomings identified at that software summit, a recent PITAC report on national priorities for cybersecurity listed security software engineering and software assurance as among the top ten goals.[16]

Software assurance contributes to trustworthy software systems. The goals of the DHS Software Assurance (SwA) program promote the security of software across its development, acquisition, and implementation

[15]Center for National Software Studies, 2005, "Software 2015: A national software strategy to ensure U.S. security and competitiveness," available online at www.cnsoftware.org/nss2report, April 29.

[16]President's Information Technology Advisory Committee (PITAC), 2005, *Cybersecurity: A Crisis of Prioritization*, February.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

life cycles.[17] The SwA program is scoped to address trustworthiness, predictable execution, and conformance to requirements, standards, and procedures. It is structured to target people, process, technology, and acquisition.

The SwA program is process-agnostic, providing practical guidance in assurance practices and methodologies for process improvement. A developer's guide and glossary discussed during this session, *Securing the Software Life Cycle*, is not a policy or standard. Instead, it focuses on touch points and artifacts throughout the life cycle that are foundational knowledge, best practices, and tools and resources for building assurance in. Integrating security into the systems engineering life cycle enables the implementation of software assurance.

It was suggested that software assurance would be well-served by standards that assign names to practices or collections of practices. Standards are needed to facilitate communication between buyer and seller, government and industry, insurer and insured. They are needed to improve information exchange and interoperability among practices and among tools. The goal is to close the gap between art and practice and raise the minimum level of responsible practice. Some current standards efforts for software and system life cycle processes include ISO SC7, ISO SC22, ISO SC27, and IEEE S2ESC. A critical aspect, it was suggested, is language: articulating structured assurance claims supported by evidence and reasoning. For example, the Object Management Group (OMG) has been working with industry and federal agencies to help collaboration in a common framework for the analysis and exchange of information related to software trustworthiness. This framework can be used for building and assembling software components, including legacy systems and large systems and networks: Looking only at product evaluation overlooks the places where systems and networks are most vulnerable, because it is the interaction of all the components as installed that becomes the problem.

One of the challenges often noted regarding standardization of practices is the lag between identification of a best practice and its codification into a standard. This is particularly challenging in areas such as software assurance, where there is rapid evolution of technologies, practices, and related standards. In the future, the goal is for customers to have expectations for product assurance—including information about evaluated products, suppliers' process capabilities, and secure configurations of software—and for suppliers to be able to distinguish themselves by delivering quality products with the requisite integrity and to be able to make assurance claims based on relevant standards.

[17]The MITRE Web site, http://www.cwe.mitre.org, can be used to track SwA progress.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

The National Security Agency Center for Assured Software

According to the historical perspective offered by one speaker, the DoD assurance requirements of 30 years ago mostly focused on what became the National Information Assurance Partnership (NIAP) and the trusted product evaluation program.[18] Developers were known and trusted intrinsically. By contrast, in today's environment, the market for software is a global one: Even U.S. companies are international. DoD has become increasingly concerned about malicious intent. Malicious code done very well is going to look like an accident. Unfortunately, it was argued, assurance is gained today the same way as it was 30 years ago. Various mechanisms for gaining confidence in general-purpose software are also being used for DoD software: functional testing, penetration testing, design and implementation analysis, advanced development environments, trusted developers, process, discipline, and so forth are mechanisms to build confidence. The intention is for the Center for Assured Software to contribute to the advancement of measurable confidence in software.

In today's environment, vendors do not have an incentive to be involved early in the design process, so testing typically is done after the fact, with a third-party orientation. The problem with this model is that it is all about penetration analysis, not building security in, and trust is bestowed by a third party. Moreover, this model does not scale very well. In one speaker's view, assurance models for COMSEC devices will not, for example, scale to the DoD's Joint Tactical Radio System (JTRS) program. In addition, composition has always been a problem in the context of assurance: The current state of knowledge about how to compose systems well and to know what we have, with the inadequacy in assurance, is compounded by the problem of malicious intent.

A challenge for NSA's Center for Assured Software is to be able to scale assurance. To do that, the future assurance paradigm needs to acknowledge the role of the development process in the assurance argument. How software is built, what processes are used, and what tools are used all have to be part of the assurance argument. That is a subtle but important shift in the paradigm.

In the speaker's view, the way to achieve scale in the development process and the way to gain assurance in the development process and in third-party analysis is by increasing the extent of automation. The cur-

[18]NIAP is a U.S. government initiative originated to meet the security testing needs of both information technology consumers and producers and is operated by the NSA (see http://www.niap-ccevs.org/). The Trusted Product Evaluation Program (TPEP) was started in 1983 to evaluate COTS products against the Trusted Computer Systems Evaluation Criteria (TCSEC).

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

rent paradigm does not embrace that means of achieving scale very well. Previous measurement techniques mostly entailed humans looking for vulnerabilities. What the Center for Assured Software is trying to do is find correlations between assurance and positive things that can be measured—for instance, the properties that are important—to give confidence that the software is indeed built appropriately. Another area where work is needed, it was suggested, is to create a science of composition that enables making an argument for levels of assurance at scale. In the mid-1980s, there were attempts to do that with the Trusted Database Management System Interpretation of the Trusted Computer System Evaluation Criteria (often referred to as the Orange Book), but the results did not scale very well.

Participants mentioned a variety of ideas being pursued in industry and academia in response to business and government needs in the area of software assurance: anomaly identification, model checking, repeatable methodology for assurance analysis and evaluation, and intermediate representation of executable code.[19] Suggested research areas mentioned during this discussion include these:

- Assurance composition,
- Verifiable compilation,
- Software annotation,
- Model checking,
- Safe languages and automated migration from unsafe languages,
- Software understanding, and
- Measurable attributes that have strong correlation to assurance.

More broadly, participants suggested that it will be important to understand how to build confidence from all of these (and other approaches) and to improve these approaches. In particular, it will be important to understand how they "combine" (that is, what multiple techniques collectively convey regarding confidence) since it is at best highly unlikely that one technique will ever by itself be sufficient.

Software Assurance: Present and Future

This vendor-perspective presentation and discussion focused on COTS (it was suggested that 80 percent of DoD systems have at least some COTS components) and on taking tactical, practical, and economical steps at the component level to improve assurance. As scale and interconnec-

[19]Other approaches include static analysis, extended static checkers, and rule-based automatic code analysis.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

tivity increase, it was argued that the assurance bar for software quality and cybersecurity attributes can be raised by (1) raising the component assurance bar (resources are finite and organizations can spend too much time and too many resources trying to patch their way to security) and (2) getting customers to understand and accept that assurance for custom software can be raised if they are willing to pay more (if customers do not know about costs that are hidden, they cannot accept or budget for them).

One set of best practices and technologies to write secure software was described. It includes

- Secure coding standards,
- Developer training in secure coding,
- Enabled, embedded security points of contact (the "missionary model"),
- Security as part of development including functional, design, test (include threat modeling),
- Regressions (including destructive security tests),
- Automated tools (home grown, commercial of multiple flavors),
- Locked-down configurations (delivering products that are secure on installation), and
- Release criteria for security.

However, these practices are not routinely taught in universities. Neither the software profession not the industry as a whole can simply rely on a few organizations doing these kinds of things. Discussion identified some necessary changes in the long run:

- *University curricula.* It was argued that university programs should do a better job of teaching secure coding practices and training future developers to pay attention to security as part of software development. If the mindset of junior developers does not change, the problem will not be solved. One participant said, "Process won't fix stupidity or arrogance." Incentives to be mindful of security should be integrated throughout the curriculum. When security is embedded throughout the development process, a small core of security experts is not sufficient. One challenge is how to balance the university focus on enduring knowledge and skills against the need for developers to understand particular practices and techniques specific to current technologies.
- *Automation.* Automated tools are promising and will be increasingly important, but they are not a cure-all. Automated tools are not yet ready for universal prime time for a number of reasons, including: Tools need to be trained to understand the code base; programmers have dif-

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

ficulty establishing sound and complete rules; most of today's tools look only for anticipated vulnerabilities (e.g., buffer overruns) and cannot be readily adapted to new classes of vulnerabilities; there are often too many false positives; scalability is an issue; one size does not fit all (it is premature for standards) and therefore multiple tools are needed; and there is not a good system for rating tools.

Conventional wisdom holds that people will not pay more for secure software. However, people already are paying for bad security—a 2002 study by the National Institute of Standards and Technology (NIST) reported that the consequences of bad software cost $59 billion a year in the United States.[20] It was argued that from a development standpoint, security cost-effectiveness should be measured pragmatically. However, a simple return on investment (ROI) is not the right metric. From a developer's perspective, the goal should be the highest net present value (NPV) for cost avoidance of future bugs—not raw cost savings or the ROI from fixing bugs in the current year. Another way of valuing security is opportunity cost savings—what can be done (e.g., building new features) with the resources saved from not producing patches. From the customer's perspective, it is the life-cycle cost of applying a patch weighed against the expected cost of the harm from not applying the patch. Customers want predictable costs, and the perception is that they cannot budget for the cost of applying patches (even though the real cost driver is the consequences of unpatched systems). If customers know what they are getting, they can plan for a certain level of risk at a certain cost. The goal is to find the match between expected risk for the customer and for the vendor—how suitable the product is for its intended use.

Certification is a way of assessing what is "good."[21] But participants were not optimistic when considering prospects for certification of development processes. There is too much disagreement and ideology surrounding development processes. However, there can be some commonality around aspects of good development processes. Certifying developers is also problematic. In engineering, there are accredited degree programs and clear licensing requirements. The awarding of a degree in computer science is not analogous to licensing an engineer because there is not the same common set of requirements, especially robustness and safety requirements. In addition, it can be difficult to replicate the results

[20]See NIST, 2002, "Planning Report 02-3: The economic impacts of inadequate infrastructure for software testing." Available online at http://www.nist.gov/director/prog-ofc/report02-3.pdf.

[21]A recent NRC study examines the issue of certification and dependability of software systems. See information on the report *Software for Dependable Systems: Sufficient Evidence*? at http://cstb.org/project_dependable.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

of software engineering processes, making it hard to achieve confidence such that developers are willing to sign off on their work. Moreover, it was argued that with current curricula, developers generally do not even learn the basics of secure coding practice. There is little to no focus on security, safety, or the possibility that the code is going to be attacked in most educational programs. It was argued that curricula need to change and that computer science graduates should be taught to "assume an enemy."

Automated tools can give better assurance to the extent that vendors use them in development and fix what they find. Running evaluation tools after the fact on something already in production is not optimal.[22] It was suggested that there is potential for some kind of "goodness meter" (a complement to the "badness meter" described in the next section) for tool use and effectiveness—what tool was used, what the tool can and cannot find, what the tool did and did not find, the amount of code covered, and that tool use was verified by a third party.

Software Security: Building Security In

Discussions in this session focused on software security as a systems problem as opposed to an application problem. In the current state of the practice, certain attributes of software make software security a challenge: (1) connectivity—the Internet is everywhere and most software is on it or connected to it; (2) complexity—networked, distributed, mobile code is hard to develop; and (3) extensibility—systems evolve in unexpected ways and are changed on the fly. This combination of attributes also contributes to the rise of malicious code.

Massively multiplayer online games (MMOGs) are bellwethers of things to come in terms of sophisticated attacks and exploitation of vulnerabilities. These games experience the cutting edge of what is going on in software hacking and attacks today.[23] Attacks against such games are

[22]It was suggested that vendors should not be required to vet products against numerous tools. It was also suggested that there is a need for some sort of Common Criteria reform with mutual recognition in multiple venues, eliminating the need to meet both Common Criteria and testing requirements. Vendors, for example, want to avoid having to give governments the source code for testing, which could compromise intellectual property, and want to avoid revealing specifics on vulnerabilities (which may raise security issues and also put users of older versions of the code more at risk). Common Criteria is an international standard for computer security. Documentation for it can be found at http://www.niapccevs.org/cc-scheme/cc_docs/.

[23]World of Warcraft, for example, was described as essentially a global information grid with approximately 6 million subscribers and half a million people playing in real time at any given time. It has its own internal market economy, as well as a significant black market economy.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

also at the forefront of so-called rootkit[24] technology. Examining attacks on large-scale games may be a guide to what is likely to happen in the non-game world. It was suggested that in 2006, security started to become a differentiator among commercial products. Around that time, companies began televising ads about security and explicitly offering security features in new products. Customers were more open to the idea of using multiple vendors to take advantage of diversity in features and suppliers.

Security problems are complicated. There is a difference between implementation bugs such as buffer overflows or unsafe systems calls, and architectural flaws such as compartmentalization problems in design or insecure auditing. As much attention needs to be paid to looking for architectural or requirements flaws as is paid to looking for coding bugs. Although progress is being made in automation, both processes still need people in the loop. When a tool turns up bugs or flaws, it gives some indication of the "badness" of the code—a "badness-o-meter" of sorts. But when use of a tool does not turn up any problems, this is not an indication of the "goodness" of the code. Instead, one is left without much new knowledge at all.

Participants emphasized that software security is not application security. Software is everywhere—not just in the applications. Software is in the operating system, the firewall, the intrusion detection system, the public key infrastructure, and so on. These are not "applications." Application security methods work from the outside in. They work for COTS software, require relatively little expertise, and are aimed at protecting installed software from harm and malicious code. System software security works from the inside out, with input into and analysis of design and implementation, and requires a lot of expertise.

In one participant's view, security should also be thought of as an emergent property of software, just like quality. It cannot be added on. It has to be designed in. Vendors are placing increased emphasis on security, and most customers have a group devoted to software security. It was suggested that the tools market is growing, for both application security (a market of between $60 million and $80 million) and software security (a market of about $20 million, mostly for static analysis tools). Consulting services, however, dwarf the tools market. One speaker described the "three pillars" of software security:

[24]A rootkit is a set of software tools that can allow hackers to continue to gain undetected, unauthorized access to a system following an initial, successful attack by concealing processes, files, or data from the operating system.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

- *Risk management*, tied to the mission or line of business. Financial institutions such as banks and credit card consortiums are in the lead here, in part because Sarbanes-Oxley made banks recognize their software risk.
- *Touchpoints*, or best practices. The top two are code review with a tool and architectural risk analysis.
- *Knowledge*, including enterprise knowledge bases about security principles, guidelines, and rules; attach patterns; vulnerabilities; and historical risks.

SESSION 5: ENTERPRISE SCALE AND BEYOND

Panelists: Werner Vogels, Amazon.com, and Alfred Spector, AZS-Services
Moderator: Jim Larus

The speakers at this session focused on the following topics, from the perspective of industry:

- What are the characteristics of successful approaches to addressing scale and uncertainty in the commercial sector, and what can the defense community learn from this experience?
- What are the emerging software challenges for large-scale enterprises and interconnected enterprises?
- What do you see as emerging technology developments that relate to this?

Life Is Not a State-Machine: The Long Road from Research to Production

Discussions during this session centered on large-scale Web operations, such as that of Amazon.com, and what lessons about scale and uncertainty can be drawn from them. It was argued that in some ways, software systems are similar to biological systems. Characteristics and activities such as redundancy, feedback, modularity, loose coupling, purging, apoptosis (programmed cell death), spatial compartmentalization, and distributed processing are all familiar to software-intensive systems developers, and yet these terms can all be found in discussions of robustness in biological systems. It was suggested that there may be useful lessons to be drawn from that analogy.

Amazon.com is very large in scale and scope of operations: It has seven Web sites; more than 61 million active customer accounts and over 1.1 million active seller accounts, plus hundreds of thousands of

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

registered associates; over 200,000 registered Web services developers; over 12,500 employees worldwide; and more than 20 fulfillment centers worldwide. About 30 percent of Amazon's sales are made by third-party sellers; almost half of its sales are to buyers outside the United States. On a peak shipping day in 2006, Amazon made 3.4 million shipments. Amazon.com's technical challenges include how to manage millions of commodity systems, how to manage many very large, geographically dispersed facilities in concert, how to manage thousands of services running on these systems, how to ensure that the aggregate of these services produces the desired functionality, and how to develop services that can exploit commodity computing power. It, like other companies providing similar kinds of services, faces challenges of scale and uncertainty on an hourly basis.

Over the years, Amazon has undergone numerous transformations—from retailer to technology provider, from single application to platform, from Web site and database to a massively distributed parallel system, from Web site to Web service, from enterprise scale to Web scale. Amazon's approach to managing massive scale can be thought of as "controlled chaos." It continuously uses probabilistic and chaotic techniques to monitor business patterns and how its systems are performing. As its lines of business have expanded these techniques have had to evolve—for example, focusing on tracking customer returns as a negative metric does not work once product lines expand into clothing (people are happy to order multiple sizes, keep the one that fits, and return the rest).

Amazon builds almost all of its own software because the commercial and open source infrastructure available now does not suit Amazon.com's needs. The old technology adoption life cycle from product development to useful acceptance was between 5 and 25 years. Amazon and similar companies are trying to accelerate this cycle. However, it was suggested that for an Amazon developer to select and use a research technology is almost impossible. In research, there are too many possibilities to choose from, experiments are unrealistic compared to real life, and underlying assumptions are frequently too constrained. In real life, systems are unstable, parameters change and things fail continuously, perturbations and disruptions are frequent, there are always malicious actors, and failures are highly correlated. In the real world, when the system fails, the mission of the organization cannot stop—it must continue.[25]

Often, complexity is introduced to manage uncertainty. However, there may well exist what one speaker called "conservation laws of complexity." That is, in a complex interconnected system, complexity cannot

[25]Examples of systems where assumptions did not match real life include the *Titanic*, the Tacoma Narrows bridge, and the Estonian ferry disaster.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

be reduced absolutely, it can only be moved around. If uncertainty is not taken into account in large scale system design, it makes adoption of the chosen technology fairly difficult. Engineers in real life are used to dealing with uncertainty. Assumptions are often added to make uncertainty manageable. At Amazon, the approach is to apply Occam's razor: If there are competing systems to choose from, pick the system that has the fewest assumptions. In general, assumptions are the things that are really limiting and could limit the system's applicability to real life.

Two different engineering approaches were contrasted, one with the goal of building the best possible system (the "right" system) whatever the cost, and the other with the more modest goal of building a smaller, less-ambitious system that works well and can evolve. The speaker characterized the former as being incredibly difficult, taking a long time and requiring the most sophisticated hardware. By contrast, the latter approach can be faster, it conditions users to expect less, and it can, over time, be improved to a point where performance almost matches that of the best possible system.

It was also argued that traditional management does not work for complex software development, given the lack of inspection and control. Control requires determinism, which is ultimately an illusion. Amazon's approach is to optimize team communication by reducing team size to maximum of 8-10 people (a "two-pizza team"). For larger problems, decompose the problem and reduce the size of the team needed to tackle the subproblems to a two-pizza group. If this cannot be done, it was suggested, than do not try to solve that problem—it's too complicated.

A general lesson that was promoted during this session was to let go of control and the notion that these systems can be controlled. Large systems cannot be controlled—they are not deterministic. For various reasons, it is not possible to consider all the inputs. Some may not have been included in the original design; requirements may have changed; the environment may have changed. There may be new networks and/or new controllers. The problem is not one of control; it is dealing with all the interactions among all the different pieces of the system that cannot be controlled. Amazon.com's approach is to let these systems mature incrementally, with iterative improvement to yield the desired outcome during a given time period.

The Old, the Old, and the New

In this session's discussions, the first "old" was the principle of abstraction-encapsulation-reuse. Reuse is of increasing importance everywhere as the sheer quantity of valuable software components continues to grow. The second "old" was the repeated quest (now using Web services

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

and increasingly sophisticated software tools) to make component reuse and integration the state of practice. Progress is being made in both of these areas, as evidenced by investment and anecdotes. The "new" discussed was the view that highly structured, highly engineered systems may have significant limitations. Accordingly, it was argued, "semantic integration," more akin to Internet search, will play a more important role.

There are several global integration agendas. Some involve broad societal goals such as trade, education, health care, and security. At the firm or organization level, there is supply chain integration and *N* to 1 integration of many stores focusing on one consumer, as in the case Amazon and its many partners and vendors. In addition, there is collaborative engineering, multidisciplinary R&D, and much more.

Why is global integration happening? For one thing, it is now technically possible, given ubiquitous networking, faster computers, new software methodologies. People, organizations, computation, and development are distributed, and networked systems are now accepted as part of life and business, along with the concomitant benefits and risks (including security risks). An emerging trend is the drive to integrate these distributed people and processes to get efficiency and cost-effective development derived from reuse.

Another factor is that there are more software components to integrate and assemble. Pooling of the world's software capital stock is creating heretofore unimaginably creative applications. Software is a major element of the economy. It was noted that by 2004, the amount of U.S. commercial capital stock relating to software, computer hardware, and telecommunications accounted for almost one-quarter of the total capital stock of business; about 40 percent of this is software. Software's real value in the economy could even be understated because of accounting rules (depreciation), price decreases, and improvements in infrastructure and computing power. The IT agenda and societal integration reinforce each other.

Core elements of computer science, such as algorithms and data structures, are building blocks in the integration agenda. The field has been focusing more and more on the creation and assembly of larger, more flexible abstractions. It was suggested that if one accepts that the notion of abstraction-encapsulation-reuse is central, then it might seem that service-oriented computing is a done deal. However, the challenge is in the details: How can the benefits of the integration agenda be achieved throughout society? How are technologists and developers going to create these large abstractions and use them?

When the Internet was developed, some details—such as quality of service and security—were left undone. Similarly, there are open chal-

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

lenges with regard to integration and service-oriented approaches. What are the complete semantics of services? What security inheres in the service being used? What are the failure modes and dependencies? What is the architectural structure of the world's core IT and application services? How does it all play out over time? What is this hierarchy that occurs globally or, for the purposes of this workshop, perhaps even within DoD or within one of the branches of the military?

Service-oriented computing is computing whereby one can create, flexibly deploy, manage, meter and charge for (as appropriate), secure, locate, use, and modify computer programs that define and implement well-specified functions, having some general utility (services), often recursively using other services developed and deployed *across time and space*, and where computing solutions can be built with a heavy reliance on these services. Progress in service-oriented computing brings together information sharing, programming methodologies, transaction processing, open systems approaches, distributed computing technologies, and Web technologies.

There is now is a huge effort on the part of industry to develop application-level standards. In this approach, companies are presented with the definition of some structure that they need to use to interoperate with other businesses, rather than, for example, having multiple individual fiefdoms within each company develops unique customer objects.

The Web services approach generally implies a set of services that can be invoked across a network. For many, Web services comprise things such as Extensible Markup Language (XML) and SOAP (a protocol for exchanging XML-based messages over computer networks) along with a variety of Web service protocols that have now been defined and are heavily used, developed, produced, and standardized (many in a partnership between IBM and Microsoft). Web services are on the path to full-scale, service-oriented computing; it was argued that this path can be traced back to the 1960s and the airlines' Sabre system, continuing through Arpanet, the Internet, and the modern World Wide Web.

Web services based on abstraction-encapsulation-reuse are a new approach to applying structure-oriented engineering tradition to information technology (IT). For example, integration steps include the precise definition of function (analogous to the engineering specifications and standards for transportation system construction), architecture (analogous to bridge design, for example), decomposition, rigorous component production (steel beams, for example), careful assembly, and managed change control. The problem is, there may be limits to this at scale. In software, each of these integration steps is difficult in itself. Many projects are inherently multiorganizational, and rapid changes have dire consequences for traditional waterfall methodologies.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

It was argued that "semantic integration," a dynamic, fuzzier integration more akin to Internet search, will play a larger role in integration than more highly structured engineering of systems. Ad hoc integration is a more humble approach to service-based integration, but it is also more dynamic and interpretive. Components that are integrated may be of lower generality (not a universal object) and quality (not so well specified). Because they will be of lower generality, perhaps with different coordinate systems, there will have to be automated impendence matching between them. Integration may take place on an intermediate service, perhaps in a browser. Businesses are increasingly focusing on this approach for the same reasons that simple approaches have always been favored. This is a core motivational component of the Web 2.0 mash-up focus. Another approach to ad hoc integration uses access to massive amounts of information—with no reasonable set of predefined, parameterized interfaces, annotation and search will be used as the integration paradigm.

It is likely that there will be tremendous growth in the standards needed to capitalize on the large and growing IT capital plant. There will be great variability from industry to industry and from place to place around the world, depending on the roles of the industry groups involved, differential regulations, applicable types of open source, and national interests. Partnerships between the IT industry and other industries will be needed to share expertise and methodologies for creating usable standards, working with competitors, and managing intellectual property.

A number of topics for service-oriented systems and semantic integration research were identified, some of which overlap with traditional software system challenges. The service-oriented systems research areas and semantic integration research areas spotlighted included these:

- *Basics*. Is there a, practical, normative general theory of consistency models? Are services just a remote procedure call invocation or a complex split between client and server? How are security and privacy to be provided for the real world, particularly if one does not know what services are being called? How does one utilize parallelism? This is an increasingly important question in an era of lessening geometric clock-speed growth.
- *Management*. With so many components and so much information hiding, how does one manage systems? How does one manage intellectual property?
- *Global properties*. Can one provide scalability generally? How does one converge on universality and standards without bloat? What systems can one deploy as really universal service repositories?

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

- *Economics.* What are realistic costing/charging models and implementations?
- *Social networking*. How does one apply social networking technology to help?
- *Ontologies of vast breadth and scale.*
- *Automated discovery and transformation.*
- *Reasoning in the control flow*.
- *Use of heuristics and redundancy.*
- *Search as a new paradigm.*

Complexity grows despite all that has been done in computer science. There is valuable, rewarding, and concrete work for the field of computer science in combating complexity. This area of work requires focus. It could prove as valuable as direct functional innovation. Participants identified several research areas to address complexity relevant to service-oriented systems and beyond, including: meaning, measuring, methodology, system architecture, science and technology, evolutionary systems design, and legal and cultural change.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

3

Wrap-up Discussion and Emergent Themes

Several major themes emerged from the day's discussions on the challenges of developing future-oriented, large-scale systems that can cope with uncertainty at scale.[1] These themes are not findings or recommendations of the study committee—those will be presented in the committee's final report later in the study. Indeed, observations offered in some sessions contradicted those from other sessions—which perhaps reflects the notion that there are different kinds of very large-scale systems and that different kinds of systems will likely warrant different sorts of approaches. Instead, this section presents a brief overview of the interrelated themes that arose over the course of the workshop's discussions:

- Architectural challenges in large-scale systems,
- The need for software engineering capability, and
- Open questions and research opportunities.

ARCHITECTURAL CHALLENGES IN LARGE-SCALE SYSTEMS

The issue of architecture and frameworks for large-scale software-intensive systems coping with uncertainty at scale was raised repeatedly. One approach to this problem that was put forward in discussions would

[1]Because neither the workshop nor this summary was intended to be (or constituted as) a stand-alone product, contradictory views also emerged from different presenters during the day—for example, the desirability of not producing software in-house versus the desirability of producing all software in-house.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

be to develop executable models and architectures that can evolve over the system's life cycle. As the system develops and evolves, pieces that were originally mock-ups could be replaced with actual subsystems—this would be a way to provide ongoing feedback on the architecture, including system performance, functionality, and so on. A broader question asks what kind of structure and constraints can be imposed at a high level such that the overall system can be decomposed (as needed) into autonomous pieces. What specific architectural rules and approaches will get one there? It was suggested that a framework would be needed, along with clear specifications at the interface between the framework and components.

Closely tied to the question of architecture is the significant challenge of how to develop systems architectures and definitions for highly decentralized organizations. For developing the sorts of systems under consideration, some believed a loosely coupled organizational style would be needed as systems scale up and more players enter the picture. Although such a shift might place intense pressure on organizational culture and management, "controlled chaos" rather than a very top-down, structured, and controlled approach might need to become the order of the day. This shift, however, might reflect a change of perspective on the essential commonalities that hold an overall system together—there could, for example, be a shift from an overall structural model as the unifying factor to particular agreements on how components and subsystems are to interact with each other through protocols, APIs, metadata standards, and the like. It may also be the case that there is an underlying issue driving this tension that is not about coupling or control, but rather about the particular nature of the architectural commonalities that hold a system together. For example, the trend towards dynamic architecture demonstrates that purely structural commonalities are giving way to semantic and other less apparent—but perhaps more essential—commonalities in large systems.

This has implications for software engineering capability (see below), in part because the frameworks and architecture for these systems will not go away—they and the people involved with them will need to persist for the lifetime of the system. And, of course, the architecture for these systems will transcend particular individual suppliers.

THE NEED FOR SOFTWARE ENGINEERING CAPABILITY

Writing large, safety-critical, real-time systems requires a commitment to genuine engineering discipline, even if it means constraining the design space—limiting flexibility—in order to conform to precedented practices that enable application of this discipline. In addition, management becomes a significant challenge when it comes to very large-scale systems

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

that will need to depend on very large networks and supply chains,. It was suggested that this may lead to a focus on community-style collaboration and integration over the long term. In any event, best practices will differ depending on context, including the type of organization as well as the type of application or system under development. Participants noted that process practices will merit attention as well as technological practices. Process will not solve everything, of course, but process is vital to assist with, among other things, the decomposition of large systems into incremental subsets for better visibility and reduced risk. It was noted that the focus on reuse may shift to earlier stages of the process and particularly to requirements. In general, supporting technology will be needed to enable the types of architectures and collaborations that large-scale systems merit. In particular, finding ways to increase the extent of abstraction and automation in all aspects of software design, implementation, testing, maintenance, and so on, may be a productive avenue, especially with regard to scale and geographic distribution. Tools and strategies for coping with design and architectures for very large, physically distributed teams of people and organizations will be increasingly important along with techniques and approaches that support high levels of trust. In addition, participants noted that the ability to use analytic techniques (such as model checking, static analysis, dynamic analysis, and so on) will be important. These types of tools are being used much more now than 5 or 10 years ago, but they are typically still early-generation tools. Continued research and investment (see below) will help improve and extend them. Even so, tools will not solve the problems of assurance or of large-scale system development. Process, expertise, and skills matter a great deal.

OPEN QUESTIONS AND RESEARCH OPPORTUNITIES

Ultimately, many of the challenges related to architecture and capability reflect problems that the community does not yet know how to solve. Over the course of the workshop, participants articulated several open research questions that should be addressed to make progress in addressing uncertainty at scale for software-intensive systems. At a high level, the architectural and organizational challenges presented by large-scale systems merit investigation: What are the key kinds of commonalities that manifest architectural commitment, beyond a top-down structural design? Industry issues were among the other topics that were spotlighted in the wrap-up discussion. These included (1) the extent to which service-oriented architecture (SOA) and SOA vendors will or will not make progress over the next 18 to 24 months in addressing DoD's need for producibility at scale—for example, in contrast to the well-established enterprise resource planning framework and application servers offered

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

by vendors—and (2) the extent to which defense contractors can allocate resources to address software challenges that fall outside current contract parameters. Contracting issues that were noted in the discussion included (1) how to establish collaborative mechanisms for contractors and the DoD to work together, particularly in iterative fashion, on software assurance and risk-reduction problems, as well as (2) contracting complexities related to the integration of large software systems that include COTS subsystems for the DoD. Another industry issue noted in the discussion was industry's misgivings about the availability of computer science and computer engineering new hires at the bachelor's and master's degree level. Related issues were software curricula development and the appropriateness of accreditation for software engineering and computer engineering and the need for computer science and computer engineering faculty and students to have hands-on industry experience in building systems.

* * *

Discussions at the workshop suggested that many of the key ideas needed to make progress in developing large-scale systems and coping with uncertainty at scale will not be found through the traditional incremental advances made in various segments of the industry. While there are lessons to be learned and gleaned from the varieties of experience and perspective presented, the types of systems that are envisioned and that serve as a driver for DoD's interests in software pose significant management, intellectual, and research challenges.

Copyright © National Academy of Sciences. All rights reserved.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Appendixes

Copyright © National Academy of Sciences. All rights reserved.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

A

Workshop Agenda

JANUARY 17, 2007
WASHINGTON, D.C.

9:00-9:15 a.m.	**Opening Remarks** *Committee on Advancing Software-Intensive Systems Producibility, William Scherlis, Chair*
9:15-10:30	**Session 1: Process, Architecture, and the Grand Scale** **Moderator: Michael Cusumano** *John Vu, Boeing* *Rick Selby, Northrop Grumman Corporation* What are characteristics of successful approaches to architecture and design for large-scale systems and families of systems? What are architecture ideas that can apply when systems must evolve rapidly? What kinds of management and measurement approaches could assist in guiding program managers and developers?
10:30-10:45	Break
10:45-Noon	**Session 2: DoD Software Challenges for Future Systems** **Moderator: Douglas Schmidt** *Kristen Baldwin, Office of the Undersecretary of Defense for Acquisitions, Technology and Logistics* *Patrick Lardieri, Lockheed Martin*

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

How are challenges for software in DoD systems, particularly cyber-physical systems, being met in the current environment? What advances in R&D, technology, and practice do we need to achieve success as demands on this capability increase, particularly with respect to scale, complexity, and the increasingly rapid rate of evolution in requirements (and threat)?

Noon-12:45 p.m. Lunch

12:45-1:20 **Session 3: Agility at Scale**
Moderator: Douglas Schmidt
Kent Beck, Three Rivers Institute
Cynthia Andres, Three Rivers Institute

How can the engineering and management values that the "agile community" has identified be achieved for larger-scale systems and for systems that must evolve in response to rapidly changing requirements?

1:20-2:30 **Session 4: Quality and Assurance with Scale and Uncertainty**
Moderator: William Scherlis
Joe Jarzombek, Department of Homeland Security
Kris Britton, National Security Agency

What are the particular challenges for achieving particular assurances for software quality and cybersecurity attributes as scale and interconnection increase? What are emerging best practices and technologies? What kinds of new technologies and practices could assist? This includes, particularly, interventions that can be made as part of the development process, rather than after the fact. Interventions could include practices, processes, tools, and so on. How should cost-effectiveness be assessed? What are the prospects for certification, both at a comprehensive level and with respect to particular critical quality attributes?

2:30-2:45 Break

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

2:45-4:00	**Session 4 (continued)** *Mary Ann Davidson, Oracle Corporation* *Gary McGraw, Cigital*
4:00-5:15	**Session 5: Enterprise Scale and Beyond** **Moderator: Jim Larus** *Werner Vogels, Amazon.com* *Alfred Spector, AZS-Services* What are the characteristics of successful approaches to addressing scale and uncertainty in the commercial sector, and what can the defense community learn from this experience? What are the emerging software challenges for large-scale enterprises and interconnected enterprises? What do you see as emerging technology developments that relate to this?
5:15-6:00	**Closing Discussion** **Moderator: William Scherlis** *All workshop participants and attendees*
6:00	Adjourn

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

B

Biosketches of Committee Members and Staff

COMMITTEE MEMBERS

William L. Scherlis, *Chair*, is a full professor in the School of Computer Science at Carnegie Mellon. He is the founding director of CMU's doctoral program in software engineering and director of its International Software Research Institute. His research relates to software assurance, software evolution, and technology to support software teams. Dr. Scherlis joined the CMU faculty after completing a Ph.D. in computer science at Stanford University, a year at the University of Edinburgh (Scotland) as a John Knox Fellow, and an A.B. at Harvard University. He was the lead principal investigator of the 4-year High Dependability Computing Project, in which CMU leads a collaboration with five universities to help NASA address long-term software dependability challenges. He is also co-PI (with two colleagues) of a new project with NASA and diverse industry and laboratory subcontractors focused on dependable real-time and embedded software systems. Dr. Scherlis is involved in a number of activities related to technology and policy, recently testifying before Congress on innovation and information technology and, previously, on roles for a federal chief information officer. He interrupted his career at CMU to serve at DARPA for 6 years, departing in 1993 as senior executive responsible for coordination of software research. While at DARPA he had responsibility for research and strategy in computer security, aspects of high-performance computing, information infrastructure, and other topics. Dr. Scherlis is a member of the NRC's Committee on Improving Cybersecurity Research in the United States and the DARPA Information

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Science and Technology Study Group. He recently finished chairing a NRC study on information technology, innovation, and e-government. He has led or participated in national studies related to cybersecurity, crisis response, analyst information management, DoD software management, and health care informatics infrastructure. He has been an advisor to major IT companies. He served as program chair for a number of technical conferences, including the ACM Foundations of Software Engineering Symposium. He has more than 70 scientific publications.

Robert F. Behler is a senior vice president in the MITRE Corporation Command and Control Center for programs and advanced command and control. The center serves MITRE's DoD sponsors and focuses on creating a joint command, control, and communications system. Mr. Behler leads the center's work for DoD sponsors. Before joining MITRE in April 2006, Mr. Behler was general manager of Precision Engagement at Johns Hopkins University's Applied Physics Laboratory. In this position he supervised over 250 scientists and engineers working on advanced command, control, intelligence, surveillance, and reconnaissance (C2ISR) programs for the DoD. Under Mr. Behler's leadership, the Precision Engagement organization turned new and emerging technologies into transformational operational capabilities. Mr. Behler retired from the Air Force as a major general in 2003. During his distinguished 31-year career, he accumulated extensive experience managing and developing advanced command, control, communications, computers, intelligence, surveillance, and reconnaissance (C4ISR) technologies at all levels. Before retiring, Mr. Behler was commander of the Air Force C2ISR Center at Langley Air Force Base, where he was principal C2ISR advisor to the secretary and chief of staff of the Air Force. Prior to that, he served as deputy commander of NATO Joint Headquarters North in Stavanger, Norway, and was the senior U.S. military officer in Scandinavia. He has also served as director of command, control, communication, computers, and intelligence at the U.S. Strategic Command at Offutt Air Force Base and as chief of the U.S. Air Force-Senate Liaison Office. Mr. Behler entered the Air Force in 1972 as a distinguished graduate of the Air Force Reserve Officer Training Corps program at the University of Oklahoma. He received his bachelor's and master's degrees in aerospace engineering from the University of Oklahoma in 1970 and 1972, respectively. He is a graduate of the U.S. Air Force Test Pilot School at Edwards Air Force Base and has accumulated over 5,000 flying hours in more than 65 aircraft types, including the SR-71 and U-2. He was a National Security Fellow at Harvard University's John F. Kennedy School of Government in 1990 and received a master's degree in business administration from Marymount University in 1991. He is an

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

associate fellow of the Society of Experimental Test Pilots and a member of the Armed Forces Communications and Electronics Association.

Barry W. Boehm, NAE, is TRW Professor of Software Engineering and Director, Center for Software Engineering, at the University of Southern California. Dr. Boehm received his B.A. degree from Harvard University in 1957, and his M.S. and Ph.D. degrees from University of California, Los Angeles, in 1961 and 1964, all in mathematics. He also received an honorary Sc.D. in computer science from the University of Massachusetts in 2000. Between 1989 and 1992, he served at the DoD as director of the DARPA Information Science and Technology Office, and as director of the DDR&E Software and Computer Technology Office. He worked at TRW from 1973 to 1989, culminating as chief scientist of the Defense Systems Group, and at the Rand Corporation from 1959 to 1973, culminating as head of the Information Sciences Department. He was a programmer-analyst at General Dynamics between 1955 and 1959. His current research interest focus on value-based software engineering, including a method for integrating a software system's process models, product models, property models, and success models called Model-Based (System) Architecting and Software Engineering (MBASE). His contributions to the field include the Constructive Cost Model (COCOMO), the Spiral Model of the software process, the Theory W (win-win) approach to software management and requirements determination, the foundations for the areas of software risk management and software quality factor analysis, and two advanced software engineering environments: the TRW Software Productivity System and Quantum Leap Environment. He has served on the boards of several scientific journals, including the *IEEE Transactions on Software Engineering*, *IEEE Computer*, *IEEE Software*, *ACM Computing Reviews*, *Automated Software Engineering*, *Software Process*, and *Information and Software Technology*. He has served as chair of the AIAA Technical Committee on Computer Systems, chair of IEEE Technical Committee on Software Engineering, and as a member of the Governing Board of the IEEE Computer Society. He has served as chair of the Air Force Scientific Advisory Board's Information Technology Panel, chair of the NASA Research and Technology Advisory Committee for Guidance, Control, and Information Processing, and chair of the board of visitors for the CMU Software Engineering Institute.

Lori A. Clarke is a professor of computer science at the University of Massachusetts, Amherst. She is an ACM Fellow, vice chair of the Computing Research Association's board of directors, and a member of the CRA-W. She is a former IEEE distinguished visitor, ACM national lecturer, IEEE publication board member, associate editor of *ACM TOPLAS* and *IEEE*

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

TSE, member of the CCR NSF advisory board, ACM SIGSOFT secretary/treasurer, vice-chair and chair, as well as a 1990 recipient of the University of Massachusetts' Chancellor's Medal, and a 1993 recipient of a university faculty fellowship. Dr. Clarke has worked in the area of software engineering, particularly on software analysis and testing, for many years. She was one of the primary developers of symbolic execution, a technique used to reason about the behavior of software systems and for selecting test data, and made contributions in the areas of software architecture and object management. Recently her work has focused on analysis of concurrent systems. With colleagues, she has developed FLAVERS, a static analysis tool that uses data-flow analysis techniques to verify user-specified properties. FLAVERS automatically creates a concise, but perhaps imprecise, model of the software system and then allows users to selectively improve the accuracy of the program model as needed to improve the accuracy of the results. The PROPEL system complements FLAVERS and other event-based, finite-state verification systems by helping users elucidate the details of the properties to be proven. FLAVERS allows users to simultaneously view and construct properties from templates of English-language phrases or finite-state automata. The long-term goal is to develop techniques that well-trained software engineers can use to improve the quality of software systems. She received her B.A. in mathematics (1969) from the University of Rochester and her Ph.D. in computer science (1976) from the University of Colorado.

Michael A. Cusumano is the *Sloan Management Review*'s distinguished professor at the Massachusetts Institute of Technology's Sloan School of Management. He specializes in strategy, product development, and entrepreneurship in the computer software industry, as well as automobiles and consumer electronics. He teaches courses on strategic management, innovation and entrepreneurship, and the software business. He has consulted for some 50 major companies around the world. He has been a director of NuMega Technologies (sold to Compuware in 1998 for $150 million) and Infinium Software (sold to SSA Global Technologies in 2002 for $105 million), as well as other private and public software companies. He is currently a director of Patni Computer Systems (software outsourcing based in India) and Entigo (warrantee management software) and an advisor to NetNumina Solutions (Internet architecture and custom solutions), firstRain (wireless and Web services software), H-5 Technologies (digital search technology), and Sigma Technology Group PLC (early-stage ventures). He has also served as editor-in-chief and chairman of the *MIT Sloan Management Review* and writes periodically for *Communications of the ACM*, *The Wall Street Journal*, *Computerworld*, *The Washington Post*, and other publications. Dr. Cusumano has published eight books. His

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

latest book, *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad*, was published in March 2004. Dr. Cusumano received a B.A. degree from Princeton in 1976 and a Ph.D. from Harvard in 1984. He completed a postdoctoral fellowship in production and operations management at the Harvard Business School from 1984 to 1986. He is fluent in Japanese and lived and worked in Japan for 7 years. He received two Fulbright fellowships and a Japan Foundation Fellowship for studying at Tokyo University.

Mary Ann Davidson is the chief security officer at Oracle Corporation, responsible for security evaluations, assessments, and incident handling. As a senior executive in the IT industry she brings both a military and a business background and in-depth experience with and perspective on industrial capacity to respond to Defense needs. She represents Oracle on the board of directors of the Information Technology Information Security Analysis Center (IT-ISAC) and is on the editorial review board of the *Secure Business Quarterly*. Ms. Davidson has a B.S.M.E. from the University of Virginia and an M.B.A. from the Wharton School of the University of Pennsylvania. She has also served as a commissioned officer in the U.S. Navy Civil Engineer Corps, where she was awarded the Navy Achievement Medal.

Larry Druffel recently retired as president and CEO of SCRA, a public, nonprofit research and development corporation engaged in the application of advanced technology. He is a member of the board of directors of Teknowledge Corporation and a member of the advisory board of Amaix Corporation. He was the director of the Software Engineering Institute (SEI) at Carnegie Mellon from 1986 to 1996, where he initiated the Computer Emergency Response Team (CERT) in 1987. Before joining SEI, he was vice president for business development at Rational Software. He served on the board of directors of Rational from 1986 to 1995. Dr. Druffel was on the faculty at the U.S. Air Force Academy. He later managed research programs in advanced software technology at DARPA. He was founding director of the Ada Joint Program Office and then served as director of Computer Systems and Software (research and advanced technology) in the Office of the Secretary of Defense. He is the coauthor of a computer science textbook and over 35 professional papers, including the chapter "Information Warfare" for the ACM fiftieth anniversary book *Beyond Computing*. He has a B.S. in electrical engineering from the University of Illinois, an M.Sc. in computer science from the University of London, and a Ph.D. in computer science from Vanderbilt University. Dr. Druffel is a fellow of the IEEE and a fellow of the ACM. He has served on

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

engineering advisory boards of the University of South Carolina, Clemson, and Embry Riddle University. He was chairman of the board of directors of the Advanced Technology Institute, a nonprofit R&D corporation. Dr. Druffel chaired the Air Force Science Advisory Board (AFSAB) study on information architecture and co-chaired the Defense Science Board's study on acquiring defense software commercially. He led the Defensive Information Warfare Panel for the AFSAB's *New World Vistas*. He has served on numerous AFSAB, DSB, and NRC committees dealing with the use of information technology, including the NRC study on engineering challenges to the long term operation of the International Space Station.

Russell Frew is the vice president, programs and technology, for Lockheed Martin's Electronic Systems Business Area (ESBA). In this capacity he is responsible for overseeing both technology development and program performance in the business sector. He is frequently called upon to lead engineering assistance teams that engage major programs across the corporation struggling with significant technical and programmatic issues. In his capacity as the ESBA chief technical officer, he is also responsible for the technology strategy and the investment plan. Additionally, Mr. Frew has executive responsibility for the Advanced Technology Laboratories in Cherry Hill, New Jersey. From 1999 to late 2003, Mr. Frew was on special assignment from the MS2 staff to the executive vice president, ESBA. In this capacity he has led major program tiger teams working on F/A-22 avionics stability, the F-35 Joint Strike Fighter's Mission System redesign, and the F-16 Block 60 Advanced Mission Computer. As part of the COTS revolution, Mr. Frew authored and leads the Lockheed Martin Proven Path electronics program. Prior to his appointment as vice president advanced technology for MS2 in 1999, he spent 18 months as vice president, technology, for Government Electronics Systems (GES) in Moorestown, New Jersey. While with GES he managed leap-ahead technology programs such as COMBATS and InfoScene. From June 1996 to March 1997, Mr. Frew was executive director of the Lockheed Martin's Advanced Technology Laboratories (ATL). During his tenure, Mr. Frew conceived and led a 9-month study for DARPA on collaborative intelligent software agents. Before that, Mr. Frew managed ATL's artificial intelligence lab for 8 years and served as a career military officer. The Army later loaned him to DARPA, where he was one of the original members of the strategic computing program that defeated Japan's Fifth Generation challenge. Mr. Frew is on the board of directors of the ISX Corporation.

James Larus is a research area manager at Microsoft Research. He manages several groups: Advanced Compiler Technology, studying compiler and language implement techniques and focused on techniques

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

for implementing modern, safe language and in compiling for highly parallel hardware; Human Interaction in Programming, which uses HCI techniques such as controlled user studies and ethnography to study software developers, testers, managers, and their teams to produce innovative software development tools that address human and social issues; Runtime Analysis and Design, which uses runtime program analysis, including hybrid static-dynamic analysis, statistical sampling, and heap analysis to improve software quality, security, and performance; Software Reliability Research, which applies program verification techniques and software measurement and modeling techniques to improve the quality of software; and Concurrency Research, which will explore ways to improve parallel programming. His research centers on Singularity, a project to focus on the construction of reliable systems through innovation in the areas of systems, languages, and tools: What would a software platform look like if it was designed from scratch with the primary goal of dependability? Singularity is working to answer this question by building on advances in programming languages and tools to develop a new system architecture and operating system (named Singularity), with the aim of producing a more robust and dependable software platform. Prior to joining Microsoft, Dr. Larus was an associate professor in the Computer Sciences Department at the University of Wisconsin-Madison. He has an M.S. and Ph.D. in computer science from the University of California at Berkeley and an A.B. in applied mathematics from Harvard University.

Greg Morrisett is Allen B. Cutting Professor of Computer Science at Harvard University. His current research interest is in the application of programming language technology for building secure and reliable systems. In particular, he is interested in applications of advanced type systems, model checkers, certifying compilers, proof-carrying code, and inline reference monitors for building efficient and provably secure systems. He is also interested in the design and application of high-level languages for new or emerging domains, such as sensor networks. Dr. Morrisett received his B.S. in mathematics and computer science from the University of Richmond (1989) and his Ph.D. in computer science from Carnegie Mellon University (1995). He spent about 7 years on the faculty of the Computer Science Department at Cornell University. In the 2002-2003 academic year, he took a sabbatical at Microsoft's Cambridge Research Laboratory. In January of 2004, he moved to Harvard University.

Walker Royce is the vice president of IBM's Worldwide Rational Lab Services. Mr. Royce joined Rational in 1994 and served as vice president of professional services from 1997 through IBM's acquisition of Rational in 2003. Over the last 10 years, he has managed large software engineer-

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

ing projects, consulted with a broad spectrum of Rational's worldwide customer base, and developed a software management approach that exploits an iterative life cycle, industry best practices, and architecture-first priorities. He is the author of *Software Project Management, A Unified Framework* (Addison Wesley Longman, 1998) and a principal contributor to the management philosophy inherent in Rational's Unified Process. Before joining Rational, Mr. Royce spent 16 years in software project development, software technology development, and software management roles at TRW Electronics & Defense. He was a recipient of TRW's Chairman's Award for Innovation for his contributions in distributed architecture middleware and iterative software processes in 1990 and was named a TRW Technical Fellow in 1992. He received his B.A. in physics from the University of California and his M.S. in computer information and control engineering from the University of Michigan.

Douglas C. Schmidt is a professor of computer science and associate chair of the computer science and engineering program at Vanderbilt University. He has published over 300 technical papers and six books that cover a range of research topics, including patterns, optimization techniques, and empirical analyses of software frameworks and domain-specific modeling environments that facilitate the development of distributed real-time and embedded (DRE) middleware and applications running over high-speed networks and embedded system interconnects. Dr. Schmidt has served as a deputy office director and a program manager at DARPA, where he led the national R&D effort on middleware for DRE systems. Dr. Schmidt has also served as the co-chair for the Software Design and Productivity Coordinating Group of the U.S. government's multiagency Information Technology Research and Development Program, which formulated the multiagency research agenda in software design. In addition to his academic research and government service, Dr. Schmidt has over 15 years of experience leading the development of ACE, TAO, CIAO, and CoSMIC, which are widely used, open-source DRE middleware frameworks and model-driven tools that contain a rich set of components and domain-specific languages that implement patterns and product-line architectures for high-performance DRE systems.

John P. Stenbit, NAE, is an independent consultant. He recently served as assistant secretary of defense for networks and information integration and as DoD's chief information officer. Mr. Stenbit's career spans more than 30 years of public and private-sector service in telecommunications and command and control. In addition to his recent service, his public service includes 2 years as principal deputy director of telecommunications and command and control systems, and 2 years as staff specialist

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

for worldwide command and control systems, both in the Office of the Secretary of Defense. Mr. Stenbit previously was executive vice president at TRW, retiring in May 2001. He joined TRW in 1968 and was responsible for the planning and analysis of advanced satellite surveillance systems. Prior to joining TRW, he held a position with the Aerospace Corporation involving command-and-control systems for missiles and satellites, and satellite data compression and pattern recognition. During this time, he was a Fulbright Fellow and Aerospace Corporation Fellow at the Technische Hogeschool, Einhoven, Netherlands, concentrating on coding theory and data compression. He has served on numerous scientific boards and advisory committees, including as chair of the Science and Technology Advisory Panel to the director of central intelligence and as a member of the Science Advisory Group to the Directors of Naval Intelligence and the Defense Communications Agency.

Kevin J. Sullivan is associate professor and Virginia Engineering Foundation (VEF) endowed faculty fellow in computer science at the University of Virginia, where he has worked since 1994. His research interests are mainly in software engineering and languages. He has served as associate editor for the *Journal of Empirical Software Engineering* and the *ACM Transactions on Software Engineering and Methodology*, and on the program and executive committees of conferences, including the ACM SIGSOFT Symposium on the Foundations of Software Engineering, the International Conference on Software Engineering, Aspect-Oriented Software Development, and ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. He and his students are broadly interested in the design and engineering of software-intensive systems, with an emphasis on the need for a value-based theory and practice of system design. Dr. Sullivan received his undergraduate degree from Tufts University in 1987 and the M.S. and Ph.D. in computer science and engineering from the University of Washington in 1994.

CSTB STAFF

Lynette I. Millett is a senior program officer and study director at the Computer Science and Telecommunications Board of the National Academies. She is currently involved in several CSTB projects, including a comprehensive exploration of biometrics systems, a study of emerging challenges to sustaining computing performance growth, and an examination of the Social Security Administration's electronic services strategy. Her portfolio includes significant portions of CSTB's recent work on software and on identity systems and privacy. She recently completed the study that produced *Software for Dependable Systems: Sufficient Evidence?* and she was the study director for the project that produced the reports

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Who Goes There? Authentication Through the Lens of Privacy and *IDs—Not That Easy: Questions About Nationwide Identity Systems*. She has an M.Sc. in computer science from Cornell University, along with a B.A. in mathematics and computer science with honors from Colby College, where she was elected to Phi Beta Kappa.

Joan D. Winston is a program officer for the Computer Science and Telecommunications Board of the National Academies. She is currently involved in CSTB projects assessing e-government strategy, the producibility of software-intensive systems, and the information technology R&D ecosystem. Before CSTB, she was an assistant director (Information Technology Team) at the Government Accountability Office. From 1998 to 2001, she was principal associate at Steve Walker and Associates, LLC, which managed early-stage venture funds focusing on information technology. From 1995 to 1998, she was director of policy analysis for Trusted Information Systems, Inc. From 1986 to 1995, she held various analytical and project direction positions at the Congressional Office of Technology Assessment (OTA) and was named an OTA senior associate in 1993. Before OTA, she worked briefly for the Congressional Research Service of the Library of Congress. Ms. Winston started her career as an engineer at the Charles Stark Draper Laboratory, Inc., in Cambridge, Massachusetts. She received an S.B. in physics and an S.M. in technology and policy, both from the Massachusetts Institute of Technology.

Margaret Marsh Huynh, senior program assistant, has been with CSTB since January 1999 supporting several projects. She is currently supporting the projects currently titled *Whither Biometrics*, *Wireless Technology Prospects and Policy*, *Advancing Software-Intensive Systems Producibility*, and *Assessing the Impacts of Changes in the Information Technology Research and Development Ecosystem.* She previously worked on the projects that produced the reports *Signposts in Cyberspace: The Domain Name Systems and Internet Navigation*; *Getting Up to Speed: The Future of Supercomputing*; *Beyond Productivity: Information Technology, Innovation, and Creativity*; *IT Roadmap to a Geospatial Future*; *Building a Workforce for the Information Economy*; and *The Digital Dilemma: Intellectual Property in the Information Age*. Ms. Huynh also assisted with the NTIA workshop on improving spectrum management through economic and other incentives (2006), the GAO/NRC forum on information resource management and the paperwork reduction act (2005), as well as the workshops on IT issues for the behavioral and social sciences. Prior to coming to the NRC, Ms. Huynh worked as a meeting assistant at Management for Meetings, April 1998-August 1998, and as a meeting assistant at the American Society for Civil Engineers from September 1996 to April 1998. Ms. Huynh has a B.A. (1990) in liberal studies with minors in sociology and psychology from Salisbury University, Salisbury, Maryland.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

C

Biosketches of Workshop Speakers

Cynthia Andres is a coauthor of *Extreme Programming Explained: Embrace Change, 2nd edition*. Her professional interests include team and individual psychology and facilitating change with large-scale transformative conversations. She holds a B.A. in psychology from Pacific Union College with advanced work in women's studies at the University of California at Santa Cruz and psychology at Portland State University.

Kristen J. Baldwin works in the Office of the Under Secretary of Defense, Acquisition, Technology and Logistics for the Director, Defense Systems. Ms. Baldwin's responsibilities span both systems engineering and systems integration functional areas. She leads the OSD oversight and implementation of software acquisition process improvement legislation, commonly referred to as Section 804. Ms. Baldwin formerly developed and managed the Triservice Assessment Initiative, which is a DoD tool for program managers to identify and mitigate system risk through independent expert assessments. Previous assignments in her career include serving as a science and technology advisor in the Army's Office of the Deputy Chief of Staff for Operations and Plans and at the Dismounted Battlespace Battle Lab at Fort Benning. Ms. Baldwin received a bachelors degree in mechanical engineering from Virginia Tech in 1990 and a master's in systems management from Florida Tech in 1995.

Kent Beck is the founder and director of Three Rivers Institute. His career has combined the practice of software development with reflection,

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

innovation, and communication. His contributions to software development include patterns for software, the rediscovery of test-first programming, the xUnit family of developer testing tools, and Extreme Programming. He currently divides his time between writing, programming, and coaching. Mr. Beck is an author and/or coauthor of *Extreme Programming Explained: Embrace Change,* 2nd Edition; *Contributing to Eclipse; Test-Driven Development: By Example; Planning Extreme Programming; The Smalltalk Best Practice Patterns;* and the *JUnit Pocket Guide*. He received his B.S. and M.S. in computer science from the University of Oregon.

Kris Britton is the director for the National Security Agency (NSA) Center for Assured Software. He has been involved with information assurance issues for the past 15 years, during which time he worked to establish commercial standards and programs to aid DoD customers in establishing trust in commercial products they purchase. He began his career as a commercial product evaluator in 1989, focusing on trust in operating systems and databases using the DoD standard (DoD 5200.28/Orange Book) and later helped to create the National Information Assurance Partnership and was named its first technical director. More recently he has been involved with software assurance issues, specifically working to evolve the NSA's software assurance paradigm to address today's evolving and complex IT environment.

Mary Ann Davidson is the chief security officer at Oracle Corporation, responsible for security evaluations, assessments, and incident handling. As a senior executive in the IT industry she brings both a military and a business background and in-depth experience with and perspective on industrial capacity to respond to Defense needs. She represents Oracle on the board of directors of the Information Technology Information Security Analysis Center (IT-ISAC) and is on the editorial review board of the *Secure Business Quarterly*. Ms. Davidson has a B.S.M.E. from the University of Virginia and an M.B.A. from the Wharton School of the University of Pennsylvania. She has also served as a commissioned officer in the U.S. Navy Civil Engineer Corps, where she was awarded the Navy Achievement Medal.

Joe Jarzombek serves as director for software assurance in the Policy and Strategic Initiatives Branch of the National Cyber Security Division within the Department of Homeland Security (DHS) and, as such, is the focal point on software integrity issues. He leads collaborative efforts in analyzing software life-cycle components, including people, processes, and technology and identifies areas for software quality and security improvement with a focus on development, acquisition, and support.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Mr. Jarzombek guides DHS initiatives in analyzing and resolving software challenges; supports the evolution of policy and guidance on software assurance, including assessment of federal policies, procedures, and evaluation schemes, such as the National Information Assurance Partnership. He functions as DHS coordinator for software quality and acquisition initiatives; working with other federal agencies, state agencies, and international allies to focus on identifying and specifying organizational software-related processes and software-enabled technologies to mitigate risks attributable to software. Mr. Jarzombek works with federally funded research and development centers (FFRDCs), consortiums, foundations, universities, and standards groups to coordinate relevant initiatives and leverage organizational resources to share best practices, tools, processes, and research to improve software assurance. He serves as DHS liaison on government/industry working groups and serves on NIST, IEEE, and ISO/IEC standards committees and advisory groups, and other executive groups to ensure software assurance needs are addressed in standards, best practices, process models and product lifecycle initiatives. He publishes best practices in software security on the Web site https://buildsecurityin.us-cert.gov/portal/ as information for developers and acquisition managers. In working with government/academic/industry groups, he leads team efforts to develop the Software Assurance Common Body of Knowledge, which is intended to provide a framework to recommend updates in curriculum to enhance IT acquisition and software-related education and training across the federal acquisition workforce curricula, within universities and colleges, and within industrial training programs. Mr. Jarzombek has an M.S. in computer information systems from the Air Force Institute of Technology, Dayton, Ohio; a B.B.A. in data processing and analysis from the University of Texas, Austin; and a B.A. in computer science from the University of Texas, Austin, where he was also an Air Force ROTC distinguished graduate.

Patrick Lardieri is manager of Distributed Processing Programs at the Lockheed Martin Advanced Technology Laboratory in Cherry Hill, New Jersey. He has spent over 10 years researching the suitability of open, standards-based middleware, operating systems, and networks for building distributed real-time systems. Recently, he has been leading Lockheed Martin's Software Technology Initiative, which is focused on developing technologies for managing the complexity of integrating large-scale software systems. He received a master's in electrical engineering from the University of Pennsylvania.

Gary McGraw, the CTO of Cigital, Inc., researches software security and sets technical vision in the area of software quality management.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

Dr. McGraw is coauthor of five best-selling books: *Exploiting Software* (Addison-Wesley, 2004), *Building Secure Software* (Addison-Wesley, 2001), *Software Fault Injection* (Wiley 1998), *Securing Java* (Wiley, 1999), and *Java Security* (Wiley, 1996). His new book, *Software Security: Building Security In* (Addison-Wesley), was released in February 2006. A world authority on software security, Dr. McGraw consults with major software producers and consumers. Dr. McGraw has written over 75 peer-reviewed technical publications and functions as PI on grants from Air Force Research Laboratories, DARPA, the National Science Foundation, and NIST's Advanced Technology Program. He serves on advisory boards of Authentica, Counterpane, and Fortify Software, as well as advising the Computer Science Department at the University of California, Davis, the Computer Science Department at the University of Virginia, and the School of Informatics at Indiana University. Dr. McGraw holds a dual Ph.D. in cognitive science and computer science from Indiana University and a B.A. in philosophy from the University of Virginia. He is a member of the IEEE Security and Privacy Task Force, and was recently elected to the IEEE Computer Society's board of governors. He writes a monthly security column for the magazine *IT Architect*, is the editor of "Building Security In" for IEEE's *Security & Privacy* magazine, and is often quoted in the press.

Richard W. Selby is the head of software products at Northrop Grumman Space Technology in Redondo Beach, California. He manages a 250-person software organization and has served in this position since 2001. Previously, he was the chief technology officer and senior vice president at Pacific Investment Management Company (PIMCO) in Newport Beach, California, where he managed a 105-person organization for 3 years. From 1985 to 1998, he was a full professor of information and computer science (with tenure) at the University of California at Irvine. Since 2004, he has held an adjunct faculty position at the University of Southern California Computer Science Department at Los Angeles. In 1993, he held visiting faculty positions at the MIT Laboratory for Computer Science and MIT Sloan School of Management in Cambridge, Massachusetts, and in 1992, he held a visiting faculty position at the Osaka University Department of Computer Science in Osaka, Japan. His research focuses on development and management of large-scale systems, software, and processes. He has authored over 100 refereed publications and given over 205 invited presentations at professional meetings. At Northrop, he led the $3 billion company to a successful enterprise-wide rating of Capability Maturity Model Integration (CMMI) level 5 for software. He served as the chief software engineer for the NASA Prometheus spacecraft to Jupiter. He also received the company's highest quality award, named after former President Tim W. Hannemann, for improvements in development, man-

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

agement, process, and quality. At PIMCO, he led the $1 billion company to be ranked as the fourth most innovative technology organization in financial services, according to *Wall Street & Technology*. At the University of California, Irvine, he coauthored an international best-selling book that analyzed Microsoft's technology, strategy, and management: *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*. The book, written with Michael Cusumano, has been translated into 12 languages, has 150,000 copies in print, and was ranked as a #6 best-seller in *Business Week*. He received his Ph.D. and M.S. degrees in computer science from the University of Maryland, College Park, Maryland, in 1985 and 1983, respectively. He received his B.A. degree in mathematics from St. Olaf College, Northfield, Minnesota, in 1981.

Alfred Spector, NAE, is currently an independent consultant working with IBM and a few small companies, and performing some government service. In his previous position as CTO and vice president of Strategy & Technology for IBM's Software Group, Dr. Spector was responsible for its technical and business strategy, standards, software development methodologies, advanced technology, and cutting-edge technical engagements. Prior to this position, Dr. Spector was vice president of IBM's worldwide services and software research, general manager of marketing and strategy for IBM's middleware business, and general manager of IBM's transaction software business. Dr. Spector was also the founder and CEO of Transarc Corporation, a pioneer in distributed transaction processing and wide-area file systems and a tenured faculty member in the Carnegie Mellon University computer science department. Dr. Spector received his Ph.D. in computer science from Stanford University and his A.B. in applied mathematics from Harvard University. He is recognized for his contributions to the design, implementation, and commercialization of reliable, scalable architectures for distributed file systems, transaction systems, and other applications. Dr. Spector is also an ACM and IEEE fellow and a recipient of the IEEE Kanai Award in distributed computing.

Werner Vogels is vice president and chief technology officer at Amazon.com, where he is responsible for driving the technology vision to continuously enhance the innovation on behalf of Amazon's customers at a global scale. Prior to joining Amazon, he worked as a research scientist at Cornell University, where he was a principal investigator in several advanced research projects that target the scalability and robustness of mission-critical enterprise computing systems. He has held positions of vice president of technology and chief technology officer in companies that handled the transition of academic technology into industry. Dr. Vogels holds a Ph.D.

Copyright © National Academy of Sciences. All rights reserved.

Summary of a Workshop for Software-Intensive Systems and Uncertainty at Scale
http://www.nap.edu/catalog/11936.html

from the Vrije Universiteit in Amsterdam and has authored a large number of articles for journals and conferences, most of them on distributed systems technologies for enterprise computing.

John Vu is a technical fellow at Boeing's engineering, operations, and technology. He has worked in various technical and management positions in Boeing, including computer-aided design and computer-aided manufacturing supporting the development of the 777 airplane, leading software and systems process improvement, and managing Boeing global software outsourcing. Prior to joining Boeing, Mr. Vu worked at Teradyne Semiconductor; Litton Industries, Motorola, and GTE. He led teams to build navigation and avionics systems (F-15 and Tomahawk cruise missile) and design the array processors for several signal processing systems (AWAC and several space exploration satellites). Mr. Vu is a visiting scientist at the Software Engineering Institute (SEI), where he focused on the development and implementation of several capability maturity models. As senior scientist at Carnegie Mellon University, he is conducting research on software trends in the industry, such as process improvement, e-business, and outsourcing. He has authored several benchmarking papers on these topics. He published over 40 technical papers on software and systems engineering disciplines, three books on software engineering and has presented papers at various software engineering conferences worldwide. He is a member of the Technical Advisory Board of IEEE Software, and adjunct faculty at Carnegie Mellon University and Seattle University.

Copyright © National Academy of Sciences. All rights reserved.

Copyright © National Academy of Sciences. All rights reserved.