Preliminary Observations on DoD Software Research Needs and Priorities

A Letter Report

Preliminary Observations on DoD Software Research Needs and Priorities: A Letter Report

Committee on Advancing Software-Intensive Systems Producibility, National Research Council

ISBN: 0-309-11877-8, 30 pages, , (2008)

Preliminary Observations on DoD Software Research Needs and Priorities

A Letter Report

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

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

Computer Science and Telecommunications Board

500 Fifth Street, NW
Washington, DC 20001
Phone: 202 334 2605
Fax: 202 334 2318
E-mail: cstb@nas.edu
www.cstb.org

Dr. Andre M. van Tilborg
Deputy Under Secretary of Defense for
Science and Technology
3030 Defense Pentagon
Room 3C913A
Washington, DC 20301-3030

Dear Dr. van Tilborg:

The Committee on Advancing Software-Intensive Systems Producibility is pleased to transmit its preliminary observations on Department of Defense (DoD) needs and priorities for software research, as well as its suggestions for a research agenda that would be executable within DoD's Science and Technology framework. This committee was appointed by the National Research Council (NRC) and convened under the auspices of the NRC's Computer Science and Telecommunications Board. *Preliminary Observations on DoD Software Research Needs and Priorities: A Letter Report* contains only the early and interim conclusions of the committee. These observations and other issues relating to the advancement of software-intensive systems producibility will be considered more comprehensively in the committee's final report.

A short summary presents the committee's main points. The first section of this letter report examines the significance of software to the defense mission. It is followed by an exploration of the role of industry and universities in software innovation, along with DoD involvement in software advancement. The report then provides some preliminary conclusions on the research challenge posed by software-intensive systems producibility.

Respectfully submitted,

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

NATIONAL ACADEMY OF SCIENCES • NATIONAL ACADEMY OF ENGINEERING • INSTITUTE OF MEDICINE • NATIONAL RESEARCH COUNCIL

Summary

The Committee on Advancing Software-Intensive Systems Producibility was appointed by the National Research Council (NRC) and convened under the auspices of the NRC's Computer Science and Telecommunications Board (CSTB) to assess the nature of the national investment in software research and, in particular, to consider ways to revitalize the knowledge and human resource base needed to design, produce, and employ software-intensive systems for tomorrow's defense needs.

This letter report responds to a request from the Department of Defense (DoD) for preliminary feedback from the committee regarding its observations on DoD needs and priorities for software research as well as suggestions for a research agenda that would be executable within the DoD's Science and Technology framework. In its response, the Committee on Advancing Software-Intensive Systems Producibility focuses on three specific questions: First, to what extent is software capability significant for DoD, and is it becoming more significant or less so? Second, will the advances in software producibility needed by DoD emerge unaided from the defense industrial base at a pace sufficient to meet evolving defense requirements? Third, in which technology areas should DoD invest in research to advance defense software producibility?

In response to the first question, software has become essential to all aspects of military system capabilities and operations. Manifesting military system capability in software offers formidable advantages such as enormous functional agility, unique scalability in capabilities and interoperation, and, of course, near-zero replication costs. Software producibility is an increasingly significant determinant of overall system capability. Although software is similar to other engineering disciplines in the need for project leaders to address trade-offs between risk and value, it is unique in that such trade-offs must be made in the context of the astonishingly rapid growth both in software technology capability and in the underlying hardware, networks, and storage systems. Additionally, it must do so in an environment where expectations about value (as manifest in systems capability, flexibility, and interoperability, for example) are also growing rapidly.

Given software's importance to DoD and the ongoing rapid advances in software capability worldwide, it is vital to ensure that the department can meet its software needs now and well into the future. A key question is, To what extent can DoD rely on industry—specifically the domestic defense industrial base—to innovate at a rate fast enough to allow it to fully meet future defense software requirements? That is, without explicit R&D stimulus from the DoD, will industry produce software innovations in areas of interest to DoD at the rate they are needed to keep up with DoD software requirements and enable the DoD to stay ahead of potential adversaries?

The committee's answer to this question is no: DoD's needs will not be sufficiently met through a combination of demand pull from the military and technology push from rapid innovation in the commercial sector. Defense has leading demand in certain critical areas: in these areas, which include architectural innovation, software assurance, and requirements management, defense needs are more demanding than the corresponding needs in commercial markets. Moreover, even where industry is highly innovative, firms may not have sufficient incentives to produce the kinds of technology and supporting tools necessary for others to take advantage of those software innovations. The committee sees a crucial role for the government in accelerating innovation in the core technologies related to software producibility.

The academic research community has traditionally worked on the core technical problems surrounding software producibility. The overall directions and priorities for sponsored research that leads to university-originated invention, however, are greatly influenced by funding levels and agency priorities. For example, DARPA's deliberately strong relationship with the IT research community, which began in the 1960s and endured for nearly 40 years, has had a profound influence on IT research priorities, the overall culture of computer science research, and the massive economic and national outcomes. Informal reports indicate that when DoD shifted funding away from university IT R&D, researchers in many areas key to DoD's software future scaled back their research teams and redirected their focus to other fields that were less significant to DoD in the long term. The impact of R&D cutbacks generally (excluding health-related R&D) has been noted by the top officers of major IT firms that depend on a flow of innovation and talent.

This letter report summarizes three technology areas where DoD has "leading demand." In these areas, DoD's requirements are more sophisticated and cutting-edge than those in the rest of the marketplace. Technological advancement would significantly benefit DoD's ability to produce the software it needs, providing a clear rationale for DoD investment in the needed research.

One area where the committee believes that new research would benefit DoD is the management of engineering risk in unprecedented large and ultra-scale systems. Such systems have engineering risks associated with early design commitments related to system functionality, non-functional attributes, and architecture. The research would focus on ways to mitigate these engineering risks at early stages of the process through new approaches to early validation, modeling, and architectural analysis.

The second area where DoD has leading demand and could benefit from technological advancement is software quality assurance for defense systems. Software assurance encompasses reliability, security, robustness, safety, and other quality-related attributes. Defense systems often include commercial off-the-shelf components and may involve global development—global sourcing is a reality for major commercial software products and, additionally, for commercial custom software and service provisioning. The needed research would focus on new ways for producers and consumers to create (and validate) a body of technical evidence to support specific claims in support of an overall judgment of fitness.

The third area, which is just as important as the first two, is the reduction of requirements-related risk in unprecedented systems without too great a sacrifice in systems capability. The challenge in this area has two parts. First, how can consequences of early commitments related to functional or nonfunctional requirements be understood at the earliest possible time during development? And, second, how can we make "requirements" more flexible over a greater portion of the system life cycle? The committee believes that the most useful research for DoD would look at ways to achieve early validation—for example, through modeling, protoptying, and simulation—and also look at how iterative development cycles can be supported more effectively and, from the standpoint of risk in program management, more safely.

These and other areas will be elaborated in greater technical depth in the committee's final report, which will answer in more detail the questions of where management attention and research activity should be focused.

Preliminary Observations on DoD Software Research Needs and Priorities

The Committee on Advancing Software-Intensive Systems Producibility (Appendix A) was appointed by the National Research Council (NRC) and convened under the auspices of the NRC's Computer Science and Telecommunications Board (CSTB) to assess the nature of the national investment in software research and, in particular, to consider ways to revitalize the knowledge and human resource base needed to design, produce, and employ software-intensive systems for tomorrow's defense needs.[1] The statement of task for the study is as follows:

> This study will bring together academic and industry software systems researchers, software and software tool vendors (suppliers), and systems integrators who comprise the community of skills required for future successes in complex software intensive systems required by the DoD. They will
> (1) Assess the emerging situation with respect to the national investment in relevant software research, the present state of and future requirements for tools for software production, testing and maintenance, and the adequacy of human resources;
> (2) Examine the needs, relationships and interdependencies expected of future DoD software research, development and maintenance needs, and consider what advances are needed for continuous improvements in the design, production and evolution of DoD software intensive systems;
> (3) Make recommendations to responsible agency, executive branch and legislative officials, and to the software technical community, about how to improve the present state of affairs and achieve future goals.

On November 16, 2007, following informal discussions with the sponsor, the National Research Council received a written request from the sponsor, the Department of Defense (DoD), for a short letter report in the spring 2008 time frame. This request, which was within the scope of the original charge to the committee, sought preliminary feedback from the committee regarding its observations on DoD needs and priorities for software research as well as research agenda suggestions that would be executable within the DoD's Science and Technology framework.

In this letter report, the Committee on Advancing Software-Intensive Systems Producibility focuses on three specific questions: First, to what extent is software capability important for DoD? Second, will the advances in software producibility needed by DoD emerge unaided from the defense industrial base (which includes both civilian and defense producers) at a pace sufficient to meet evolving defense requirements? Third, in which technologies should DoD invest in research to advance software producibility? A more comprehensive consideration of these and other questions relating to the advancement of software-intensive systems producibility will be provided in the committee's final report. The additional questions that will be addressed in the committee's final report include these: What actions can DoD take to accelerate the pace of innovation in software producibility, where such acceleration is necessary to meet DoD's particular needs? What are the most pressing research challenges that must be faced to enable DoD to meet its evolving needs? Another important consideration, related to the second question above, merits mention here, to spotlight its importance: What would be some potential consequences for DoD if the leadership in innovation for development of custom software were to move offshore?

[1]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.

This letter report builds on the discussion of technical and organizational issues captured in the committee's report *Summary of a Workshop on Software Intensive Systems and Uncertainty at Scale.*[2] This report also builds on additional briefings received by the committee. Appendix B contains the names of all these briefers.

The next section examines the significance of software in defense. The section after that explores the role of industry and universities in software innovation, along with DoD involvement in software advancement. The letter report finishes with some preliminary conclusions on the research challenge for software-intensive systems producibility.

THE SIGNIFICANT ROLE OF SOFTWARE IN DEFENSE

The pivotal role of software in defense has been noted in multiple studies. Software is a key enabler of net-centricity and of integrated systems of systems ("ultra-scale systems"). The ability to achieve these integrations and to maintain agility as these configurations of interconnected systems evolve is determined by the ability of the DoD to produce and evolve software. A software capability is thus a unique source of strategic and military advantage, and software producibility is a key component of military strength and capability. The Defense Science Board's (DSB's) Task Force on Mission Impact of Foreign Influence on DoD Software, which explored the essential role of software in defense, released its report in September 2007.[3] The report notes that "each year the Department of Defense depends more on software for its administration and for the planning and execution of its missions. This growing dependency is a source of weakness exacerbated by the mounting size, complexity, and interconnectedness of its software programs"[4] and "in the Department of Defense, the transformational effects of information technology (IT), joined with a culture of information sharing, called Net-Centricity, constitute a powerful force multiplier. DoD has become increasingly dependent for mission-critical functionality upon highly interconnected, globally sourced, IT of dramatically varying quality, reliability and trustworthiness."[5] Manifesting system capability in software presents a classic trade-off between risk and value. The research challenge, in short, is to mitigate engineering risk while enhancing value to the mission. The most obvious risks relate to unprecedented requirements and the need for innovative system architectures. The value is in the extent of systems capability embodied in software, the enablement of interoperation, and, perhaps most significantly in view of today's environment of asymmetric, agile warfare, the ease and flexibility of adaptation. Also, as scale and interconnection grow, a third dimension becomes increasingly important, which is the ability of a system to continuously evolve and interconnect with other evolving systems. As the role of software becomes more important, it becomes more urgent—indeed more strategic—to vigorously address this challenge.

The rapid growth of software in defense systems parallels the growing role of software in other domains and is a natural outcome of the special engineering characteristics of software. Software is uniquely unbounded and flexible, having relatively few limits on the degree to which it can be scaled in complexity and capability. Software's ability to be remotely and electronically upgraded adds even more value with respect to DoD's need to rapidly adapt to changes in adversary threats, mission priorities, technology, and the environment. The overall number of lines of DoD code in service has been increasing by more than an order of magnitude every decade; similar growth has been exhibited in

[2]NRC (National Research Council), *Summary of a Workshop on Software Intensive Systems and Uncertainty at Scale*, The National Academies Press, Washington, D.C., 2007.

[3]DSB (Defense Science Board), *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software,* Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, Washington, D.C., September 2007.

[4]Ibid., p. v.

[5]Ibid., p. vii.

individual, long-lived military systems.[6] In addition to this growth in size (as well as growth in other system aspects such as resource usage), overall system capabilities and complexities are increasing. A key attribute of software in any domain is that its potential capability is less constrained by the laws of physics and more by the ability of the human intellect to develop conceptual and engineering tools to gain insight into, master, and provide assurance of greater levels of complexity.

The value that software contributes to major systems is increasing rapidly and becoming more fundamental to system capability. The DSB Task Force report on Defense Software (2000)[7] illustrates this in the case of combat aircraft. The percentage of system functions performed by software rises from 8 percent of the F-4 in 1960, to 45 percent of the F-16 in 1982, to 80 percent of the F-22 in 2000.[8] Software has become essential to all aspects of military system capabilities and operations, and software-specific investment is critical to them.[9] Macroeconomic data show analogous growth in the role software plays in the commercial world. This is significant because commercial vendors are key contributors to the defense software supply chain—for Future Combat Systems,[10] for example, 27 million source lines of code (over 42 percent of the total delivered executable source lines of code) are commercial off-the-shelf (COTS) or open source.[11] It is also significant because software capability has become a strategic source of market differentiation in many industries, from financial services and health care to telecommunications and entertainment. A 2002 report by the National Research Council's Board on Science, Technology, and Economic Policy noted that since 1995 the IT and networking industries had accounted for 20 percent of the nation's economic growth, even though they accounted for only 3 percent of gross domestic product.[12] Comparable figures exist in the European Community.[13]

This critical role of software in defense is also noted in the more recent DSB Task Force report on foreign software, which states, "the DoD now relies upon networked, highly-interconnected systems for many mission-critical capabilities, and this reliance is projected to increase. The software in these systems is the key ingredient that provides much of the increased capability delivered to the warfighter, just as it represents the key factor in increased productivity and new capabilities for industry today. For the DoD, this advanced technology is a force multiplier."[14] Given software's importance to DoD, a vital

[6]Ibid., Figure 5. An estimate of overall growth in DoD software in terms comparable to those used for hardware is provided in Barry Boehm, "Managing Software Productivity and Reuse," *IEEE Computer,* September 1999: 111-113; also available as CSSE-TR-2000-508 at http://sunset.usc.edu/csse/TECHRPTS/2000/2000_main.html. Accessed February 20, 2008. See Figure 1 for growth of DoD code in service.

[7]DSB, *Report of the Defense Science Board Task Force on Defense Software*, Office of the Under Secretary of Defense for Acquisition and Technology, Washington, D.C., November 2000.

[8]Ibid., Table 3.3a. Available at http://www.acq.osd.mil/dsb/reports/defensesoftware.pdf. Accessed February 25, 2008.

[9]Boehm, Kind, and Turner quote an unidentified U.S. Air Force General, "About the only thing you can do with an F-22 without software is take a picture of it." Barry Boehm, Richard Turner, and Peter Kind, "Risky Business: Seven Myths About Software Engineering that Impact Defense Acquisitions," *Program Manager*, May 1, 2002.

[10]Future Combat Systems (FCS) is "the Army's modernization program consisting of a family of manned and unmanned systems, connected by a common network, that enables the modular force, providing our Soldiers and leaders with leading-edge technologies and capabilities allowing them to dominate in complex environments." U.S. Army, "Future Combat Systems." Available at http://www.army.mil/fcs/. Accessed March 3, 2008.

[11]DSB, *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software,* op. cit., p. 77.

[12] NRC, *Measuring and Sustaining the New Economy: Report of a Workshop*, The National Academies Press, 2002, p. 52. Available at http://www.nap.edu/openbook.php?record_id=10282&page=52. Accessed February 20, 2008.

[13]Stephen McGibbon, "Growth and Jobs from the European Software Industry," *European Review of Political Technologies*, December 2005, viz., "The latest EU25 data show that the [ICT] sector represents just over 5% of the EU GDP. But this 5% drives 25% of overall growth and about 40% of our increase in productivity, which is the fundamental source of new wealth in the economy."

[14]DSB, *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD*

question is how the department can ensure that it will be able to meet its software needs now and into the future. The next section explores this issue.

THE ROLE OF INDUSTRY AND UNIVERSITIES IN SOFTWARE INNOVATION

A key question regarding how DoD can meet its ongoing software needs is to what extent it can rely on industry, and specifically the domestic defense industrial base, to fully address its requirements now and into the future. This question needs to be answered in the context of two significant shifts in the environment of technology innovation. The first is the growing globalization of the software industry, with rapid gains in capability in Asia and Russia as well as steady gains in Europe.[15] The second shift is the reduction in recent years of direct DoD investment in advancing software capability in the defense industrial base and its supply chain.[16]

These two shifts raise additional considerations. First, to what extent can the DoD successfully address its mission assurance requirements[17] given the increasing extent to which DoD software will be developed in foreign countries? The cybersecurity dimensions of this question were taken up in the 2007 DSB Task Force report *Mission Impact of Foreign Influence on DoD Software*[18] and are therefore not covered in this letter report.[19]

The second consideration, which this letter report does address, is whether industry, without explicit R&D stimulus from the DoD, will produce innovations in areas of interest to DoD at the rate they are needed to meet the ongoing rapid growth[20] in DoD software requirements. The DSB Task Force findings regarding the shifting global center of gravity of software development are highly relevant to this question. DoD has particular requirements—for example, in high-performance embedded systems, large-scale systems with unprecedented architecture, highly interconnected systems, software assurance for critical systems, and the management of complex and evolving requirements. Moreover, these DoD requirements must be dealt with on systems that are both very large scale and have life-critical mission requirements. It is tempting but overly optimistic to conclude that DoD needs such as these will somehow be addressed in a sufficiently timely way through a combination of demand pull from the DoD

Software, op. cit., p. 12.

[15]For a general discussion of the increasing globalization of the software industry, see Association for Computing Machinery Job Migration Task Force, *Globalization and Offshoring of Software*, W. Aspray, F. Mayadas, and M. Vardi, eds., Association for Computing Machinery, Washington, D.C., 2006, and NRC, *Assessing the Impacts of Changes in the Information Technology R&D Ecosystem: Retaining Leadership in an Increasingly Global Environment,* forthcoming 2008.

[16]For example, the 2000 DSB Task Force report on defense software addressed the need to "strengthen and stabilize the technology base" in order to be able to leverage commercial technology and retain key researchers in DoD research organizations. DSB, *Report of the Defense Science Board Task Force on Defense Software*, Office of the Under Secretary of Defense for Acquisition and Technology, Washington, D.C., November 2000, pp. 30-34.

[17]Mission assurance includes test and evaluation related to a variety of issues such as cybersecurity, reliability, and other quality attributes.

[18]DSB, *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software,* op. cit. The full report is on the DSB Web site at http://www.acq.osd.mil/dsb/reports/ 2007-09-Mission_Impact_of_Foreign_Influence_on_DoD_Software.pdf. Accessed February 21, 2008.

[19]The committee notes, however, that the DSB Task Force recommends that DoD fund "a comprehensive Science and Technology Strategy and programs to advance the state-of-the-art in vulnerability detection and mitigation. . . . This program should monitor what markets are delivering, identify gaps between what the market is delivering and what DoD needs, and fill this gap." The Task Force also recommends that the most critical software be developed by cleared United States-based providers.

[20]Here, "growth" is used to capture the increase in both number and complexity of requirements, as well as the increasing distinctiveness (compared to the commercial sector) of some DoD requirements where DoD leads demand.

and technology push resulting from rapid innovation in the commercial sector. In many other industries, this may be a legitimate conclusion. In those industries, the best policy may be for DoD to follow the market. However, this is not generally true for all software technology, where DoD has leading demand in certain areas (see section "The Need for DoD Involvement in Software Advancement"). Moreover, even where industry is highly innovative, it may not have sufficient incentives to produce the technology and supporting tools necessary for others to take advantage of those software innovations. This is a key element of the need for government to play a role in accelerating core technological innovation in software producibility, especially where DoD requirements are more demanding than the corresponding requirements in commercial markets and where commercially driven advances therefore will not suffice—from the standpoint of either function or time to market—for DoD's purposes. In many of these areas, other countries are making concerted government investments in software technology development.[21]

In response to DoD's request, this letter report focuses on priorities for software research. This section, "The Role of Industry and Universities in Software Innovation," describes the importance of the DoD role in facilitating that research and eventual innovation, emphasizing four areas: (1) the diverse sourcing of the software components and technologies used in DoD software; (2) the enabling technologies (including software architectures) that are critical to DoD systems, and the importance of maintaining a national role in the definition of these technologies, (3) the role and effect of DoD involvement in R&D aimed at the advancement of software producibility; and (4) the role of academic R&D in software innovation generally.

DoD Supply Chains and Sourcing of Software Components and Technologies

The supply chain structure for modern defense software is significantly more complex, and international than it was even just a decade ago. This complexity is due to a combination of the powerful economic forces of globalization, a ubiquitous Internet, the availability of a trained workforce, and great demand for repetitive projects and associated infrastructure.[22] All of this is also coupled with the rapid maturation of key technological enablers, such as advanced networking, software architectures and frameworks, libraries, tools, programming and scripting languages,[23] and team collaboration capability

[21]See, for example, NRC, *Funding a Revolution: Government Support for Computing Research*, National Academy Press, Washington, D.C., 1999; NRC, *Innovation in Information Technology*, The National Academies Press, Washington, D.C., 2003; and Businesses Roundtable, *Securing Cyberspace: Business Roundtable's Framework for the Future*, Washington, D.C., May 2004. For discussion of other countries' initiatives in software, see, for example, *From Underdogs to Tigers,* Ashish Arora and Alfonso Gambardella, eds., Oxford University Press, Oxford, England, 2005, pp. 171-206; Rafiq Dossani and Martin Kenney, "The Evolving Indian Offshore Services Environment: Greater Scale, Scope and Sophistication," Sloan Industry Studies Working Papers, 2007 Number WP-2007-34, 2007. Available at http://www.industry.sloan.org/industrystudies/workingpapers/index.php. Accessed February 26, 2008; and Organisation for Economic Co-operation and Development (OECD), "China Will Become World's Second Highest Investor in R&D by End of 2006, Finds OECD," 2006. Available online at http://www.oecd.org/document/26/0,2340,en_2649_201185_37770522_1_1_1_1,00.html. Accessed February 26, 2008.

[22]With globalization, firms are increasingly seeking out and using offshore resources, whether these be offshore corporate facilities or outsource suppliers. A ubiquitous Internet facilitates geographically distributed workforces and suppliers, taking advantage of trained workforces in different countries. Historically, repetitive, routine projects are regarded as well suited for outsourcing or offshoring. But offshore suppliers are moving up the value chain, shifting from an initial emphasis on black-box testing to coding, design, and beyond. Use of foreign suppliers and offshore facilities introduce particular complexities for the DoD supply chain. A general discussion of globalization and information technology will be available in NRC, *Assessing the Impacts of Changes in the Information Technology R&D Ecosystem: Retaining Leadership in an Increasingly Global Environment,* forthcoming 2008.

[23]Advances in object-oriented programming languages with first-class encapsulation (Java and C# principally)

including process support, collaborative engineering environments, and advanced software build technologies. Beyond simply writing code, software today is much more about defining appropriate and scalable architectures; selecting, using, and adapting infrastructure such as frameworks, components, and libraries; deploying best practices and tools for collaboration, process support, and validation; and doing all of this in the context of an increasingly complex supply chain structure.

The complexity of software supply chain structures is evident in diverse sectors. A single enterprise software firm, for example, may develop software at dozens of separate sites around the world, and this software depends on infrastructure elements from dozens of vendors, each of whom may also have global operations. The richness of modern software technology—particularly object-oriented frameworks, emerging service-oriented architecture (SOA) concepts, team and collaboration tools, and process support—specifically enables the complexity of the modern supply chain structure for software systems. DoD is applying these ideas in the service-oriented Net-Centric Enterprise Services (NCES) program at DISA,[24] which builds on commercial SOA foundations. The framework and SOA concepts create interfaces within a system that are intended to enable participation by diverse suppliers, resulting in complex supply chains. Recent developments highlight the potential for enriched supply chains and the flexibility of diverse suppliers, but considerable technical and management challenges remain to deliver on this potential, particularly in a way that is effective for DoD.

It is becoming apparent first of all that the economic factors driving the supply chain structure include not only the direct cost of development, but also management agility (such as the ability to revisit choices in infrastructure, technology, and particular suppliers) and rapid access to specialized expertise (domain knowledge and requirements, code development, vendor components, testing and evaluation, process structuring, software architecture, and so on) and, secondly, that these factors combine to enable large firms to quickly adapt and enhance their business models to address competitive challenges. As international suppliers provide increasingly sophisticated functions (see below), the need to offer them incentives for improved quality increases accordingly.

An important element of the globalization phenomenon is the pace at which global suppliers outside the United States are moving up the value chain—that is, accounting for an increasing share of the overall value embodied in a product or service. Global suppliers, who in the early days focused primarily on offerings such as providing black-box testing services for Web-based software systems developed in the United States and elsewhere or on provisioning remote technical support capability, are now developing the software for those systems directly, as well as engaging in requirements analysis, architecture, and design for those systems. This commercial trend exacerbates DoD concerns about the mission impact of foreign influence on DoD software—namely, the risk of unwanted functionality in delivered software.[25]

As noted both in the workshop report issued by this committee[26] and in the recent SEI report on ultrascale systems,[27] these issues may be made worse by the extent of architectural innovation and overall scale required for modern interconnected defense systems. These systems experience a great deal of architectural risk due to the often long delay until the consequences of early engineering decisions are felt and understood. Additionally, central decision making and coordination can often focus risk and amplify

are essential to the success of the now-ubiquitous software frameworks in application areas ranging from e-commerce to mobile systems.

[24] See http://www.disa.mil/nces/about.html for more information.

[25]This tension was a focus of the DSB's 2007 *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software*. How to mitigate risk from COTS software, given global supply chains, continues to be a focus of DoD activities, including those of the National Security Agency (NSA), which is seeking a COTS strategy (proposal forthcoming).

[26]NRC, *Summary of a Workshop on Software Intensive Systems and Uncertainty at Scale*, The National Academies Press, Washington, D.C., 2007.

[27]SEI (Software Engineering Institute), *Ultra-large Scale Systems: The Software Challenge of the Future*, 2006.

rather than mitigate it. Also, overly conservative choices regarding how to measure progress and earned value can lead toward local optima but away from overall systems-scale success. Finally, overcommitment to particular requirements early in the process can result in lost opportunities for radical cost savings or capability improvements downstream. These risks could potentially be mitigated by both technological and process measures.

In the next section, on research showing promise for advancing software producibility, the committee suggests research that could lead to more effective approaches for large-scale systems with unprecedented architectures. These approaches include both technological advances and organizational measures. They could include, for example, incentives for (the many) participants in larger-scale systems development efforts and the development of better tools and practices to manage trade-offs between risk and value at different levels of structure within large projects. The key insight is that this is not a simple trade-off between higher levels of capability (mission "value") and the engineering risks that must be accepted to potentially achieve that capability. Rather, there are practices and techniques that can "push outward" and improve the trade-off curve by making the risk mitigation process more efficient through, say, iteration, prototyping, and technical and economic modeling. These ideas are also elaborated in that section, and more will be said in the committee's final report.

Technical Innovation Leadership

The United States has generally retained leadership in software innovation in two key technical areas.[28] One of these is the structure of the "stacks"—the internal industry standard interfaces that define key architectural elements of many systems such as network protocols and the principal e-commerce application programming interfaces (APIs) and frameworks.[29] The other leadership area is the set of core technical and design concepts for many of the infrastructural elements of the architectural stacks on which systems are built—such as operating systems, databases, application servers, and real-time kernels.

These "stacks" define the conventional interfaces and elements, generally in the form of de facto industry standards that exist within major complex systems.[30] In addition to their core technical layers (for example, the operating systems, databases, application servers, and network protocol stacks) these stacks also typically include key system frameworks and libraries, as well as the languages, tools, standards, and quality assurance technologies used in systems development. Such conventional aggregations of interfaces and components exist for most major categories of systems, including Internet-based systems of all kinds, e-commerce systems and Web services, mobile applications, and embedded and real-time control applications. They are emerging in areas ranging from robotic systems to data-intensive supercomputing. Over the years, DoD has attempted to codify its preferences regarding the components and interfaces in these conventional aggregates with efforts such as the Defense Information Infrastructure Common Operating Environment (DIICOE), the Joint Technical Architecture (JTA), and the System of Systems Common Operating Environment (SOSCOE). A particular challenge for DoD in

[28]The workforce has also been a significant factor in U.S. innovation leadership. Ceding workforce development to other nations is clearly a cost to the United States in terms of defense work per se and also in terms of the innovation ecosystem that industry, academia, and the government have nurtured. The success of additional investment by DoD in computing research is heavily dependent on corresponding investments in innovation and competitiveness throughout the nation.

[29]For example, owing to their broad utility, robustness, and low adoption risk, infrastructure, frameworks, and libraries for e-commerce are sometimes employed in defense systems that have no connection with e-commerce.

[30]The use of the term "stack" follows the industry convention for referring to the aggregation of infrastructural system elements organized according to conventional or standard interfaces. This aggregation is typically complex and, in fact, is not generally structured in a stack-like fashion. The term probably came from the layered structure of the familiar Internet network protocols. DoD uses the term "common operating environment" to convey similar meaning.

defining such standards is to keep up with rapidly evolving technology[31] and to select those for which certified components and/or trusted suppliers exist. Stacks in different systems and system categories may share elements and typically support diverse ranges of applications. Modern e-commerce frameworks, for example, are ubiquitous in Internet-based commerce, but they have also been adopted as internal coordination frameworks for large-scale DoD systems.[32]

The stacks contribute enormous value to software and systems projects that rely on them by allowing developers to leverage an enormous investment whose costs are spread across a wide base of users rather than taking on the full effort and expense of developing an entire software system from the ground up, and the extent of that value is constantly growing. As capabilities that previously were innovative become commonplace across firms, the task of advancing them is shifted from internal resources to external ones (outsource suppliers), enabling the firms to redirect their internal resources to new areas where they can differentiate themselves from their competitors. This is how, for example, the central database for many firms evolved from early network and hierarchical databases into relational transactional databases into virtualized application server capabilities wrapped around relational databases with the full functionality being provided by outside vendors. An issue for the DoD, however, as for other entities seeking to maintain leadership in software use and development, is how to effectively track the evolution of the conventional interfaces and architectures and not fall behind (see discussion of "surprise reduction," below). Another issue, more directly related to innovation, is how to work with the broad technology community to ensure that where DoD has leading demand its requirements can be met as the infrastructure evolves.

The modern stack, and particularly its upper levels, is enabled by a wide range of computer science and software engineering advances. The modern software application frameworks essential to Web-based systems, e-commerce, and graphical user interfaces of all kinds are enabled by the same advances in programming language design that led to languages such as Java, C#, and Ada95. Many of these advances are legacies of past DoD investment in computing technology R&D, primarily in the form of 6.1 and 6.2 extramural research funding.[33] The best-known example is the networking foundation provided by the Internet protocol suite that stemmed from defense investment in ARPANET, but lesser known examples such as the operating system architectures for network-based, distributed computing have had similar impacts.[34]

That global suppliers are moving up the value chain suggests the possibility that U.S. leadership may be eclipsed in many of the core technologies related to systems architecture, languages and tools, and software assurance, as well as with respect to key design elements of the software infrastructure.[35] This suggests a key question: Is there, in fact, strategic value in retaining U.S. leadership? In addition to making a number of recommendations to improve the overall quality of defense software and provide for more knowledgeable acquisition, the DSB Task Force report on foreign software asks this question in the particular area of software assurance, noting the essential requirement that the United States maintain

---

[31]Standards setting can involve a number of challenges and trade-offs, including a trade-off between currency and stability. This is why many firms follow industry conventions, which may evolve relatively more rapidly than formally ratified standards; this can further complicate matters for DoD.

[32]The extent to which DoD moves to purchase software services rather than to own or create certain types of software-intensive systems may be an indicator that it is moving toward a network-centric services support philosophy.

[33]6.1 is the DoD budget category for basic research; 6.2 is the category for applied research. "Extramural" research is that done outside government.

[34]See, for example: NRC, *Funding a Revolution: Government Support for Computing Research*, National Academy Press, Washington, D.C., 1999; and NRC, *Innovation in Information Technology*, The National Academy Press, Washington, D.C., 2003, especially the tire tracks diagram in Figure 1 of that report (pp. 6-7).

[35]Much of the technical progress now being made in software assurance for critical systems is in Europe (at companies such as Praxis, Esterelle, and others), due largely to sustained investment in creating the technologies and to extensive early experimentation in adoption by industry and government.

advanced capability for "test and evaluation" of IT products.[36] In other words, reputation-based credentialing of software needs more and more to be augmented by direct, artifact-focused means to support acceptance evaluation.[37] The Task Force recommends more effective direct evaluation by consuming organizations throughout the software supply chain, including better ways for producers to create software for which direct evidence of critical quality and functionality attributes can be provided. In other words, the Task Force concluded that test and evaluation should be supported by a broad range of software engineering technologies and interventions, not just those employed at the late test phase of development. Both the 2007 DSB Task Force report on foreign software and the DSB 2000 report on defense software also highlight the importance of commercial technology to DoD, including the essential elements (operating system, databases, application servers, and so on) of most of the predominant software stack architectures.[38]

In summary, previous studies highlight two priorities: (1) effective use of COTS technologies and components and (2) the ability to be as effective as possible at custom software engineering to rapidly achieve high levels of capability, then adapt with maximum agility to changes in the operating environment. A significant loss of U.S. leadership in either area could threaten DoD's ability to produce and assure the software it requires.

The Need for DoD Involvement in Software Advancement

Will the advances in software-intensive systems producibility needed by DoD emerge unaided from the defense industrial base at a pace sufficient to meet evolving defense requirements? This is a central question for software-intensive systems producibility. Many of the industry's advances are not undertaken with a focus on DoD priorities. As DoD relies more on the software capabilities of its supply chain,[39] this becomes more of a problem.[40] The issue is not areas where DoD has "unique" requirements, but the much broader category of areas where it has leading demand with respect to particular kinds of requirements, such as software assurance and risk-managed, unprecedented architectural design of the

[36]The phrase "test and evaluation" is widely used in DoD to refer to a broad range of activities related to validation and verification. For the purposes of this report, a broad interpretation is given, to include the more essential (and, arguably, cost effective) preventive measures related to requirements, architecture, design, and implementation.

[37]For a discussion of how to credential and certify software systems, see NRC, *Software for Dependable Systems: Sufficient Evidence?*, The National Academies Press, Washington, D.C., 2007. A new NRC study on improving processes and policies for the acquisition and testing of information technologies in the Department of Defense will focus on testing and evaluation of IT; see http://www.cstb.org for more information.

[38]"The System of Systems Common Operating Environment (SOSCOE) and the Integrated Computer System/Operating System (ICS/OS) [rely] predominantly on COTS and Open Source Software. The ICS/OS is almost 99% COTS/OS. The SOSCOE, essentially the 'middleware' for FCS, is almost 80% COTS/OS." DSB, *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software*, Office of the Undersecretary of Defense for Acquisition, Technology, and Logistics, Washington, D.C., September 2007, p. 77. See also DSB, *Report of the Defense Science Board Task Force on Defense Software*, Office of the Under Secretary of Defense for Acquisition and Technology, Washington, D.C., November 2000, p. 17.

[39]See, for example, DSB, *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software,* Office of the Undersecretary of Defense for Acquisition, Technology, and Logistics, Washington, D.C., September 2007, which grapples with the implications of foreign entities in its COTS supply chain.

[40]The overall supply of programmers is not a large problem, but the supply of highly capable program managers, software architects, and senior technical talent definitely is. See CSIS (Center for Strategic and International Studies), Defense-Industrial Initiatives Group, "An Assessment of the National Security Software Industrial Base," October 16, 2006. Available at http://www.diigcsis.org/pdf/Chao_SoftwareIndustrialBase_NDIASoftware.pdf. Accessed February 21, 2008.

kind required to create high-interconnectivity systems such as FCS, net-centric systems, and many other major defense platforms that have few commercial precedents or analogues.

The committee notes that, historically, the answer to this question has been that needed advances will not emerge at sufficient pace without an explicit and substantial investment by DoD. The evidence for this is DoD's role in initially creating and sustaining the innovation advantage of the United States in IT. This role was evident from the earliest days of computing during World War II until recently. The extent to which DoD funding has had a leveraged impact on innovation in information technologies throughout the economy is documented in several national studies.[41] It has enabled the United States to retain innovation leadership, but in recent years software issues critical to DoD have not been adequately addressed. There appears to be less direct DoD investment in advancing software capability in its industrial base and the associated supply chain. Additionally, this supply chain increasingly includes COTS, open source components, and globally-sourced components. Given the magnitude, cost, and criticality of DoD's software-intensive projects, the potential impacts from research and other initiatives focused on reducing the cost of supply-chain and other software problems in these projects could be substantial.

Threats to Leadership

There are at least three factors that could cause the United States to lose innovation leadership in software areas important to DoD. First, the DoD investment in software producibility has in recent years diminished considerably from its prior levels, which had been sustained over more than three decades.[42] The second factor is the ramping up of investment by foreign governments in their national IT capabilities, including in software.[43] The third factor is the inexorable trend of globalization, driven not just by simple cost economics but also by the potential for agility and rapid access to expertise, all enabled by modern software architectural advances coupled with advanced, Internet-based collaboration technologies. Of course, very strong shifts overseas have happened in other sectors, such as consumer electronics, and there is still debate regarding the strategic impact of these shifts. It is the committee's view, however, that the leveraged role of software and the particular special role of software in defense and national security systems of all kinds make this kind of shift much more consequential for defense software producibility and for U.S. ability to advance overall defense system capability.[44]

[41]See Kenneth Flamm, *Creating the Computer: Government, Industry, and High Technology*, Brookings Institution Press, Washington, D.C., 1988; NRC, *Funding a Revolution: Government Support for Computing Research*, National Academy Press, Washington, D.C., 1999; and NRC, *Innovation in Information Technology*, The National Academies Press, Washington, D.C., 2003.

[42]According to the annual R&D budget request data by component area published by the National Coordination Office for Networking and Information Technology Research and Development (NITRD), in FY 2007/2008 budget requests for Software Design and Productivity decreased by almost an order of magnitude from the requests in FY 2002/2003. See http://www.nitrd.gov/pubs/bluebooks/2002/bb2002-final.pdf, p. 35; http://www.nitrd.gov/pubs/bluebooks/2003/03BB-final.pdf, p. 37; http://www.nitrd.gov/pubs/2007supplement/07%20Supp%20Entire%20Book/07Supp_FINAL-022306.pdf, p. 20; and http://www.nitrd.gov/pubs/2008supplement/08Supp_FINAL-August.pdf, p. 20.

[43]See OECD, *Information Technology Outlook 2006*, available via http://www.oecd.org/document/10/0,3343,en_2649_37441_37486858_1_1_1_37441,00.html#TOCat. Accessed February 26, 2008**;** and NRC, *Assessing the Impacts of Changes in the Information Technology R&D Ecosystem: Retaining Leadership in an Increasingly Global Environment,* forthcoming 2008. See also From Underdogs to Tigers, Ashish Arora and Alfonso Gambardella, eds., Oxford University Press, Oxford, England, 2005, pp. 171-206; Rafiq Dossani and Martin Kenney, "The Evolving Indian Offshore Services Environment: Greater Scale, Scope and Sophistication," Sloan Industry Studies Working Papers, 2007, Number WP-2007-34, 2007. Available at http://www.industry.sloan.org/industrystudies/workingpapers/index.php. Accessed February 26, 2008; and OECD, "China Will Become World's Second Highest Investor in R&D by End of 2006, Finds OECD," 2006. Available at http://www.oecd.org/document/26/0,2340,en_2649_201185_37770522_1_1_1_1,00.html. Accessed February 26, 2008.

[44]See also DSB, *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on*

Economic Drivers for Federal R&D

There is no compelling return on investment (ROI) case for industry to produce the needed innovations that have a nonappropriable character. This is a familiar issue to those involved in defining industry-wide best practices, standards, and other commonalities. Many of the most important and highly leveraged, government-originated innovations are in the economic "commons." A recent NRC report on software and the economy notes that "the economic rationale for government investment is based on the nonappropriability of many significant IT innovations, including the most widely used idiomatic data structures and algorithms, as well as design and architectural patterns. Moreover, the IT industry relies on a number of technical and process commonalities or standards such as the suite of Internet protocols, programming languages, core design patterns, and architectural styles."[45] These innovations effectively raise everyone's "boat" in the same way as do government investments in bioscience, health care, and other strategically important scientific disciplines. This includes many of the most highly leveraged areas of software research such as improvements in abstraction mechanisms, design notations, programming languages, software analysis and model checking, basic algorithms, design patterns and architecture concepts, and other core techniques.

One of the significant challenges associated with creating value of this nonappropriable kind, which diffuses broadly into the engineering discipline and the economy, is the difficulty of measuring that value. Since the value has broad and diffuse benefits, direct measurement of ROI is probably not possible. Moreover, the way in which software is protected as intellectual property is often distinct from what is done in other fields (such as biomedicine). When dealing with a software system, it is often a combination of protected elements that are at issue, which makes the non-appropriable, yet valuable, aspects of the work even harder to identify. When research results are not appropriable, researchers are less likely to be able to secure patents. Thus, it can be hypothesized that direct revenues from licensing university-owned patents are likely to be significant underestimates of the value created by federally funded research, especially in the case of software-related university inventions.

One problem with looking to the software industry for needed R&D is that the industry—in terms of both firms and revenues—is shifting away from a focus on software products to a focus on integration, custom development, and other services.[46] More than half the revenues of software product companies are now coming from services rather than product sales, which tend to be the most scalable and profitable part of the business.[47] Additionally, the software-product sector is shrinking in numbers, going from more than 400 to less than 150 publicly listed software product companies on U.S. stock exchanges in the last 8 years.[48] With the industry in a consolidation mode and facing declining revenues from products, the ability of software product companies to do new research is weakening.

Generally speaking, moreover, the research done by software product companies is primarily focused on improving their products. While individual companies have the incentive to improve their

---

DoD Software, Office of the Undersecretary of Defense for Acquisition, Technology, and Logistics, Washington, D.C., September 2007.

[45]NRC, *Measuring and Sustaining the New Economy Software, Growth and the Future of the U.S. Economy: Report of a Symposium*, The National Academies Press, Washington, D.C., 2006, footnote 14, p. xvii. These commonalities include frameworks, service-oriented architecture concepts, infrastructure APIs, and so forth.

[46]For quite a few years, about two-thirds of global revenues in the software industry have actually been from services (such as custom software development, maintenance, IT consulting, and technical support) and only one-third of revenues have come from the product companies. See Michael A. Cusumano, *The Business of Software*, Free Press, New York, N.Y., 2004, p. 46, footnote 19, citing Standard & Poor's annual data.

[47]Thus, only about one-sixth of global software industry revenues (half of the one-third of revenues from products) is from product sales. See Michael A. Cusumano, "The Changing Software Business; Moving from Products to Services," *IEEE Computer,* January 2008, pp. 20-27.

[48]Michael A. Cusumano, "The Changing Software Business; Moving from Products to Services," *IEEE Computer,* January 2008, p. 22.

own productivity, there is less incentive to engage in research that is focused on industry-wide improvements, especially since (as is evident historically) many of those improvements come only after sustained commitments and much technical exploration. In a competitive market, individual companies generally have few market incentives for such investments. Even when productivity products for software development do find their way into the market, they are most often targeted for normal-scale enterprise IT systems, not for very-large-scale architectures, real-time embedded architectures, or other less conventional architectures that characterize DoD systems. These architectures, critical to DoD but not common elsewhere, may require to-purpose tools and practices or adaptations to existing tools and practices. The scale of the non-DoD market is too small to justify investment in advancing these areas unless DoD is a partner in that investment.

There may also be cases where industry has little economic incentive to acknowledge fundamental gaps in knowledge. This is not so unusual. One obvious example is ill-structured contracts, often from the government, where project difficulties may accrue to a contractor's economic benefit in the form of increased contract costs and profits, along with long-term revenue streams resulting from costly post-deployment repair and maintenance requirements. A second example is the consequence of poor measurement capability, particularly relating to quality and security—"assurance metrics" in the terminology of the 2007 DSB Task Force report. When metrics and observables are lacking, it is difficult to construct a business case for improvement of the underlying phenomena of concern—quality and security in this case. A third example is inadequate best practices. When metrics are weak, we must rely increasingly on best practices and process maturity to achieve product-related goals. Process compliance is relatively easier to achieve and certify than quality, but in software it is often a weak correlate.[49] Worse, fundamental improvements in best practices to enhance what can be achieved in terms of systems capability, productivity, quality, agility, and other characteristics are often in the category of non-appropriable innovations discussed above and thus may not be readily shared. These factors combine to lower industry economic incentives to address the producibility challenge.

Role of Academic R&D

The academic research community has traditionally addressed core technical problems related to software producibility. It is the committee's impression,[50] however, that in recent years many of the researchers in these areas have moved into other fields or scaled down their research efforts as a result of, among other things, DoD's having shifted funding priorities away from software-related R&D, apparently on the assumption that industry can address the problems without government intervention.[51]

More aggressive engagement with academic software researchers would enable DoD to take advantage of the unique value that university-based R&D would afford to DoD.[52]

[49]For more on the limitations of process compliance, see NRC, *Software for Dependable Systems: Sufficient Evidence?*, The National Academies Press, Washington, D.C., 2007.

[50]External views have shared the committee's impression. See John Markoff, "Pentagon Redirects Its Research Dollars," *The New York Times*, April 5, 2005, quoting academic researchers and industry research managers.

[51]The ROI required by industry tends to be very short term. As a result, incentives are lacking to support the longer-term, innovative research that is needed to support the escalating demands of the DoD.

[52]An excellent example of the payoff of academic research in middleware is provided in a paper tracing in considerable detail how Ph.D. dissertations in middleware evolved into major commercial middleware capabilities. See W. Emmerich, J. Sventek, and M. Aoyama, "The Impact of Research on the Development of Middleware Technology," *ACM Transactions on Software Engineering and Methodology* 17(4), 2008. Available at http://www.cs.ucl.ac.uk/staff/W.Emmerich/publications/impact_middleware/tosem.pdf. Accessed February 22, 2008.

Value from Academic R&D

The academic value proposition has several direct components: The first is workforce. University graduates are the core of the engineering workforce. The most talented and highly trained graduates—those who contribute to innovation in a primary way—tend to come from Ph.D. programs. More generally, the research community generates a steady supply of people—graduates at all levels—educated at the frontiers of current knowledge in important areas of specialization. The economics of these programs depend on externally funded research projects. It is perhaps too obvious to say so, but cleared individuals with top technical qualifications[53] are most likely to be graduates of U.S. universities.[54]

The second component is new knowledge. The research community is focused on creating new knowledge that provides solutions to, or at least progress on, important open problems. While industry does play a limited role in performing research relevant to fundamental open problems, there is no institution in the United States other than the research community, located primarily at universities, that focuses on broad, nonappropriable advancements to practice. Indeed, most major corporate labs that do not focus primarily on appropriable research (such as Bell Labs and Xerox PARC) have been restructured or scaled back in recent years.

Academic R&D is also a major generator of entrepreneurial activity in IT. The small companies in that sector have an important role in developing and market testing new ideas. The infrastructure to support these ventures is an important differentiator of the U.S. innovation system.[55] This infrastructure includes university intellectual property and people supported by university R&D projects. These companies may sometimes disrupt the comfortable market structures of incumbent firms, but not to the same extent as foreign innovation. Regardless, weak incumbents tend to fall by the wayside when there is any disruption. Strong incumbents become stronger. This constant disruption is a characteristic of IT. It is essential that the DoD itself be effective as a strong incumbent that is capable of gaining strength through disruptive innovations. The intelligence community's Disruptive Technology Office (DTO, now part of Intelligence Advanced Projects Research Agency) can be presumed to have been founded upon this model.

A third area of value provided by university-based R&D is surprise reduction. Computing technology is continuing to experience very rapid change, at a rate that has been undiminished for several decades. Given the rapid change intrinsic to IT, the research community (in academia and in industry, especially start-up companies) serves not only as a source of solutions to the hardest problems, of new concepts and ideas, and of trained people with high levels of expertise, but also as a bellwether, in the sense of anticipating technological changes. For software, the potential for surprise is heightened by a combination of the rapid growth of globalization, the concurrent movement up the value chain of places to which R&D has been outsourced, and the explicit investments from national governments and the European Commission in advancing national technological capability. Given the role of externalities in IT economics, it is not unreasonable to consider the innovation center of gravity changing rapidly in many key areas, which could shift control in critical areas of the technology stack described above. This is

[53]See DSB, *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software*, Office of the Undersecretary of Defense for Acquisition, Technology, and Logistics, Washington, D.C., September 2007, p. xi.

[54]Data on computer science and engineering Ph.D. production are available from the Taulbee surveys conducted by the Computing Research Association. For 2005-2006, over half of new Ph.D. students came from outside North America. Since about 2000, nonresident aliens have received the majority of Ph.D.'s granted by U.S. institutions. See http://www.cra.org/statistics/survey/0506.pdf, Table 5a and Figure 5. Accessed March 28, 2008.

[55]The information technology R&D ecosystem, including the role of academia and funding sources, is the focus of the NRC report *Assessing the Impacts of Changes in the Information Technology R&D Ecosystem: Retaining Leadership in an Increasingly Global Environment,* forthcoming 2008.

already happening in several areas of IT infrastructure, such as chip manufacturing. In this sense, the research community has a critical role in defense-critical areas that are experiencing rapid change.

The fourth component of the academic R&D value proposition is nonappropriable invention, as described above. This is one of the several forms of innovation carried out by the university research community. In a market economy, this is a role nearly unique to universities and similar institutions. For IT in particular, is essential to national competitiveness and to increases in marketwide value. Although the openness of university research is sometimes considered a negative factor with respect to the advancement of technology for national security, it also the case that universities have unique incentives, unlike industry, to advance the discipline even when the hard-won results are nonappropriable or difficult to fully appropriate. As noted above, it is evident from the history of the field that the advancement of IT and software producibility disproportionately depends on this kind of technology advance.[56] Of course, universities also create an enormous body of appropriable intellectual property that has the potential to be transitioned into practice.

DoD Influence on Academic R&D

The overall directions and priorities for sponsored research that leads to university-originated invention are greatly influenced by funding levels and agency priorities. For example, DARPA's deliberately strong relationship with the IT research community, which began in the 1960s and endured for nearly 40 years, has had profound influence on IT research priorities, the overall culture of computer science research, and the massive economic and national outcomes.[57] When DoD shifted funding away from university IT R&D, informal reports indicate that researchers in many areas key to DoD's software future scaled back their research teams and redirected their focus to other fields that were less significant to DoD in the long term. For example, relevant to the topic of this report, DoD budget requests for research in Software Design and Productivity fell from $60 million in FY 2003 (predominantly from DARPA) to $6.8 million in FY 2007; the FY 2009 request is $7.8 million (predominantly from OSD and service research organizations).[58] The impact of R&D cutbacks generally (excluding health-related R&D) has been noted by the top officers of major IT firms that depend on a flow of innovation and talent.[59]

[56]In the area of cybersecurity, for example, "there is an enormous distance between the development of a good idea and its widespread use. . .and traversing that distance often entails additional research activity that is significant in its own right." NRC, *Toward a Safer and More Secure Cyberspace,* The National Academies Press, Washington, D.C., 2007, p. 62.

[57]In addition to the NRC reports previously cited, see Arthur L. Norberg, Judy E. O'Neill, and Kerry J. Freedman, *Transforming Computer Technology: Information Processing for the Pentagon, 1962-1986*, The Johns Hopkins University Press, 2000.

[58]Annual R&D budget request data by component area are provided in the supplements to the President's Budget published by the National Coordination Office for NITRD. These indicate that budget requests for Software Design and Productivity were over $45 million for FY 2002 (mostly from DARPA) and $60 million for FY year 2003 (mostly from DARPA). However, for FYs 2007, 2008, and 2009, these requests had decreased to only $6.8 million, $4.3 million, and $7.8 million, respectively (from OSD and DoD service research organizations, not DARPA). See http://www.nitrd.gov/pubs/bluebooks/2002/bb2002-final.pdf, p. 35; http://www.nitrd.gov/pubs/bluebooks/2003/03BB-final.pdf, p. 37; http://www.nitrd.gov/pubs/2007supplement/07%20Supp%20Entire%20Book/07Supp_FINAL-022306.pdf, p. 20; http://www.nitrd.gov/pubs/2008supplement/08Supp_FINAL-August.pdf, p. 20; and http://www.nitrd.gov/pubs/2009supplement/NITRD-09Supp_FINAL-Web.pdf, p. 21. By contrast, NSF's FY 2009 budget request in this area was $70.8 million.

Developing recommendations that would strengthen the research infrastructure in areas of critical importance to national security was the focus of K. McKeown, L. Clarke, and J. Stankovic (organizers), "CRA Workshop on Research Related to National Security: Report and Recommendations," Computing Research Association, 2002, available at http://www.cra.org/CRN/articles/sept03/mckeown.html. Accessed February 26, 2008.

[59]See, for example, "The Future of Information Technology: Growing the Talent Critical for Innovation," Microsoft Research, July 2006. Available at http://research.microsoft.com/workshops/FS2006/papers/

Academic R&D, Looking Forward

There are some challenges to proceeding with a new program of academic R&D related to software-intensive systems producibility. These challenges relate generally to saliency, realism, and risk. University faculty tend to be aware of broadly needed advances, but they generally have inadequate visibility into the particular characteristics of the leading demand of very large-scale, complex industrial and military systems. This awareness is hindered by many things, including national security classification, professional connectivity, and cost, in the sense of time and effort required to move up the learning curve.[60] In a different domain, DARPA took a positive step in this regard by initiating the DARPA Computer Science Study Group, wherein junior faculty are given clearances and so are able to gain direct exposure to military challenge problems.[61] Several specific DoD programs have undertaken similar efforts to give faculty a domain exposure, often with great success. One example is the Command and Control University (C2U) created by the Command Post of the Future (CPOF) program, and the committee understands that prototype systems from this program are now deployed in Iraq.[62]

With respect to scale, companies such as Google and Yahoo!, and national laboratories such as Los Alamos, have developed collaborative programs to expose faculty and graduate students to high-performance computing systems and the software approaches being taken with those systems. These companies, like the DoD, have worked out a level of exposure that enables researchers to productively engage without compromising core intellectual property. DoD has a track record of success in this regard as well.

GENERAL RESEARCH AREAS OFFERING PROMISE FOR ADVANCING PRODUCIBILITY

In the preceding sections, the committee addressed the significant role of software in defense. It also provided an economic and historical rationale for the active involvement of DoD in fostering advances in the producibility of software and not resting its future on the hope that (1) industry will address its needs and (2) offshoring the center of gravity of innovation in software and software infrastructure will be acceptable from a national security standpoint.

In this section, the committee summarizes three technology areas where DoD has leading demand and where technological advancement would benefit DoD's capability to produce the software it needs. In the first two areas, a loss of innovation leadership—national competitive advantage—could adversely affect DoD's relative technological advantage in building the complex systems of the future. The committee notes, additionally, that all three areas will help to achieve the vision articulated by John Young (then acting Under Secretary of Defense for Acquisition, Technology, and Logistics) regarding prototyping early in the systems development process to reduce software-related engineering risk and support design validation, cost estimation, and specification refinement.[63]

TheFutureofInformationTechnology.pdf, accessed February 26, 2007; Michael D. Lemonick, "Are We Losing Our Edge?" *Time*, February 5, 2006, available at http://www.time.com/time/magazine/article/0,9171,1156575-1,00.html. Accessed February 26, 2008; and Chappell Brown, "Research Funding Teeters," *EE Times*, No. 1396, November 7, 2005, p. 1. See also NRC, *Rising Above the Gathering Storm: Energizing and Employing America for a Brighter Economic Future*, The National Academies Press, Washington, D.C., 2007.

[60]Additionally, many problems facing DoD may not be otherwise interesting to (or may even be considered "solved" by) academics. Moreover, the huge gap between a research paper and a finished piece of software is difficult to bridge.

[61]For more information on the DARPA Computer Science Study Group, see https://cs2p.ida.org/. Accessed February 26, 2007.

[62]For information on the Command and Control University (C2U), see http://www.cs.cmu.edu/~waisel/c2u.htm. Accessed February 26, 2008.

[63]John J. Young, Jr., Acting Under Secretary of Defense for Acquisition, Technology, and Logistics, "Memorandum for Secretaries of the Military Departments, Chairman of the Joint Chiefs of Staff, Commander, U.S.

These areas will be elaborated in greater technical depth in the committee's final report, which will respond in greater detail to the question of where attention and research should be focused. In this report, the committee provides an overview of three key high-level areas wherein advancement would offer DoD significant leverage.

Unprecedented Ultrascale Systems

In the foregoing discussion, the committee noted that software architecture is often the key to successful development. There is ample experience in commercial practice and, in some instances within DoD, to validate software architecture's importance. Despite some successful programs, however, DoD has not focused directly on this issue. Nevertheless, it is important that the DoD research agenda address software architectural issues such as those discussed in this section if DoD is to have the potential to achieve the operational benefits promised by these technology developments.

As noted above, defense systems are increasingly complex and interconnected. They consist of multiple components interacting and include functionality that cuts across multiple traditional defense functional areas. FCS is a good example. Previous platforms—conventional tanks and armored vehicles—generally combined in one co-located system an array of sensors, battle-command and human interfaces, and weaponry such as cannons. Platforms communicated with each other through human speech by radio and relatively simple telemetry. In the FCS model, by contrast, sensors, battle command, and weaponry are no longer co-located but are interconnected over a battlefield network. This enables, for example, battle command to exploit data from multiple sensors and to direct fire from cannons or mortars at those specific locations in theater from which the mission can be accomplished most effectively. This model enables a shooter to be guided by a multitude of geographically dispersed sensors. Through the use of autonomous and teleoperated vehicles, the model enables the positioning of sensors and shooters in high-risk locations. It thus affords tremendous power and agility to theater commanders. Additionally, because the architecture is fundamentally driven by interoperation requirements, if it is suitably managed, it can be a great enabler for joint (multiple military services, including air, land, sea, space, and cyber) and combined (international and coalition) warfare.

From the standpoint of systems engineering, however, this power comes at a significant price, which is the high level of complexity and engineering risk that comes from the extent of coupling required among the multitude of sensors, weapons, and battle command centers. For example, how can architectures be developed and validated to support the kind of local autonomy necessary for a vehicle to navigate effectively over mixed terrain? How can software and systems architectures be evolved, for example, as algorithms and machine-learning capabilities improve? Moreover, by specifying interfaces where testing or measurement is possible, by defining reusable components, and by separating critical from noncritical parts of the system, architecture plays an essential role in assurance.

FCS is but one example of the emerging architectural model of modern network-centric warfighting. And, indeed, the trend to networked distributed computing, which was foreseen by DARPA 40 years ago, leads not only to systems such as FCS but also to the horizontally linked, distributed computing models that characterize the most modern enterprise information systems. These systems have shifted from functionally focused departmental systems (analogous to a single tank or airplane) to process-focused enterprise systems (analogous to FCS and other network-centric systems concepts). Indeed, the process-focused systems now include links not only within the enterprise but also deep into supply chains (as was the case with the Boeing 777 global development project 20 years ago) and into the customer infrastructure.

As was described to the committee several times in the January 17, 2007, workshop on uncertainty at scale, the successful design of scalable architectures for this category of systems is extremely challenging. Presentations to the committee from companies such as Boeing and Amazon

Special Operations Command, Directors of Defense Agencies: Prototyping and Competition," September 19, 2007.

reinforce this point. The challenge derives, in large measure, from the unprecedentedness of such systems. While the lack of precedent creates engineering risk, it is also important to recognize that acceptance and management of this engineering risk is fundamental to success in achieving the potential new value that can be derived from the new concepts.[64] This is not to say that the value side of this equation is nonnegotiable. It seems plausible—indeed, nearly certain—that there will be situations where the risk is sufficiently high that reducing the target value is an appropriate decision. Moreover, there can be a significant political/managerial dimension to increased risk. That is, the concept of "engineering risk" as addressed routinely by technical experts in making internal design decisions is different from the concept of overall "project risk" as considered by executives and senior leaders, which relates to the likelihood of a project failing or falling short of goals. Two consequences are worth noting. First, in software there are techniques to mitigate and manage risk without necessarily compromising value. These techniques primarily involve various kinds of well-managed iteration, including prototyping, simulation, modeling, and various other approaches to the exploration of trade-offs among requirements, architectural, design, and quality assurance. System architecture should not be considered a given, and using architectures that are more partitioned can reduce risk at the expense of performance. A second consequence is that, in principle, the success or failure of any approach to risk reduction is best viewed over a larger set of projects—extrapolating from a small number of data points (or even, as is common, from anecdotes) is inappropriate but is, unfortunately, highly likely if the process is not understood in this added political/managerial dimension.

The engineering challenge, therefore, is not how to eliminate the risk but rather how to manage it successfully. In all engineering projects, and particularly software engineering projects, this usually means moving as early in the process as possible the time at which the consequences of provisional or "risky" decisions can be understood. If the consequences are understood only late in the process, then the costs of unwinding previous bad decisions may have become prohibitive, and instead the architecture becomes a legacy infliction that is constantly worked around. If, instead, consequences can be understood early, then it remains realistic to refactor and optimize the architecture.

From a research standpoint, there are a number of approaches that could be taken to address this issue. One is to further the development of modeling and simulation tools suitable to informing architectural decisions. This creates a try-before-you-buy approach to key architectural decisions, wherein architectural concepts are modeled using tools and analysis and testing can be done to assess scalability, performance, robustness, and resiliency to failures and attacks. In this context, the committee means something broader than the current theory and practice of testing and analysis (see footnote 34), which focuses on conformance of program behavior to specified behavior. The goal is to augment this with tests and analyses that provide information to support evaluation and validation of architecture concepts at the earliest possible stages of the process. Another approach is to develop audit and instrumentation tools to provide early data once architectures are designed and initially populated. Another approach is to develop and analyze a family of precedented "scalable architectural patterns" that could provide a well-understood infrastructure of building blocks out of which very-large-scale architectures could be designed. This could facilitate the use of multiple suppliers at the architectural level.[65] Additionally, if tools are in place that can support more aggressive restructuring, then a more principled approach can be taken to architectural design that includes iterative development, currently almost always impossible at the architectural level. This could also enable constructive response even to relatively late-breaking news about the consequences of early architectural commitments. Some combination of all these approaches will likely be necessary.

[64]A summary of briefers' presentations and the workshop discussion can be found in NRC, *Summary of a Workshop on Software Intensive Systems and Uncertainty at Scale*, The National Academies Press, Washington, D.C., 2007. For discussion of the limitations of testing, see National Research Council, *Software for Dependable Systems: Sufficient Evidence?*, The National Academies Press, Washington, D.C., 2007.

[65]Software engineering "patterns" may be defining a new dimension to "software reuse."

Assurance and Early and Continuous Validation

One of the great challenges for both defense and civilian systems is software quality assurance. Software assurance encompasses reliability, security, robustness, safety, and other quality-related attributes. Diverse studies suggest that overall software assurance costs account for 30-50 percent of total project costs for most software projects.[66] Despite this cost, current approaches to software assurance, primarily testing and inspection, are inadequate to provide the levels of assurance required for many categories of critical systems.[67] As systems grow in size, complexity, interconnection, and use of third-party components, these challenges will grow substantially. A further source of challenge is the dynamic nature of modern software architectures, including SOAs, architectures for autonomy and robotic systems, and other emerging architectural concepts.

To respond to assurance needs, a number of approaches are being explored in the research community, including model checking, static and dynamic analysis, verification, semantics-based testing, and others. Leading vendors have invested internally in developing various kinds of semantics-based tools and associated practices to address quality issues. Additionally, and perhaps most importantly, the core technologies of programming languages, frameworks, development tools, and process support are all being adapted to better support assurance requirements.[68] That is, assurance can no longer be thought of as a phase of systems development, but rather a consideration that needs to be taken into account from the earliest phases of systems conceptualization.

[66]In "Software Debugging, Testing, and Verification," *IBM Systems Journal* (41)1, 2002, B. Hailpern and P. Santhanam say, "In a typical commercial development organization, the cost of providing this assurance via appropriate debugging, testing, and verification activities can easily range from 50 to 75 percent of the total development cost." In *Estimating Software Costs* (McGraw-Hill, 1998), Capers Jones provides a table relating (percentage of defects removed; to percentage of development effort devoted to testing), with data points, including (90; 39), (96; 48), and (99.9; 58). In *Software Cost Estimation with COCOMO II* (Prentice Hall, 2000), B.W. Boehm, C. Abts, A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece indicate that the cost of test planning and running tests is typically 20-30 percent plus rework due to defects discovered. In *Balancing Agility and Discipline* (Addison Wesley, 2004), B. Boehm and R. Turner provide an analysis of the COCOMO II Architecture and Risk Resolution scale factor, indicating that the increase in rework due to poor architecture and risk resolution is roughly 18 percent for typical 10-KSLOC (KSLOC stands for thousand software lines of code) projects and roughly 91 percent for typical 10,000-KSLOC projects. (Note: COCOMO II, or COonstructive COost MOdel II, is a software cost, effort, and schedule estimation model.)

[67]The challenges relating to assurance were highlighted by several briefers to the committee. In addition, this issue is a core concern in DSB, *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software*, Office of the Undersecretary of Defense for Acquisition, Technology, and Logistics, Washington, D.C., September 2007, pp. 30-38.

The 2007 NRC report *Software for Dependable Systems* also addressed the issue of testing and noted that "Testing. . . will not in general suffice, because even the largest test suites typically used will not exercise enough paths to provide evidence that the software is correct nor will it have sufficient statistical significance for the levels of confidence usually desired." (p. 13)

In *Balancing Agility and Discipline* (Addison-Wesley, 2004), B. Boehm and R. Turner provide an analysis of the COCOMO II Architecture and Risk Resolution scale factor, indicating that the increase in rework due to poor architecture and risk resolution is roughly 18 percent for typical 10-KSLOC (KSLOC stands for thousand software lines of code) projects and roughly 91 percent for typical 10,000-KSLOC projects. This analysis suggests that improvements are needed in up-front areas as well as in testing, as well as supporting the importance of architecture research, especially for ultralarge systems.

[68]For example, the first-class encapsulation in languages such as Java, C#, and Ada95 permits much more effective analysis and inspection than the full-access model of the C language, Another example is development tools: The SLAM model checking tool used at Microsoft, for example, checks whether device driver code is consistent with a set of finite-state models that describe protocol requirements for the Windows device driver API. This checker eliminated the cause of most of the "blue screen" failures that were so evident in the 1990s.

Assurance is becoming more important across society for many reasons, including the increasingly critical role of software in national security systems, infrastructural systems, core financial systems, health care, utilities, transportation, mobile devices, and other sectors. The challenges in these sectors are diverse, but there are nonetheless common technical themes that can be addressed through a well-managed research program that addresses scalable techniques for error prevention and for evaluation and assurance and that also addresses opportunities to improve core software engineering technologies and practices that range from programming languages and frameworks to development tools and process and architectural principles. An important aspect of tool capability is the development of precise engineering models that represent intermediate steps from requirements to software code. These models can express key aspects or perspectives on an evolving system and can serve as links in a chain of evidence that creates traceability between requirements and implementation. Modern software engineering tools can assist in assuring consistency between models and implementations and also consistency among various kinds of models. This represents a significant shift from earlier techniques, where models were informal and tool-assisted consistency management was infeasible.

Software technology continues to evolve rapidly—and DoD, like other technology users who have leading demand, can be an active participant in fostering advances in the technology that will enable its needs to be better met. Such participation will likely include significant DoD involvement with, and encouragement of, research that addresses issues of unprecedented scale and quality assurance.

An important consideration in the development of such a research program, is that, in addition to the progress in achieving scalable technical approaches, there are also some developments in the operating environment that exacerbate the difficulties. One of these is the increasing complexity and interconnection of systems, as noted in the discussion on architecture. It is even more essential that the designs for these new ultralarge systems are framed with assurance considerations in mind. That is, assurability needs to be addressed at the earliest stages of the life cycle.

Another source of challenge is composability, resulting from the success of frameworks, the emerging service-oriented approaches, interoperation infrastructure, and other means by which large systems can be built from separately developed components.[69] These successes highlight the importance of developing software evaluation techniques that can support some degree of composability—the ability to achieve a system-level assurance on the basis of assurances achieved separately for separate components, including COTS black box components.[70]

Another source of challenge is the increasing success in carrying out large software development projects with teams in geographically dispersed locations. An important specific case of this phenomenon is global outsourcing, with various development and other software engineering tasks allocated to locations around the globe that can offer lower costs, greater expertise, or more timely and flexible access to available developers. Another important specific case is open-source software development. This phenomenon suffers from much misunderstanding but in any case would not exist without the technology and practices that support well-organized, but highly distributed development projects. Regardless, the new capabilities for global development and even distributed development within an organization heighten the importance of addressing the assurance challenges related to composability. These capabilities also heighten the importance of developing techniques for robust architecture concepts supported by API designs that support "assurability" of the separately developed components.

[69]For composability to work well, components may have to be designed to be reusable. This generally means added attention to the design of software interfaces that incorporate an appropriate level of generality and robustness. Thus, cost-effective development of reusable components is a related challenge.

[70]This issue of assurance for COTS software components is beyond the scope of this letter report. The subject has been taken up in a number of studies (including DSB's 2007 *Report of the Defense Science Board Task Force on Mission Impact of Foreign Influence on DoD Software)* and most recently by the NSA in its forthcoming COTS strategy proposal.

With respect to composability and geographic dispersion, there is a theme of "component acceptance evaluation." This phrase may suggest an after-the-fact approach whereby components are developed and later evaluated. An approach that involves "design for assurability" and that includes active effort by software component producers to create a body of evidence that can effectively support an efficient evaluation process may be beneficial in such circumstances. Thus, for example, an operational test and evaluation (OT&E) organization could initiate collaboration with system program managers early in the process to assist in achieving designs, processes, and implementations that can support efficient and effective operational test and evaluation.[71]

From Requirements to Criteria

Unlike many organizations that rely deeply on software, DoD must often create systems whose requirements and architecture are relatively less precedented than systems in other sectors. The lack of precedent creates engineering risk because it is difficult to predict the downstream consequences of particular early decisions on requirements or architecture. With precedented systems, there is greater ability to predict on the basis of experience in the form of both informal anecdote and analytic models. Why does DoD need to break away from precedent? The rationale is clear: If the new systems concepts and architectural innovations are successful, then DoD will be more effective in meeting its particular mission requirements related to the capability of defense systems and the often challenging characteristics of the operating environment for the software that is developed. In business terms, the added engineering risk is balanced by the potential for increased value.

The opportunity for researchers, therefore, is to assist DoD in reducing engineering risk without compromising the potential value gained in the form of systems capability, flexibility, interoperation, and so on. There are many different kinds of interventions in the process that might shift the balance more toward value. One example is techniques to support early validation of choices made during requirements engineering. In old-style acquisition processes, the operational consequences of a particular decision regarding requirements may not be understood until years after the decision was made, coming to light, for example, during an operational test and evaluation process. Modern acquisition processes, by contrast, can incorporate storyboarding and prototyping, for example, to achieve an early validation of potential requirements and associated architectural commitments regarding the structure of the human–system interaction.[72] This affords early opportunity to adjust requirements in response to early feedback from key stakeholders. It may also be true that the models and simulations used to connect requirements to the eventual software system—all during its development—can be used to drive out the bad elements in stated requirements (often derived from naïve overspecification) earlier in the acquisition process.

One of the challenges of this early validation and requirements adjustment approach is measurement of progress. How can developing and discarding a prototype, for example, be counted as

[71]It is important to note, however, that all components have properties that are not defined as part of their interface (information hiding). This leads to a situation where, in practice, component providers argue that they can change anything not defined as part of the interface, while component users over time become to depend on some of these undefined aspects, making them a de facto part of the interface. This is a challenge that must also be addressed when evaluating components and generating the body of evidence to support the evaluation. R Walker and G Murphy refer to this implicit context as "extraneous embedded knowledge" (R. Walker and G. Murphy, "Implicit Context: Easing Software Evolution and Reuse," *Proceedings of the 8th ACM SIGSOFT International Symposium on Foundations of Software Engineering: Twenty-First Century Applications*, ACM (Association for Computing Machinery), New York, N.Y., 2000). See also D. Garlan, R. Allen, and J. Ockerbloom, "Architectural Mismatch or Why It's Hard to Build Systems out of Existing Parts," *Proceedings of the 17th International Conference on Software Engineering*, May 1995, pp. 179-185. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.3774. Accessed April 23, 2008.

[72]See NRC, *Human-System Integration in the Systems Development Process*, The National Academies Press, Washington, D.C., 2007.

progress? From an engineering perspective the value can be understood: It is the extent to which the prototype serves to reduce the uncertainty associated with particular engineering commitments. The question is, How can this, from the standpoint of objective measures of progress such as earned value management systems,[73] be accomplished easily on a routine basis? The more general issue is, What technical steps can be taken to make iterative development more routine? For iterative development, the key to successful application of the spiral development model[74] is an identification and prioritization of "risks" in the sense of engineering risks.[75]

Iterations in requirements, design, and, indeed, diverse aspects of implementation can be conducted, often with significant concurrency, with each iteration addressing some subset of the overall set of engineering risks. For example, in human-centered systems, it may be prudent to address the human–system interaction design in early iterations, since the outcomes of those iterations could affect architectural decisions later. One might expect the result of an iteration to be some segment of code to be delivered as part of a deliverable artifact. But in the prototyping example above, this is not the case. Instead, the result can be considered as evidence, which is constructed to support a particular claim concerning a design commitment. In this approach, evaluation of progress toward milestones can be based on an accumulation of evidence of various kinds to support the soundness of various design decisions.

This idea also applies to the identification of specific requirements. Starting, for example, with a general definition of the scope of a system, iterations involving various stakeholders can lead to evidence supporting particular requirements. This may be in the form of more restricted definitions of scope, or scenarios and use cases, or identification of domain objects and relationships, or architectural commitments, and so on.

From the standpoint of systems management, this iterative approach could be used to advance the point in the process at which the high-order engineering risks are addressed and to reward development teams that can resolve these risks early through evidence-based early validation of key design commitments. This approach shifts the focus from traditional ordering of requirements, architecture, design, implementation, and so on to an ordering based on engineering risk. This enables, for example, a program manager to focus in early stages on architectural feasibility, if that is the focus of risk, or on human–systems interface, if that is the focus of risk. Thus a systems development effort could start by defining the scope of the mission and include in its iterative process the refinement of requirements concurrently with architectural risk reduction and commitment.

This approach could have some value for ultrascale systems, which are often in a constant state of development, in the sense that they are constantly evolving. This is not an unusual scenario—most successful commercial software products, from desktop productivity applications to infrastructural components such as databases and application servers, are developed in what amounts to an endless series of iterations of scoping, feature definition, design, implementation, and release, all followed by feedback in support of defining priorities for the next release cycle.[76] With ultrascale systems a similar kind of

[73]Earned value management is a project management practice used across DoD as well as in the federal government and the commercial sector. According to DoD, it is "the use of an integrated management system that coordinates the works scope, schedule, and cost goals of a program or contract, and objectively measures progress toward these goals." See http://www.acq.osd.mil/pm/faqs/faq.htm. Accessed March 3, 2008.

[74]See, for example, Barry Boehm, "Spiral Development: Experience, Principles, and Refinements," Wilfred J. Hansen, ed., CMU/SEI-2000-SR-008, July 2000. Aavailable at http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00sr008.pdf. Accessed February 28, 2008.

[75]Engineering risk, roughly, is the product of the probability of an unexpected outcome and the consequences of that particular outcome. Or, more generally, it is the sum of a set of such terms for various possible outcomes.

[76]To be sure, many large-scale commercial systems succeed because they are designed to be self-healing or fault resistant—they do not have to work flawlessly 100 percent of the time, so long as critical functions work correctly. For software, these faults include design faults as well as the transient or intermittent faults affecting attached physical components in a system. Thus, software can have varying levels of criticality depending on the context in

endless iteration is possible, except that the components of such systems are likely all evolving concurrently. In these circumstances other techniques are needed. Rather than trying to refine and reduce the risk in any given area, research is needed into design precepts that support adaptivity[77] and agility to allow ultrascale systems to evolve.

The research questions should be evident from the foregoing. What are the particular kinds of evidence that might be produced? How does this evidence relate to the modeling formalisms that are important for architecture, design, and assurance (as noted in the previous two area descriptions)? How can iterative processes be defined so they are feasibly executed for mainstream defense software development? How must these processes be adapted for the constant development cycles of ultrascale (or system-of-systems) programs? This may appear to be a management topic, but there is in fact substantial technical challenge in the identification and evaluation of the particular kinds of models that can be used to frame the evidence that is created.

Research Topics in the Context of the Overall Study

The foregoing overview of three research topics provides the committee's interim feedback to DoD on research that holds promise for improving the producibility of software-intensive systems. These and additional research areas that emerge from committee deliberations will be addressed in more depth in the final report for this study.

Although a clear message in this letter report is that additional, focused research is necessary and has the potential to alleviate many difficult problems, merely funding additional research will not completely solve the software producibility problem for DoD. A multifaceted, carefully managed, ongoing approach is necessary. This is because software technology, unlike other engineering disciplines rooted in physical artifacts, is essentially unbounded. Indeed, it is characteristic of both software technology and defense program requirements that it impossible to completely solve this problem or even fully routinize the core value creation in software development. We can, however, and will (with proper support) make genuine progress. Continuous progress is in fact a necessity given the growing need for software, the value contribution of software, and thus the demands placed on expertise and experience in software producibility. Through this growth cycle there will likely always be some critical dimensions of software engineering that appear to be primary sources of cost and risk. Understanding the nature of progress in software technology and producibility, as contrasted with progress in other engineering disciplines, is essential to create appropriate expectations for how these risks can be managed and, more generally, how the risk/value balance can be addressed as requirements and technology continue to evolve.

As part of an overall research program, the issues that the DoD faces must be addressed in the short to medium term as well. These intermediate-term results are a risk mitigator for the overall research investment, and the feedback from those results can provide midcourse guidance for the longer-term work. The issues that research funding alone cannot solve include institutional as well as technical problems, such as industry incentive structures, the quality and quantity of technical talent within DoD and its software industrial base, and development practices. Addressing these issues, as well as issues of research management and strategy, may have as much or more impact as specific technical breakthroughs in a research project. Particularly with respect to industry, a complementary approach that adds development-stage investments for industrial-base and system-of-systems toolsets, knowledge, and capabilities would be beneficial. These and related considerations will be covered in the committee's final report.

---

which it operates.

[77]See NRC, *Toward a Safer and More Secure Cyberspace*, Washington, D.C., The National Academies Press, 2007. The report has a focus on security research, looking at security holistically and including research to address adaptability and resilience.

Appendix A

Members of the Committee on Advancing Software-Intensive Systems Producibility

WILLIAM SCHERLIS, Carnegie Mellon University, *Chair*
ROBERT BEHLER, MITRE Corporation
BARRY W. BOEHM, University of Southern California
LORI CLARKE, University of Massachusetts at Amherst
MICHAEL CUSUMANO, Massachusetts Institute of Technology
MARY ANN DAVIDSON, Oracle Corporation
LARRY DRUFFEL, Software Engineering Institute
RUSSELL FREW, Lockheed Martin
JAMES LARUS, Microsoft Corporation
GREG MORRISETT, Harvard University
WALKER ROYCE, IBM
DOUG 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 (until November 2007)
MORGAN MOTTO, Senior Program Assistant (beginning December 2007)

Appendix B
Briefers to the Committee on Advancing Software-Intensive Systems Producibility

SEPTEMBER 27-28, 2006, FIRST COMMITTEE MEETING

Annmarie Bunts, Lockheed Martin Systems Integration
Grady Campbell, Software Engineering Institute
Larry Druffel, SCRA (retired)
Robert Gold, Office of the Deputy Under Secretary for Science & Technology/Information Systems
Ronald T. Kadish, Booz Allen Hamilton
James Larus, Microsoft Research
Robert Nesbit, MITRE Center for Integrated Intelligence Systems
Linda Northrop, Software Engineering Institute
Tom Rodgers, Lockheed Martin
Walker Royce, IBM
Andre van Tilborg, Office of the Deputy Under Secretary for Science & Technology/Information Systems

JANUARY 17, 2007, WORKSHOP ON UNCERTAINTY AT SCALE

Cynthia Andres, Three Rivers Institute
Kristen J. Baldwin, Office of the Under Secretary of Defense, Acquisition, Technology, and Logistics
Kent Beck, Three Rivers Institute
Kris Britton, National Security Agency, Center for Assured Software
Mary Ann Davidson, Oracle Corporation
Joe Jarzombek, Department of Homeland Security
Patrick Lardieri, Lockheed Martin
Gary McGraw, Cigital, Inc.
Richard W. Selby, Northrop Grumman
Alfred Spector,[*] Google, Inc.
John Vu, Boeing
Werner Vogels, Amazon.com

APRIL 24, 2007, THIRD COMMITTEE MEETING

Thomas Blann, Office of the Director of Operational Test and Evaluation

[*]As of November 2007. At the time of the workshop, Dr. Spector was an independent consultant.

Appendix C

Reviewers for This Letter Report

This letter 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 letter report as sound as possible and to ensure that the letter 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:

John M. Gilligan, SRA International
Anita K. Jones, University of Virginia
Butler Lampson, Microsoft Corporation
Alan MacCormack, Harvard University
David Notkin, University of Washington
Harry D. Raduege, Jr., Deloitte Center for Network Innovation
Alfred Z. Spector, Google, Inc.

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 letter report before its release. The review of this letter report was monitored by William H. Press, Los Alamos National Laboratory. Appointed by the NRC, he was responsible for making certain that an independent examination of this letter report was carried out in accordance with institutional procedures and that all review comments were carefully considered. Responsibility for the final content of this letter report rests entirely with the authoring committee and the institution.