Summary of a Workshop on Software Certification and Dependability

Committee on Certifiably Dependable Software Systems, National Research Council

ISBN: 0-309-54619-2, 57 pages, 8 1/2 x 11, (2004)

Summary of a Workshop on Software Certification and Dependability

Committee on Certifiably Dependable Software Systems

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 PRESS 500 Fifth Street, N.W. Washington, DC 20001

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

Support for this project was provided by the National Science Foundation, the National Security Agency, and the Office of Naval Research. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor.

International Standard Book Number 0-309-09429-1 (Book)
International Standard Book Number 0-309-54619-2 (PDF)

Cover designed by Jennifer M. Bishop.

Additional copies of this report are available from:

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

Copyright 2004 by the National Academy of Sciences. All rights reserved.

Printed in the United States of America

Preface

Systems on which the safety or security of individuals may depend are frequently subject to certification: a formal assurance that the system has met relevant technical standards, designed to give confidence that it has some specific properties—for example, that it will not unduly endanger the public. Today, certification of the dependability of a software-based system frequently relies at least as heavily on assessments of the process used to develop it as it does on the system's observable properties. While these assessments can be useful, few would dispute that direct evaluation of the artifact ought to provide a stronger kind of assurance than the credentials of its production methods could hope to provide. Yet the complexity of software systems, as well as their discrete nature, makes them extremely difficult to analyze unless great care has been taken with their structure and maintenance.

To further understand these and related issues, the High Confidence Software and Systems Program at the National Coordination Office for Information Technology Research and Development initiated discussions with the Computer Science and Telecommunications Board (CSTB) of the National Research Council (NRC). These discussions resulted in agreement to undertake a study to assess the current state of certification in dependable systems, with the goal of recommending areas for improvement. Initial funding for the project was obtained from the National Science Foundation, the National Security Agency, and the Office of Naval Research. The Committee on Certifiably Dependable Software Systems was appointed to conduct the study.

The task of the committee is to identify the kinds of system properties for which certification is desired, describe how that certification is obtained today, and, most important, determine what design and development methods, as well as methods for establishing evidence of trustworthiness, could lead to systems structures that are more easily certified.

To accomplish its mission, the committee divided this study into two phases: a framing phase and an assessment phase. This report is the outcome of the first phase, the framing phase, which included a public workshop organized by the committee and attended by members of industry, government, and academia. Held on April 19-20, 2004, the workshop featured a variety of participants invited to present their views on issues surrounding certification and dependability (see Appendix A for the workshop agenda). Six panels were organized, and each panelist gave a short presentation that addressed the theme of the panel. The workshop panelists are listed in Appendix B. Each panel session was followed by an extensive discussion involving all of the workshop participants and moderated by one or two committee members. The committee met three times: once to plan the workshop, then to hold the workshop, and, last, to distill information from the workshop and develop the report. This report is the committee's summary of the panelists' presentations and the discussions that followed.

Although the summary is based on presentations and discussion at the workshop, the participants' comments do not necessarily reflect the views of the committee, nor does the summary present findings or recommendations of the National Research Council. In fact, the committee took care in writing this report simply to summarize the discussions, and to avoid any bias or appearance of bias in favor of one opinion or another. Because it did not seem sensible to attempt a distillation across panels, the committee tried to record something of the spirit of each individual panel session. Nor is this report intended to be complete; topics that were not discussed at the workshop are not mentioned, however important they might be.

In the second phase of the study, the committee will analyze the information gathered in the workshop and summarized here, along with information and input it gathers from other experts and related studies. This assessment phase will deliver a final report (planned for release in 2005) with findings and recommendations from the committee.

The Committee on Certifiably Dependable Software Systems consists of 13 members from industry and academia who are experts in different aspects of systems dependability, including software engineering, software testing and evaluation, software dependability, embedded systems, human-computer interaction, systems engineering, systems architecture, accident theory, standards setting, key applications domains, economics, and regulatory policy (see Appendix C for committee and staff biographies).

The committee thanks the many individuals who contributed to its work. It appreciates the panelists' willingness to address the questions posed to them and is grateful for their insights. The study's sponsors at the National Science Foundation, the National Security Agency and the Office of Naval Research have been most supportive and responsive in helping the committee to do its work. The reviewers of the draft report provided insightful and constructive comments that contributed significantly to its clarity.

The committee is particularly grateful to the CSTB staff: Lynette Millett, program officer, who as the study director for this project has provided excellent advice and assistance throughout; Phil Hilliard, research associate, whose work in note-taking and summarizing discussions, and in obtaining and organizing materials for the committee, has been invaluable; and Penelope Smith, senior program assistant, who deftly handled all kinds of administrative issues, including most of the arrangements for the workshop and associated meetings. The success of the workshop is a testament to their commitment and hard work. Susan Maurizi from the Division on Engineering and Physical Sciences' editorial staff and Cameron Fletcher made significant editorial contributions to the final manuscript.

Daniel Jackson, *Chair*
Committee on Certifiably Dependable Software Systems

Acknowledgment of Reviewers

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

Anthony Hall, Independent Consultant
John C. Knight, University of Virginia
William Scherlis, Carnegie Mellon University
William Stead, Vanderbilt University
Jeannette Wing, Carnegie Mellon University

Although the reviewers listed above have provided many constructive comments and suggestions, they were not asked to endorse the conclusions or recommendations, nor did they see the final draft of the report before its release. The review of this report was overseen by Daniel P. Siewiorek, Carnegie Mellon University. Appointed by the National Research Council, he was responsible for making certain that an independent examination of this report was carried out in accordance with institutional procedures and that all review comments were carefully considered. Responsibility for the final content of this report rests entirely with the authoring committee and the institution.

Contents

1

Overview of Workshop Discussions

This report summarizes a workshop on software certification and dependability held April 19-20, 2004, in Washington, D.C., under the auspices of the Committee on Certifiably Dependable Software Systems. Several items should be kept in mind when reading this report:

- The workshop focused on the subset of areas that the committee believed would best help frame the program of work for the remaining study period. There are areas of direct relevance for the study that are missing from the workshop agenda, either because of time constraints or because the panelists chose to address different areas. The committee plans to gather input on those areas in subsequent activities; feedback and additional input from readers of this report are welcome.
- During the workshop, committee members deliberately refrained from questioning views expressed at the workshop—they preferred to use the time to gather input from workshop participants in an impartial manner. In addition, the committee chose not to extend the discussions in this first-phase report, instead reserving that task for the final report. Consequently, this report does not provide a free-standing overview of the current state of software development, of certification, or of anything other than the views expressed at this particular workshop.
- The panel summaries have not been edited to make the terminology used by each panel consistent across the entire report. Meanings should be clear from the context. Deciding on appropriate and consistent terminology is a task for the committee as it prepares its final report.

Listed below are the main themes arising from each panel session. These themes are not conclusions or findings of the committee; they incorporate ideas extracted from each panel that seem to represent the major thrusts of each discussion. Each panel session discussion is elaborated in Chapter 2.

PANEL A
THE STRENGTHS AND LIMITATIONS OF PROCESS

- While following particular processes cannot alone guarantee certifiably dependable software, comprehensive engineering processes are nevertheless important to achieving this goal.

- In evaluating a system, appropriate metrics should be used; it may be necessary to measure secondary artifacts (e.g., process quality) as surrogates if what is actually of interest cannot be measured.
- Developing ways to determine how best to allocate resources (e.g., understanding where errors are likely to cluster) can improve both dependability and cost-effectiveness.

PANEL B
LOOKING FORWARD: NEW CHALLENGES, NEW OPPORTUNITIES

- Over the past several decades society has become increasingly dependent on software. While desktop systems are not generally regarded as safety-critical, these days they are often task-critical for their users. This is true not only at the individual level but also at the organizational level.
- One increasingly sophisticated set of tools that can help in the software development process with respect to dependability is the set of tools related to programming languages, such as type checkers, static analyzers, and model checkers.
- Systems integration is a growing and challenging problem. Additional tools and strategies are needed to cope with large-scale systems integration issues.

PANEL C
CERTIFICATION AND REGULATION: EXPERIENCE TO DATE

- The process of certification may add value in a collateral fashion because attention must be paid to issues that might not receive it otherwise; given that software and its uses and contexts change over time, any value that certification has decays over time as well.
- Market forces and the cost structure of the software industry may create incentives to release flawed software.
- Validation—determining what the software should do—is often harder than verification, or determining whether the software does it correctly, and may be more important. Despite the difficulties of achieving validation systematically, however, many critical systems seem to function well.

PANEL D
ORGANIZATIONAL CONTEXT, INCENTIVES, SAFETY CULTURE, AND MANAGEMENT

- Systems are certified only within a particular context and up to specified system boundaries; certification of a system may not guarantee the dependability and usefulness of a system over its lifetime.
- As a system's reliability increases or is demonstrated over long periods of time, dependence on that system may increase to an extent not anticipated in the original design.
- Accountability, reporting, and communication are difficult issues that must be planned and managed in detail across an organization.

PANEL E
COST-EFFECTIVENESS OF SOFTWARE ENGINEERING TECHNIQUES

- There are interesting substantive overlaps in approaches to software development that seem philosophically opposed on the surface. In particular, agile methods such as "Extreme Programming" seem to share important elements with methods that employ formal notations for early modeling and analysis.
- Understanding what is meant by "dependability" is critical; it was observed that, given its ubiquitous use and deployment, software is treated as though generally dependable.
- Achieving dependable, certifiable software will require emphasis on process, people, and tools.

PANEL F
CASE STUDY: ELECTRONIC VOTING

- Structural flaws in the voting system go beyond the absence of voter-verifiable paper trails.
- The lack of detailed risk analysis, coupled with a lack of openness in the voting system certification process, poses serious challenges to achieving a dependable voting infrastructure.
- The current certification process does not seem to have resulted in secure or dependable electronic voting systems.

2

Summary of Panel Sessions and Presentations

PANEL A: THE STRENGTHS AND LIMITATIONS OF PROCESS

Panelists: Isaac Levendel, Gary McGraw, and Peter Neumann
Moderator: Martyn Thomas

Software development involves many activities—for example, requirements elucidation, requirements analysis, specification, version management, architectural design, modularization, component design, programming, testing, component integration, verification, performance analysis, prototyping, error correction, feature enhancement, reviews, walkthroughs, and typechecking. These activities may be undertaken in many different sequences; collectively they are referred to as the software development process. Most development standards for dependable software (such as DO-178B[1] or IEC 61508[2]) recommend that particular processes be adopted; typically, the recommended processes vary for different levels of criticality (different target probabilities and consequences of failure).

In the Panel A session, participants from industry and academia explored the strengths and limitations of using process recommendations and requirements to achieve certifiably dependable software. Some of the themes that emerged during this discussion include these:

- While following particular processes cannot alone guarantee certifiably dependable software, comprehensive engineering processes are nevertheless important to achieving this goal.
- In evaluating a system, appropriate metrics should be used; it may be necessary to measure secondary artifacts (e.g., process quality) as surrogates if what is actually of interest cannot be measured.
- Developing ways to determine how best to allocate resources (e.g., understanding where errors are likely to cluster) can improve both dependability and cost-effectiveness.

[1] DO-178B, Software Considerations in Airborne Systems and Equipment Certification. Issued December 1, 1992, by RTCA, Inc. Available at <http://www.rtca.org/>.

[2] IEC 61508, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. Issued December 1998 by the International Electrotechnical Commission (IEC). Available at <http://www.iec.ch/zone/fsafety/fsafety_entry.htm>.

The panelists and participants in the workshop largely agreed that mature, comprehensive engineering processes reduce the risk that a software development project will overrun or deliver poor-quality software. But it was also noted that a good process does not guarantee that the resulting software will be of high quality. "Process properties are not predictive of product properties," was a typical opinion; another was, "You should not believe that if you do it right it will come out right." The idea that it is possible to "get dirty water out of clean pipes" summarized much of this discussion. Process was described as a "skeleton" on which a dependable system is built.

Process quality is important even if it does not directly predict product quality. For example, it is important to be able to show that the version of the software that has been tested and certified is the exact version that is running, unmodified, in the operational system. Similarly, if the testing process cannot be trusted, it may be difficult to establish confidence in the test results.[3]

Software development processes were described as a "chain of weak links" and as "weakness in depth," in that each process step can introduce flaws that damage the product. Product certification requires adequate evidence of the properties of the product, and the weight that can be given to this evidence depends, in turn, on the quality of the processes that created the evidence. So the quality of these processes is an important factor in certification. It is important to distinguish between evidence that the process has been followed (which gives credibility to the outcome of the process) and evidence of the system properties that the process produces.

A detailed record of a systematic process can be essential for the many development tasks that depend on information not easily obtainable from the program itself. For example, reliable build logs and version control are necessary for tracing the affected versions of systems in the field once a fault has been discovered. Similarly, avoiding complete recertification of a system after a maintenance change requires sufficient evidence of the maximum possible impact of the change, so that only the affected areas need to be recertified.

Panelists discussed the artifacts produced as part of development (program code, specifications, designs, analyses, and the like). One panelist noted that it is the properties of these artifacts that should be measured: "I run your code, not your process." For this reason, a key activity in certification should be measurements of the product and of the intermediate artifacts. Such measurements could include system performance, test coverage, the consistency and completeness of specifications, and/or verification that a design implements a specification. While there does not seem to be a single metric that can predict dependability, several participants said that measures such as these, when used in combination, are good predictors of dependability. It is important to measure properties that actually matter directly, and economic theory suggests that measurement skews incentives: "If you want A and measure B, you will get B." All of this suggests the importance of good empirical research that relates the attributes under consideration and makes it possible to discern what the dependent and independent variables are.

There was some discussion of the phenomenon that software errors are not evenly distributed throughout a system—they tend to cluster in the more complex areas, creating black holes of software defects. These black holes can be located by looking at the past and recent history of a current release. However, there is an underlying assumption that there are resource constraints and never enough resources to analyze an entire system in this manner. Furthermore, some forms of analysis or testing may be impossible, requiring orders of magnitude more resources than could possibly be made available. One must focus the analysis on areas one thinks deserve the most attention. These black holes in the resulting software can often be traced to black holes in the design or specification, so deep analysis of the quality of these artifacts, early in the development of software, can be very cost-

[3] It was suggested that while somewhat too prescriptive, the Capability Maturity Model is correct in its assessment of which processes matter as well as in its insight that there is a necessary progression in process improvement (it is not possible to leap easily to Level 4 from Level 1). The committee will explore this and other process-oriented models in the second phase of the study.

effective. Panelists and participants generally agreed that limitations on staffing and time are a major consideration in developing dependable software.

Several participants commented that most problems in delivered systems turn out to have their root cause in specification errors, and one said (provocatively and perhaps only half-seriously) that “programming is a solved problem.” It was observed that security problems in contrast often flow from implementation errors such as buffer overflows. “No system is 100 percent secure. Proactive security is about doing it right. Security problems come from software flaws—we must do it right.”

Doing it right was suggested to involve the following:

- Producing the right artifacts and using the knowledge of computer scientists and software engineers to guide how those artifacts are produced so that they have the right properties and can be measured or analyzed, and
- Measuring the artifacts to confirm those properties.

Panelists and participants also largely agreed that the world is imperfect and not all the software properties of interest can be measured directly. In these cases, secondary evidence must be used—perhaps including measurements of process quality and staff competence—although it is important not to lose sight of the limitations of such evidence.

PANEL B: LOOKING FORWARD: NEW CHALLENGES, NEW OPPORTUNITIES

Panelists: Robert Harper, Shriram Krishnamurthi, James Larus, and André van Tilborg
Moderators: John Rushby and Lui Sha

Panel B's topic for discussion was what has changed in the past decades with respect to software certification and dependability. Several themes emerged during the discussion:

- Over the past several decades society has become increasingly dependent upon software. While desktop systems are not generally regarded as safety-critical, these days they are often task-critical for their users. This is true not only at the individual level but also at the organizational level.
- One increasingly sophisticated set of tools that can help in the software development process with respect to dependability is the set of tools related to programming languages, such as type checkers, static analyzers, and model checkers.
- Systems integration is a growing and challenging problem. Additional tools and strategies are needed to cope with large-scale systems integration issues.

While society is increasingly dependent upon software, desktop systems are not generally regarded as safety-critical. However, with respect to what has changed about the use of software systems, one panelist pointed out examples of errors in desktop spreadsheet programs and Web-based travel reservation systems. The existence of such errors illustrates that for the people using such software, even seemingly mundane applications can be critically important for achieving a larger goal. Systems are increasingly interconnected and interdependent, which means that local faults attributable to poor design can lead to wider failures. The argument proposed was that systems are not task-critical because someone in authority says they are; rather, a system becomes task-critical when someone performing a critical task uses that system.[4] Dependence on systems happens not only at an individual level, as in these examples, but also at an organizational level. For example, the majority of medical-related computing systems (including patient records and lists of appropriate dosages for drugs) are run on standard consumer platforms. When these systems fail, hospital operations can be placed in jeopardy. Thus, some argued that software dependability has become an increasingly important quality-of-life issue and should not be limited to safety or security applications such as aviation and nuclear power plants.

Some panelists and participants argued that programming languages and their associated tools are the key to dependability on the grounds that "code is the ultimate reality." It was claimed that advances in this field are both broad and deep and that fundamental research on types and programming logics, theorem proving, model checking, and so on has advanced the state of the art. Process-oriented methods have limitations—for example, they do not tend to scale well in a world with thousands of disparate components, and they tend to rely heavily on continuity and corporate knowledge. Programming-language-based approaches may scale better, but only for establishing rather local and low-level properties. The challenge is to find a way to combine the best techniques that can be used at the system level with the best lower-level techniques, and to generate all the evidence required for certification as an integral part of these development processes.[5]

[4] At the same time, it should be noted that there is disagreement about what should be considered task-critical. As the use and scope of software expand, the number of critical components expands. However, some argue that not everything should be labeled "critical" and that the degree of criticality can vary greatly. The committee will continue to explore this and related issues in the next phase of the project.

[5] This is not to suggest that appropriate languages and tool sets will guarantee dependability, but rather that incompletely defined languages can pose problems that even the most skilled programmers have difficulties

Others argued that programming is basically a solved problem and that all the significant issues arise at the system level. It was suggested that language choice matters much less than system-level considerations. However, there seemed to be general agreement that the dependability problem is larger than this seeming dichotomy would suggest and cannot be solved by any one technique, be it architecture, static analysis, process, language, or human factors. Software poses unique challenges, but, as was noted, also unique opportunities. It is not possible to look at and work with the last 10 versions of a particular bridge, for example, in the way software systems can be examined. A rough consensus seemed to be that while the dependability problem cannot be solved by any single method alone, dependability can be seriously compromised by any single weak link. Nevertheless, while there are no silver bullets, there are many specific promising interventions that, taken together, could make a very big difference.

Panelists pointed to great progress in the development of formal methods and their application in industry. Specification languages, logics, and verification methods have matured and are increasingly applicable in the context of large and complex systems (although usually only in rather limited ways). Type systems represent the most elementary use of formal methods, but they are used with success widely (e.g., in languages such as Java and C#); applied type theory was cited as one of the greatest successes in formal methods. Others pointed out that many system failures are environment- and input-dependent. Current type systems do not address the role of the environment in which the system operates, since they can only express local properties of code modules. Stronger forms of static analysis have great promise, although some participants expressed skepticism about the effectiveness of many such forms that are currently in vogue, due to their incomplete and unsound nature.

It was noted that there has been great progress in the construction of stand-alone systems. For example, the desktop system has become more powerful, has more useful features, and has become much more reliable than it was 10 years ago. Unfortunately, the same cannot be said about large systems of systems. Systems integration has remained a vexing challenge. The development of large systems of systems often has the unfortunate characteristic that early phases experience relatively smooth subsystem development while later phases encounter serious integration problems. For example, one participant remarked that during systems integration and testing, complex avionics tend to fail repeatedly when the pilot attempts to use the radar, communication, navigation, identification, and electronic warfare systems concurrently. These integration problems are very costly to fix and are often traced back to complex and unanticipated interactions between subsystems. While there are usually voluminous documents of system interface definitions, they often deal only with normal and predictable functional interfaces. There are many integration problems caused by unanticipated interactions between different technologies developed in isolation, especially in aspects related to real-time properties, fault tolerance, security, and concurrency control—namely, those properties that cannot be established locally. How to control and manage interactive complexity is a key challenge for systems integration.

Before one can start solving the problem of interactive complexity, it will be important to have good instrumentation that allows one to observe the details of the system state and to trace system behavior. Failures that arise in the integration of systems are often highly sensitive to workload and timing, which makes them extremely difficult to track down. There are very few working approaches available that succeed at properly instrumenting a system in spite of the important role it plays.

There was rough agreement that dependable system development should be risk-driven. It is important to determine which of the various aspects of dependability must be determined at the outset

avoiding. Modern languages with associated deep analysis tools can guarantee the absence of whole classes of defects and greatly assist the programmer in achieving dependable systems cost-effectively.

and which can be deferred. These critical aspects are project-dependent, and the designers' and developers' responsibility is to determine the key needs of the project. How to proceed, what aspects to certify, and to what levels of dependability are unclear. Improved notations and modeling tools are needed to address dependability concerns in all phases and in all aspects of development, but it is important that all software artifacts—whether specifications, requirements, models, or code—be kept in sync with each other. How to achieve this remains a serious challenge.

PANEL C: CERTIFICATION AND REGULATION: EXPERIENCE TO DATE

Panelists: Brent Goldfarb, Mats Heimdal, Charles Howell, and Robert Noel
Moderators: Michael DeWalt and Scott Wallsten

In the Panel C session participants discussed what is meant by "certification" and "regulation," how much has been done, and how much is possible. Panelists and audience members also discussed whether a well-functioning marketplace for software and software development services would be sufficient to achieve dependability. The panelists noted that these are increasingly important questions as software plays an ever larger role in our lives. Several themes became apparent during this discussion:

- The process of certification may add value in a collateral fashion because attention must be paid to issues that might not receive it otherwise; given that software and its uses and contexts change over time, any value that certification has decays over time as well.
- Market forces and the cost structure of the software industry may create incentives to release flawed software.
- Validation—determining what the software should do—is often harder than verification, or determining whether the software does it correctly, and may be more important. Despite the difficulties of achieving validation systematically, however, many critical systems seem to function well.

The panelists started by focusing on two heavily government-regulated industries: avionics and medical software. They noted that, in general, the software in these industries seems to work fairly well. The Federal Aviation Administration (FAA) works closely with the avionics industry to ensure that regulations are adequate but not suffocating. The FAA also appoints people within these companies to provide software approvals on behalf of the government. This helps the process work more smoothly. This approach is also being discussed by the medical community for the medical device industry, but only in a preliminary way.

One panelist stressed the traditional distinction between verification and validation (verification is a check that software satisfies set requirements, and validation is a check that specifications match the actual need). The general consensus of the panel, and the focus of the ensuing discussion, was that validation is both harder than verification and much less well addressed by current processes. That is, deciding what the specific goals are is the most important step. Once the goals are clearly defined, it becomes easier (though by no means necessarily easy) to verify that those goals are met.[6] Some claimed, in addition, that validation matters more than verification for critical systems, because few of the errors that have been critical in practice indicate failures of verification.

The value of certification itself was debated extensively by the panel and audience. There was, however, general consensus on three points about certification.

First, even if certification is not directly effective, it may add value in the sense that it forces developers to pay closer attention than they might otherwise. Participants noted that despite the difficulties in achieving validation, many safety-critical systems seem to function well, especially in the arenas of avionics and (to a lesser extent) medicine. One panelist argued that processes such as those recommended in DO-178B have collateral impact, because even though they fail to address

[6] It should be noted that there are many kinds of functional and so-called nonfunctional requirements, and that the relative importance of verification and validation varies among these (and also among the various stakeholders in the requirements).

many important aspects of a critical development, they force attention to detail and self-reflection on the part of engineers, which results in the discovery and elimination of flaws beyond the purview of the process itself.

Increasing automation and over-reliance on tools may actually jeopardize such collateral impact. As people are removed from the testing process, for example, in favor of automated procedures that specifically test a set number of items, they are less likely to notice gross omissions in testing or invalid environmental assumptions.

Second, any value that certification has decays over time. Certification, in some sense, is a legal status and is considered a Boolean value—either the software is certified or it is not. Worse, the certification applies to a particular system at a particular time. In reality, things are more complicated. For any change made to a system, it is important to know how far the system has moved away from the version that was certified. One panelist argued that at one extreme, the value of certification is zero as soon as the product is delivered. Even if that assertion exaggerates the problem, there was general agreement that the value of certification does decay rapidly because of changes in the way a product is used, the way people interact with the product, and ways (many unexpected) in which the product must interact with other systems (see the summary of the Panel D session for more on this point). Many system failure postmortems begin with the statement, "A small software change resulted in . . . ."

Third and finally, a potential benefit of certification is that it establishes a baseline from which changes in performance/dependability/reliability can be measured later. A related question about certification is what, exactly, should be certified. Participants debated whether people should be certified, rather than or in addition to software, but there was no agreement on this question.

There was a claim that markets are not efficient for safety-critical software, especially when used by the military, though this point was disputed. The economics of the software industry, including incentives for the firms involved and how those affect reliability, were discussed. A major problem is the lack of information flow between groups, a problem that can prevent markets from functioning properly. Reference was made to George Akerlof's seminal paper on the used-car market, where asymmetric information flows prevented that market from working efficiently.[7] One person noted that for used cars, the Internet and, in particular, services that permit the prospective buyer to purchase a vehicle history report have helped solve that problem. Thus sometimes innovation and markets can help resolve the problem of asymmetric information. It was also noted that there is not a one-size-fits-all approach—market forces work well with some critical systems but not as well with others. Market forces might not work well with some military systems, for example.

It was suggested that the cost structure of the software industry may create a perverse incentive to release software with errors. In particular, software is a high-fixed-cost, low-marginal-cost industry where the cost of releasing corrective patches is low and becoming lower in most cases (although not for embedded control systems). This can create an incentive to release a product before it is ready in order to capture market share, knowing that a patch can be released later at almost zero cost. Although the practice of patching may apply more to the market for desktop consumer applications than for critical systems, this and other aspects of the cost structure of the industry merit serious consideration.

[7] Akerlof, George A. 1970. "The Market for 'Lemons': Quality Uncertainty and the Market Mechanism." *Quarterly Journal of Economics* 84(3), 488-500.

PANEL D: ORGANIZATIONAL CONTEXT, INCENTIVES, SAFETY CULTURE, AND MANAGEMENT

Panelists: Richard Cook, Gene Rochlin, and William Scherlis
Moderators: Charles Perrow and David Woods

Panel D explored implications of certification and dependability requirements within an organizational context. Some of the themes that emerged during the discussion were the following:

- Systems are certified only within a particular context and up to specified system boundaries; certification of a system may not guarantee the dependability and usefulness of a system over its lifetime.
- As a system's reliability increases or is demonstrated over long periods of time, dependence on that system may increase to an extent not anticipated in the original design.
- Accountability, reporting, and communication are difficult issues that must be planned and managed in detail across an organization.

The panelists stressed the idea that system design must consider the end users of a system and how that system will be used in context. Certification likewise must address not only the process of creating dependable software, nor even just the final product, but also the many ways it might be used. The participants noted a tension between the narrower and crisper notion of certification, and the broader and blurrier notion of dependable usefulness. Although some certification processes take into account the ultimate context of use, many emphasize primarily the internal consistency, coherence, and completeness of the software itself. Dependable usefulness, in contrast, entails the extrinsic characteristics of the computer systems within the domain of interest, and different domains cope with certification demands differently.[8]

During the discussion, workshop participants explored the sorts of difficulties related to certification and context that the committee should consider during the rest of the project. In a sense, some suggested, complex software might just as well be acknowledged as a natural phenomenon: software-intensive systems have become too complex to be completely understood before the fact, and they can behave in surprising ways. Thus the tendency is to work back from the behavior of systems to mechanisms in order to understand the workings of these systems.

Although overt software failures do occur, it was observed that in many instances dependability failures are not the type that could have been mitigated by certification—many failing systems have internal consistency of the sort that certification tests. Instead, failures result from a mismatch between the device requirements defined by the developers and the requirements of the domain in which the devices are used. Indeed, some now believe that virtually all instances of "user error" with computer-based medical devices are examples of this mismatch. These mismatches would not easily be detected by certification processes, such as DO-178B, that pay only minimal attention to analysis of requirements in context. Instead, the risk is that the certification process itself could obscure the fact that the device operating as designed actually lends itself to uses for which it will prove undependable within the larger context.

In general, it was argued that what often first appear to be user or operator errors turn out to be design errors resulting from a lack of recognition of the conditions under which the product is or will be used. A complicating factor is that the introduction of a new system typically changes the

[8] Many consider certification in avionics (e.g., DO-178B requirements) to be extremely good and wish such stringency could be applied to their own domains. Others within avionics acknowledge that the certification methods there are not perfect and do not solve every problem.

user's context as new uses are found, and thus new paths to failure must be anticipated and guarded against. As systems become more useful, they may no longer be backups or add-ons to older systems but instead become the primary mechanisms that users rely on. Moreover, systems and the threats against those systems coevolve. All these observations highlight the importance of clearly articulating the properties of the environment in which the system will be deployed and, in particular, the properties of the environment that the system users depend on.

Reliable software is desirable, of course, but operators and users must recognize that it may be counterproductive to depend too heavily on the reliability of the software. Instead, users should retain skills that might appear to have become superfluous so that they can learn to recognize unreliable behavior and cope with it appropriately. Unfortunately, as systems become more reliable, users grow to depend on them uncritically. In other words, depending on success can be dangerous.[9]

There was some discussion of the role of accountability. Some argued that the lack of accountability was a problem in the development and deployment of dependable software, and that blame should be assigned with accompanying sanctions to those whose actions resulted in failures. Others noted that locating a single point of failure or finding individuals to blame can be difficult. Moreover, assigning blame may actually make matters worse by retarding the flow of information. One might therefore reward information flow instead, even though it may expose weak points and blemishes. It was observed that positive reinforcements are more effective in changing behavior than negative reinforcements, and that decades of human performance literature and research demonstrate this fact. It was also noted that care must be taken when the introduction of a new system is likely to increase workload burdens within an organization. In fact, any new system will impose some workload costs, and if those who bear that cost do not receive direct benefits or increased capabilities from the change, the introduction is likely to founder.[10]

The licensing of staff who have key roles in critical projects was discussed and compared with the licensing of professionals in other disciplines. There are, of course, advantages and disadvantages to licensing. However, some participants noted that licensing could be considered one possible way of imposing standards of competence, but urged that such licenses should have to be renewed relatively frequently.

There was some disagreement over the role of regulation and whether further regulation would help or hinder. Most participants believed that there was a positive role for regulation under the proper conditions, and that, indeed, much more regulation of software was required. Who should do the regulating—industrial trade organizations, professional societies, or the government—was not discussed.

Field testing, while desirable, may be prohibitively expensive. Simulation is therefore an attractive alternative and might be acceptable for some kinds of certification. The aviation industry was singled out for having exploited simulation successfully to increase dependability. Simulations

[9] An example was given of an accounting system that performed quite well over a period of time. Eventually, as individuals with experience before the system was adopted were replaced, the level of understanding of the underlying substantive accounting issues diminished. Users understood the operation of the system, but did not understand what the system was doing deeply enough to manage well when a failure occurred. Another example along these lines was of the difference in caution between a user of a somewhat unreliable operating system who is therefore careful to save documents frequently and a user of a generally reliable operating system who is not so careful. In the latter case, infrequent failure can be more damaging than frequent failure.

[10] Norman, D.A. 1988. *The Psychology of Everyday Things*. New York, NY: Basic Books.; R.I. Cook and D.D. Woods. 1996. "Adapting to New Technology in the Operating Room." *Human Factors* 38(4), 593-613.

in other software-rich environments might offer something similar to this kind of testing. Such simulation capabilities could be explored to help in the definition of requirements as well as for validation. However, it was also noted that good simulation imposes additional costs and can be very expensive for many kinds of applications, which relates to issues raised in Panel C.

PANEL E: COST-EFFECTIVENESS OF SOFTWARE ENGINEERING TECHNIQUES

Panelists: Kent Beck, Matthias Felleisen, and Anthony Hall
Moderators: Peter Lee and Jon Pincus

The purpose of the Panel E session was to provide some basis for understanding the cost-effectiveness of software engineering techniques as they relate to dependability and certification. The general question to the panel was whether there is evidence for the cost-effectiveness of various software engineering techniques, either today or looking toward the future. The panelists cited metrics such as frequency of warranty claims and delivered defect density. One panelist argued that the creation of a market for dependable software components would lead to stronger evidence for cost-effectiveness. All three panelists described promising directions that may lead to more cost-effective software engineering techniques for creating dependable software. Overall, several key themes emerged from this panel, among them:

- There are interesting substantive overlaps in approaches to software development that seem philosophically opposed on the surface. In particular, agile methods such as "Extreme Programming" seem to share important elements with methods that employ formal notations for early modeling and analysis.
- Understanding what is meant by "dependability" is critical; it was observed that, given its ubiquitous use and deployment, software is treated as though generally dependable.
- Achieving dependable, certifiable software will require emphasis on process, people, and tools.

Perhaps most surprising to the majority of participants was the strong overlap in two software development approaches described by the panelists: "Extreme Programming" (XP) and "Correctness by Construction" (CbC). While on the surface these two approaches seem philosophically opposed and differ greatly in terms of their provenance, discussion revealed intriguing similarities. Specifically, proponents of each emphasized the importance of taking a risk-based approach, in which the highest-risk (often the least-well-understood) areas of a project are tackled first.

When asked how their approaches would respond to the issue of integration discussed in Panel C, proponents of each approach noted that if integration were clearly the highest-risk issue in a project, the appropriate response would be to address it from the start, and to integrate the subsystems "early and often," in the hopes of mitigating this risk. Even though much of the functionality would not be available in the earliest integration phases, architectural-level mismatches would be discovered much earlier as a result. It was also noted that having an overall architecture in place, along with an empowered central architecture team, seemed important to address this problem.

Both XP and Correctness by Construction stress the importance of clear requirements. CbC follows a detailed requirements elucidation method (REVEAL[11]), whereas XP's "test-first" approach argues for the use of failed test cases as a record of the evolving requirements. Both approaches can be viewed as attempts to make requirements explicit as early as possible (although REVEAL, unlike XP, crucially addresses properties that are not observable at the system interface and is less susceptible to the risk of focusing on low-level issues). The XP community has apparently devised new techniques for improving the observability of nonfunctional properties by making them testable

[11] For more about REVEAL, see "Will It Work?," Jonathan Hammond, Rosamund Rawlings, and Anthony Hall, in *Proceedings of RE'01*, 5th IEEE International Symposium on Requirements Engineering, August 2001, available at <http://www.praxis-cs.co.uk/pdfs/Will_it_work.pdf> and <http://www.praxis-cs.co.uk/reveal/index.htm>.

(e.g., by defining explicit trials to establish usability), although some properties—such as the absence of race conditions or buffer overflows—are not easily tested at all.

A number of questions regarding safety-critical (and in particular real-time) software were raised with respect to XP. It was claimed that while XP has not been applied to safety-critical software, the best XP teams record defect rates much lower than those typically found in commercial software, along with substantially higher productivity. It was also noted that while XP had originally been applied in small (8-10 person) teams, it had now scaled to an organization of approximately 250 engineers and thus that the principles could apply (perhaps with some modifications to the practices) to larger, cross-organization projects.

It was reported that a series of projects using the CbC approach have extremely low defect rates: from a 1992 project with a delivered defect rate of 0.75 defects per thousand lines of code (KLOC) to a 2002 project (100,000 lines of code (LOC), with a productivity of 28 LOC/person-day) with a delivered defect rate of 0.04 defects/KLOC. Improvements over that time frame were attributed to the introduction of the REVEAL requirements analysis method, adoption of more powerful static analyses (in particular the use of the Spark Ada subset and its Examiner tool), and a better engineering process, with earlier integration and more frequent builds.

It was pointed out that software is indeed dependable, inasmuch as all the attendees had depended on software to arrive at the workshop. However, given that far too many software development projects fail, or have overruns in cost and time, the reliability of the software production process was decried.

A proposal to reinstate the goal of building a “software market” of interchangeable software components[12] was based on the following rationales:

- Market mechanisms provide concrete measures of utility and value; such measures are currently missing in software engineering, and thus comparisons are difficult to make.
- Improvements in process, education, and technology often come as a result of the competition afforded by markets.
- A market of interchangeable components might provide a reliable means of choosing specific software products for specific tasks.

The importance of process, people, and tools (in various orders of importance) was emphasized, not surprisingly. Also discussed at some length was the importance of continuing education for programmers, by analogy to health care workers (specifically, surgeons). A substantial amount of discussion related to the role of language, and not only at the level of programming. All the panelists agreed that the choice of language or notation is important, as it affects the thought processes of the people involved in the project. There was also a rough consensus that at least local properties should be kept “with the code” via strong interface specifications. Different languages might be appropriate for different purposes, again casting the choices in terms of risks.

A number of other topics were also raised and discussed briefly. Panelists emphasized the importance of approaching software from an engineering perspective. Just as chemical engineers and mechanical engineers have process differences due to the different materials they work with, software engineers with their very different “materials” likely have different processes as well.

[12] It was observed that this notion of a market of components has often been discussed but that there is little historical support for such an idea and considerable skepticism regarding its present feasibility.

PANEL F: CASE STUDY: ELECTRONIC VOTING

Presenters: David Dill, Douglas Jones, Avi Rubin, and Ted Selker
Moderators: Reed Gardner and Daniel Jackson

Panel F addressed the controversial dilemmas posed by current electronic voting systems.[13] Most of the discussion was focused on the flaws of these systems and the process by which they are produced, although it was noted that computing-related issues may be overemphasized, distracting attention from more serious flaws in the wider electoral system. The major themes of discussion were as follows:

- Structural flaws in the voting system go beyond the absence of voter-verifiable paper trails.
- The lack of detailed risk analysis, coupled with a lack of openness in the voting system certification process, poses serious challenges to achieving a dependable voting infrastructure.
- The current certification process does not seem to have resulted in secure or dependable electronic voting systems.

While much attention has been focused on whether the use of paper (either in the form of paper ballots or a voter-verifiable paper trail) is an effective corrective for problems with electronic voting systems, it was noted that voter verifiability is not the only challenge today with the voting infrastructure, and perhaps not even the most important problem. Votes go “adrift” everywhere. In the 2000 U.S. presidential election, for example, 2 percent of votes were lost because of problems in registration databases. The panelists noted that although attention has been focused mostly on the voting machines themselves, voting is a complex system in a complex social setting, and any weak link can compromise the entire process. While there are problems with the current paper voting system, and the electronic voting system has the potential to prevent some of these errors, it also has the potential to introduce new errors. The panelists focused on errors related to software flaws, inadequate security, distributed database problems, communications inadequacies, limited auditing capabilities, and poor user interfaces.

There has apparently been no serious attempt to perform a detailed risk analysis of electronic voting systems—a standard procedure in the development of safety-critical systems. In fact, it is not clear that those involved in electronic voting systems—from vendors to election officials and certifiers—even appreciate these systems’ critical nature, perhaps because the purely algorithmic aspect of voting (count the votes and the candidate or initiative with the most votes wins) is so simple. Most of the panelists and other workshop participants were confident that such a risk analysis could be conducted, and that it might lead to a simplification of the voting process and the identification of a small trusted computing base within the larger system that would correctly be the focus of attention.

The lack of openness in the voting system certification process is a serious problem. The certification of electronic voting systems is currently conducted by companies that report to local election officials in states and counties. Each jurisdiction makes its own decisions about which companies to use and interprets the results in its own way. Usually, the choice of certifier, the inputs to the certification process, the certification report, and even the criteria used are not made known to the public. A participant reported that, in some cases, certifiers have not been willing to talk to outside experts. As a result, confidence in the system has eroded and insights that outsiders might bring have not been taken advantage of. Given these concerns, how can the certification process itself come to be trusted? (This is indeed a problem that certification will face in any domain.)

[13] For the purposes of this discussion “electronic voting” is distinct from Internet voting.

Even for a given vendor's system, performance varies widely because of differences in ballot design and user interface. Even small differences in a user interface or ballot design can have dramatic effects on the dependability of the system as a whole. In the United States, ballots tend to be very complicated; one panelist noted that a typical California voter makes more choices in a single election than a typical British voter does in a lifetime. Ballots differ widely across counties and may present dozens of choices. As a consequence, the layout of the ballot is complex and highly variable. The user interface of an electronic voting machine may give some propositions or candidates an advantage—either by design or by accident. The issues here have not been well studied.

An end-to-end check, which would ascertain that each person's vote was correctly acquired, recorded, transmitted, and counted, might protect against software flaws and malicious attacks. But designing such an audit within the constraints of ballot secrecy requirements is a difficult problem. The idea of using DREs (Direct Recording Electronic systems) simply as intelligent printers for generating completed ballots that might then be optically scanned—a voter-verifiable paper trail—was suggested, although one panelist argued that the attention paid to this issue has diverted resources from more important issues. It was also noted that it may be infeasible to expect voters to check a paper audit consisting of dozens of choices, and that audio feedback, for example, might be more useful to ensure that entry errors are not made.

A further risk is the lack of a process to ensure that the software used in an election is the software that was certified. Electronic voting systems, highly dependent on software, have the advantage of flexibility; unfortunately, the same flexibility allows late changes to the software that have not been certified, and participants noted that there have been reported cases in which voting machine software has been modified at the last minute, long after certification has been completed. Lack of expertise with computers and software among local election officials exacerbates this problem: officials might be able to detect tampering with a traditional paper system, but they are not necessarily qualified to understand the issues involved in the loading and updating of software.

Lack of observability is also a serious problem. Critical systems sometimes suffer because their failures may not be immediately observable, but in the worst cases catastrophic failures are rarely invisible: a medical device that kills a patient, for example, will likely eventually be discovered. But a voting system could fail repeatedly, corrupting the results of large elections, without discovery. The secrecy requirement of voting is one of the root causes of this problem. Mechanisms for detecting electronic voting system failures must be developed and tested, and perhaps local policies and processes will need to be changed to allow such failure detection and prevention.

Panelists agreed that the certification process for electronic voting systems has failed demonstrably. Despite the procedures set up by local election officials and the work of third-party certifiers, voting systems have been found by outside experts to contain egregious flaws and to be susceptible to easy attacks. Some of the reasons for this failure include:

- *The closed nature of the process and the skewing of incentives*. Local election officials are often involved in the procurement of systems and have little incentive to report problems with such procurements. Political incumbents are often involved in the choice of vendors and certifiers, compromising the independence of the selection.
- *The fact that purchasers are typically ignorant about complex computer systems, and lack the ability to assess risks and evaluate certification efforts*.
- *A mismatch in expectations*. While certifiers may recognize how incomplete and limited their analyses are, their reports are often misinterpreted as implying that the system is unassailable.

In addition, there are core technical problems with current electronic voting systems, among them the inherent complexity of the software, the nature of collaborations between vendors and purchasers, and the infeasibility of meeting certain requirements. The software problem is a complex one, and it is not clear that a specific and evidently correct solution is possible. The end users—voters and local election officials—are not likely to be able to understand the code themselves or to appreciate the implications of its complexity for errors. Inappropriate collaboration between local politicians and the vendor, along with the ability to control the setup of the ballot, may be problematic not only because of potential malfeasance but also because neither the vendor nor the politicians are likely to be experts in designing usable ballots. Requirement specification may also be problematic if the requirements mandated in legislation are technically infeasible or unspecific.

It was observed that electronic voting systems pose a fundamentally harder challenge than many other safety-oriented critical systems because of the high risk of motivated, malicious attack. Because voting is so foundational to our democracy and because there are strong incentives for rogue states, terrorists, political parties, special-interest groups, and even individuals to influence election results, the threat of attack on such systems is highly likely. One panelist estimated that in the current system bribing only a handful of people could allow serious compromise. At the moment, large-scale attacks on medical and avionics system software are relatively minimal; there seems to be little motivation for such attacks because these types of systems tend to be very distributed and physically inaccessible. Accordingly, certification in these domains has evolved without much attention to the kinds of adversaries that voting systems might face,[14] although there is increasing concern that such systems may themselves become targets of attack.

The difficulty in identifying the user of the system creates additional challenges for building dependable and certifiable voting systems. In most system development environments, there is a user who can evaluate the delivered system for fitness of purpose, and the same party that evaluates the system has the primary vested interest in its quality. For voting systems, there is no single user. Local election officials play a key role in acquiring and managing these systems, but it is arguably the voter who is the true user. Unfortunately, the voter has little influence on the process of acquisition and certification, and cannot easily even assess how likely it is that his or her vote was recorded correctly.

[14] This situation highlights the difference between security and safety. While each is needed to a greater or lesser degree in all systems, the techniques and lessons learned in an effort to achieve one are not necessarily applicable in achieving the other.

3

Summary of Closing Session

The workshop finished with a brainstorming session, the goal of which was to identify the important questions and issues raised during the workshop and to determine those that merit further investigation by the committee. The participants reviewed the following aspects of software certification: applications, lessons learned from failures and successes, scope, techniques, process-based and product-based methods, and obstacles to effective certification.

Certification has traditionally been applied to safety-critical infrastructure that is procured or regulated by the government. Participants wondered whether it should be applied more widely. Some suggested that even consumer software was an appropriate target for certification, as it has a substantial impact on the quality of life, while others advocated certification of operating systems and development platforms. Some participants cautioned that too-strict certification requirements could have negative impacts, such as the expenditure of resources on the production of ultimately unhelpful documents. At the same time, it was noted that regulation and associated certification processes exist to defend and protect those who have no other voice in the product development process, and that it would be unfortunate if truly high quality software were confined to the domain of nuclear reactors and airplanes. The committee was encouraged to host a panel of representatives from areas where certification is generally considered effective (such as avionics) to discuss the potential for broader applicability of certification.

It was also suggested that the committee study certification failures. Failures may include certification of a device or system that failed to achieve the goals for its dependability and certification whose costs caused the project to fail. The National Security Agency's Orange Book[1] was suggested as an example of a certification effort with, at best, mixed outcomes. According to one panelist, "Systems were evaluated and blessed at C2 [a particular level of assurance determined in the Orange Book]; people thought that it meant something, but it didn't." Participants suggested that the committee attempt to get honest assessments from a few people who would be candid about what went wrong. It was also suggested that the committee investigate certification successes. The FAA, for example, has been certifying software for quite some time and may be able to offer lessons learned from what has worked and what has not.

Participants disagreed about the appropriate scope of certification. Some thought it should focus on one or two narrow attributes, while others argued that it should encompass all realistic requirements in a common context in order to address the fact that requirements frequently interfere

[1] More formally known as the *DOD Trusted Computer System Evaluation Criteria*, DOD 5200, 28-STD, December 26, 1985.

with one another. There did appear to be some consensus among panelists that certification processes should focus on the desired properties rather than on the manner in which the artifact is constructed. It was observed, however, that some properties that are not directly observable may require secondary evidence. There was substantial interest in certifying components for composability. There was also interest in certification processes that can be applied to a software component as it evolves.

There was some discussion of appropriate techniques for certification. Several participants contended that formal methods have a role to play in certification and that although there are limits to what they can achieve, "we should continue the long march of formal methods." It was suggested that scale and adoptability should be the primary aim of continued research in this area, with a focus on "small theorems about large programs." There was also controversy over the maturity of certification techniques and of software development in general. Some felt that generally reliable software can be achieved using existing software development techniques but that certification could be used to "raise the bar." Others emphasized the need for more research on the use of tools and techniques to increase software quality.

The question of process-based versus product-based certification was discussed. There was some consensus that a combination of the two is necessary, and that "process evidence is only secondary to product evidence." The new European standards ESARR 6[2] and SW01[3] were offered as examples of the combination approach. It was suggested that formal methods could offer support for "baking the evidence of process into the artifact itself," using such techniques as proof-carrying code.

Panelists warned of several obstacles to effective certification. For example, standards that mandate adherence to untestable criteria are unlikely to be effective, as are certification processes that lack accountability. Standards that provide insufficient connection to the end goal can increase the cost of creating software without increasing its actual dependability. Standards may be too technology dependent, mandating methods of constructing (or not constructing) software. They may mandate the production of enormous documents that have little if any impact on the dependability of the software, or the measurement of "trivia" that is easily measurable but offers only a false sense of security.

Finally, participants briefly touched on some additional considerations, and the session concluded with suggestions for the committee in the preparation of its final report. There was some support for the notion that software development companies should be required to publish defect rates. It was noted that software products are often licensed without the "fitness-for-purpose" requirements that apply to other goods. The education of both professionals and consumers was a recurring topic: "We should make tools that give people best practices demos, so people can tell how to write good software, and can tell the difference. Consumer pressure [to produce dependable systems] is not as powerful as people had hoped."

The committee was encouraged "not to be hidebound by current practices" and to take a longer-range view about what people involved in the development of complex software systems need to learn and know. What are the criteria for a good study in the area of certifiably dependable systems? What should the committee do to ensure that such a study provides value to the community and capitalizes on the wisdom and experience of the community at large?

It was suggested that the committee's final report should, at least in part, be positive. Participants observed that the community tends to be negative but that much has been achieved. It

[2] The Eurocontrol Safety Regulatory Requirement (ESARR 6) provides a set of guidelines to ensure that risks associated with operating any software in safety-related ground-based air traffic management systems are reduced to a tolerable level. This guideline was developed by the European Organisation for the Safety of Air Navigation.

[3] SW01 is part of a larger set of civil aviation guidelines and standards called CAP 670 established by the Civil Aviation Authority in the United Kingdom. SW01 deals specifically with the design and assessment of ground-based services, including air traffic control systems.

was suggested that clear recognition of the problems and acknowledgment that they are being worked on are equally important. Achieving certifiably dependable systems is a long-term goal, with many unknowns and the potential for significant changes in the way things are done. Changes will take a long time to implement, and so the goal should be to set out a road map. It is important to clearly distinguish between what can be done now and what is desirable but requires technology that does not yet exist.

Appendixes

A

Workshop Agenda

MONDAY, APRIL 19, 2004

Welcome
Charles Brownstein, Director, Computer Science and Telecommunications Board
Daniel Jackson, Chair, Committee on Building Certifiably Dependable Systems

Panel A: The Strengths and Limitations of Process

Isaac Levendel, Independent Consultant
Gary McGraw, Cigital
Peter Neumann, SRI International

Moderator: *Martyn Thomas*

The focus of this panel is the contribution of particular processes and process characteristics to the successful development and effective certification of dependable systems.

- What are the important characteristics of the processes that you believe should be followed when developing certifiably dependable systems? What evidence exists to support your opinion? How would it be possible to gain stronger evidence?
- How important is evidence of the development process (or the absence of such evidence) to certification that a system meets its dependability objectives? Does your answer depend on the nature of the system under consideration? If so, in what way?
- What specific processes, if carried out effectively, could provide sufficient evidence that a system meets its functional requirements? Does your answer change if the system is (a) preparing customer bills for a major utility company; (b) controlling a radiotherapy system; (c) providing flight-control for a fly-by-wire civil airliner; (d) protecting military secrets in a system accessible to staff with lower-level security clearances?
- How would your answers change for the same question, applied to nonfunctional requirements, such as performance and usability?
- How do you measure or demonstrate the correlation between process metrics and product metrics for attributes such as reliability and security?

- Can it ever be reasonable to argue that a system is more dependable than the evidence available can demonstrate scientifically? What should be the role of engineering judgment in certifying systems?
- What do you consider to be the strengths and limitations of process metrics in assessing the dependability of a computer-based system?

Panel B: Looking Forward: New Challenges, New Opportunities

Robert Harper, Carnegie Mellon University
Shriram Krishnamurthi, Brown University
James Larus, Microsoft Research
André van Tilborg, Office of the Secretary of Defense

Moderators: *John Rushby, Lui Sha*

The focus of this panel is what has changed in the last 30 years with respect to certification.

- How have the development of new technology and the spread of computing changed both the problems we face in certifying software, and the potential solutions to the certification problem?
- How does the increasingly pervasive use of software in infrastructural systems affect the need for certification?
- Does the greater sophistication of today's users affect the problem?
- What challenges and opportunities are presented by the widespread use of COTS software and outsourcing? How can we build and certify systems in which critical and noncritical components work together?
- Should we move certification from a process-centric process to a product-centric process over time? If so, how?
- What technologies are promising for aiding certification efforts? What role will there be for static methods such as static analysis, proof systems, and model checking? And for dynamic approaches involving, for example, runtime assertions and fault detection, masking, and recovery?
- Is incremental certification in traditional safety-critical systems such as flight control an important goal to work toward? What is the technology barrier to success?

Panel C: Certification and Regulation: Experience to Date

Brent Goldfarb, University of Maryland
Mats Heimdahl, University of Minnesota
Charles Howell, MITRE Corporation
Robert Noel, MITRE Corporation

Moderators: *Michael DeWalt, Scott Wallsten*

The focus of this panel is to understand how certification and regulation affect software development.

- How do regulation and certification affect current mission-critical software development?
- What are the differences and similarities between industry-standard, self-imposed regulations, and government- or policy-imposed regulations and standards?

- How do developers consider trade-offs between improved safety/reliability and lower costs associated with less dependable (but perhaps more available) systems?
- How do regulations and certification affect innovation?
- Within your field of expertise, what are the top three issues in regulation or government oversight that hamper system dependability? What are the top three regulatory approaches that have provided significant improvements in system dependability?
- How are regulations and guidance within your organization promulgated and approved?
- What are the differences and similarities between developing regulations and guidance material for hardware dependability and developing them for software dependability?
- What are possible future challenges (economic, technological, or otherwise) with respect to current regulatory and certification approaches?
- In your answers to these questions, what supporting data are available and what supporting data are needed to buttress analyses?

Panel D: Organizational Context, Incentives, Safety Culture, and Management

Richard Cook, University of Chicago
Gene Rochlin, University of California, Berkeley
William Scherlis, Carnegie Mellon University

Moderators: *Charles Perrow, David Woods*

The focus of this panel is to explore the implications of certification within the organizational context.

- How are software development organizations responsible for failures? How can software development organizations learn from failure? How can software development as a model of operations be integrated with operations?
- How can software development better anticipate the reverberations of technology change?
- Do we need to highlight particular problems with organizational performance in the certification area, distinct from dependability in general? Are the "mental models" of organizational routines more vulnerable in this area, thus requiring more demanding safeguards or personnel? How might this be achieved? By outsourcing, special training, incentives?
- What role might insurance and liability play in achieving higher levels of certification? Might liability threats promote a better safety culture? Would the availability of insurance help (or make matters worse—the moral hazard problem)? Might insurers require evidence of reliability practices to make insurance available or reduce high premiums? Are there precedents for this in other areas of safety in low-probability/high-risk endeavors, and is there evidence the effort is successful?
- To what extent should certification be left entirely to the producer? When should a firm hire specialists? When should a consumer require that an independent agency do the certification? Are there trade-secret issues with outside involvement?
- Products are sold on the basis of performance and features. How can we make the promise of dependability attractive to consumers given its added cost?

Reactions to Panels

TUESDAY, APRIL 20, 2004

Panel E: Cost-Effectiveness of Software Engineering Techniques

Kent Beck, Three Rivers Institute
Matthias Felleisen, Northeastern University
Anthony Hall, Praxis Critical Systems

Moderators: *Peter Lee, Jon Pincus*

The focus of this panel is to understand the cost-effectiveness of current software engineering techniques as they relate to dependability and certification.

- What is the evidence for the cost-effectiveness of various software engineering techniques, either today or looking toward the future? Ideally, this would focus on the techniques' roles in producing dependable software; however, strong evidence for cost-effectiveness in other domains is also interesting.
- To the extent that evidence is currently limited, what kind of investigation could lead to strengthening it in the future?
- Are there particularly promising directions that can lead to particular software engineering techniques becoming more cost-effective for creating dependable software?

Panel F: Case Study: Electronic Voting

David Dill, Stanford University
Douglas Jones, University of Iowa
Avi Rubin, Johns Hopkins University
Ted Selker, Massachusetts Institute of Technology

Moderators: *Reed Gardner, Daniel Jackson*

The focus of this panel is to explore a particular application domain within the context of certification, dependability, and regulation.

- What role does software play in voting? How crucial is it? Does it make things worse or better?
- What properties of the software might be certified? What current approaches might help?
- What would the certification process, if any, be? Who would do it? What credibility would it have? Who has to be trusted? What ulterior motives are at play?
- With respect to issues of dependability and certification, is this case study typical, or unique in some ways?

Group Brainstorm

Moderator: *Daniel Jackson*

What are the important questions that have come out of this workshop that the committee should address in the rest of its study?

B

Panelist Biographies

PANEL A: THE STRENGTHS AND LIMITATIONS OF PROCESS

Isaac Levendel is an executive technology leader with 30 years of experience in managing large teams responsible for developing software and hardware products in telecommunications and computing. He has held leadership positions in all product development phases from the front-end to factory level and in customer support. His work has focused on on-time delivery of quality products to outside customers. Levendel has achieved international recognition in hardware and software quality assessment and prediction, fault tolerance and dependability, and software technologies. He has authored numerous publications and books and has earned several patents, awards, and honors. Levendel spent many years at Lucent and Motorola. Currently he works as an independent consultant on company start-up and delivers concentrated workshops on software development cost-effectiveness. Levendel earned a B.S. in hardware engineering from Technion Israel, an M.S. in computer science from the Weitzman Institute of Science, and a Ph.D. in computer engineering from the University of Southern California.

Gary McGraw is chief technology officer at Cigital (formerly Reliable Software Technologies). Working with Cigital Professional Services and Cigital Labs, McGraw sets software quality management technology strategy and oversees the Cigital technology transfer process. His aim is to bridge the gap between cutting-edge science and real-world applicability and to transfer advanced technologies for use in the field. In addition to consulting with major commercial software vendors and consumers, he founded Cigital's Software Security Group and chairs the Cigital Corporate Technology Council. He has written more than 50 peer-reviewed technical publications and functions as principal investigator on grants from the Air Force Research Laboratory, DARPA, National Science Foundation, and NIST's Advanced Technology Program. He serves on the advisory boards of Authentica, Counterpane, Fortify, and Indigo Security as well as advising the Computer Science Department at the University of California at Davis. He writes a monthly column on software security for *Software Development* magazine and is a department editor for *IEEE Security and Privacy* magazine. McGraw is coauthor of five popular books: *Exploiting Software*; *Java Security: Hostile Applets, Holes, & Antidotes*; *Software Fault Injection: Inoculating Programs against Errors*; *Securing Java: Getting Down to Business with Mobile Code*; and *Building Secure Software*. McGraw holds a B.A. in philosophy from the University of Virginia and a dual Ph.D. in cognitive science and computer science from Indiana University.

Peter Neumann is principal scientist at SRI International's Computer Science Laboratory. He is concerned with computer systems and networks, security, reliability, survivability, safety, and many risk-related issues such as voting-system integrity, crypto policy, social implications, and human needs including privacy. He moderates the ACM Risks Forum, edits CACM's monthly Inside Risks column, chairs the ACM Committee on Computers and Public Policy, cofounded People for Internet Responsibility (PFIR), and cofounded the Union for Representative International Internet Cooperation and Analysis (URIICA). Neumann is a fellow of the ACM, IEEE, and AAAS, and is also an SRI fellow. He is a member of the U.S. General Accounting Office Executive Council on Information Management and Technology and of the California Office of Privacy Protection advisory council. Prior to joining SRI International, Neumann was at Bell Labs, during which time he was heavily involved in the Multics development jointly with MIT and Honeywell. In addition, he has served on the faculties at Stanford University, the University of California at Berkeley, and the University of Maryland. He is the 2002 recipient of the National Computer System Security Award. He received a Ph.D. from Harvard University and the Technical University of Darmstadt.

PANEL B: LOOKING FORWARD: NEW CHALLENGES, NEW OPPORTUNITIES

Robert Harper is a professor of computer science at Carnegie Mellon University, where he has been a member of the faculty since 1988. From 1985 to 1988 he was a research fellow in the Laboratory for Foundations of Computer Science at Edinburgh University. His research is concerned with the development and application of type theory to computer programming. As a graduate student he was a charter member of the PRL Project, which pioneered the mechanization of constructive type theory as a foundation for a comprehensive proof and program development system. While at Edinburgh, Harper collaborated with Robin Milner on the design, semantics, and implementation of Standard ML. He designed and built the first implementation of the Standard ML module system, and he coauthored (with Milner and Mads Tofte) *The Definition of Standard ML*, which consists of the static and dynamic semantics of the language. Also at Edinburgh he collaborated with Gordon Plotkin on the design of the LF Logical Framework. At Carnegie Mellon, Harper, together with Peter Lee and Frank Pfenning, directed the Fox Project, which sought to apply fundamental programming language theory and advanced compiler technology to the practice of building systems. His work on the Fox Project includes fundamental research on type systems for modular programming, the development of typed intermediate languages, type-directed translation to support efficient compilation methods, and the construction of certifying compilers. Harper's current research interests are type refinements for programming languages, applications of language technology to grid computing, and the use of self-adjusting computation to implement incremental and dynamic algorithms. Harper earned a Ph.D. from Cornell University in 1985.

Shriram Krishnamurthi is an assistant professor of computer science at Brown University. His research lies at the confluence of programming languages, software engineering, and computer-aided verification. His recent work has focused on the semantics, verification, and use of new forms of software composition and interaction. He is a coauthor of the DrScheme programming environment, the FASTLINK genetic linkage analysis package, and the book *How to Design Programs*. He has more recently written the text *Programming Languages: Application and Interpretation*. He also coordinates the TeachScheme! high school computer science outreach program. Krishnamurthi earned his Ph.D. from Rice University.

James Larus is an assistant director of Microsoft Research. In his career, he has applied programming languages and compiler technology and techniques to many areas of computer science. From 1989 until 1999, Larus was a professor of computer science at the University of Wisconsin-Madison. His research covered a number of areas: new and far more efficient techniques for measuring and recording executing programs' behavior, tools for analyzing and manipulating compiled and linked programs, new programming languages, tools for verifying program correctness, and techniques for compiler analysis and optimization. In addition, he also comanaged the DARPA- and NSF-sponsored Wisconsin Wind Tunnel research project, which developed new computer architectures and programming techniques for shared-memory parallel computing. After a sabbatical at Microsoft Research, Larus decided to stay and establish the Software Productivity Tools (SPT) research group in Microsoft Research. This group has developed, built, and demonstrated advanced tools that improve the design, development, debugging, and testing of software. Larus received an A.B. from Harvard College and an M.S. and a Ph.D. from the University of California, Berkeley.

André van Tilborg is director of the Information Systems Directorate in the Office of the Deputy Under Secretary of Defense (Science and Technology). He has oversight responsibility for the information technology research programs of the military services and agencies, including DARPA. Prior to assuming this position in 2002, van Tilborg served, starting in 1994, as director of the Mathematical, Computer, and Information Sciences and Technology Division at the Office of Naval Research. He joined ONR in 1987 and was promoted to director of the Computer Science Division in 1989. From 1984 to 1986, he was employed as a research faculty member in the Computer Science Department at Carnegie Mellon University. His specialized areas of research included decentralized resource management of distributed computing systems and networks, and real-time embedded computing systems. In 1983-1984, van Tilborg was employed as principal computer systems scientist at Honeywell Systems and Research Center, where he was program manager of the Secure Ada Target trusted computer project. Prior to working at Honeywell, he was principal computer scientist at Cornell Aeronautical Laboratory, where he served as head of the Distributed Computing Division. He is the author of approximately 30 open-literature refereed technical publications and the editor of two books on real-time computing systems. He has served as conference chair and program chair for numerous international symposia, conferences, and workshops, particularly in the distributed and real-time computing systems technical areas. He holds a Ph.D. in computer science from the State University of New York.

PANEL C: CERTIFICATION AND REGULATORY EXPERIENCE AND ISSUES

Brent Goldfarb is an assistant professor of management and entrepreneurship at the Robert H. Smith School of Business at the University of Maryland. Goldfarb studies how the production and exchange of technology differ from those for more traditional economic goods, and the implications of these differences for both business and public policy. Goldfarb's research has focused on government procurement of research at universities and the sale of and subsequent commercial development of their technologies. In particular, he has asked how research funds and incentives of knowledge producers are structured, and when the uncertainty inherent in producing and describing new technologies leads to poor market outcomes. A key result of this research is that while markets are problematic mediums for technology exchange, key institutions often evolve to mitigate problems. Goldfarb earned an undergraduate degree in computer science and economics from Tel-Aviv University in 1996 and earned his Ph.D. in economics from Stanford University in 2002.

Mats Heimdahl is currently a McKnight Presidential Fellow and an associate professor of computer science and engineering at the University of Minnesota. In addition, he is the director of the University of Minnesota Software Engineering Center (UMSEC). His research interests are in software engineering, safety critical systems, software safety, testing, requirements engineering, formal specification languages, and automated analysis of specifications. He is currently pursuing his interest in the following areas: static analysis of system and software requirements, for example, through model checking and theorem proving; how dynamic methods (e.g., simulation and testing) can be used to validate requirements specifications; model-based software development; automated test case generation; and software certification. Heimdahl is the recipient of the NSF CAREER award, a McKnight Land-Grant Professorship, and the McKnight Presidential Fellow award at the University of Minnesota. He earned an M.S. in computer science and engineering from the Royal Institute of Technology in Stockholm, Sweden, and a Ph.D. in information and computer science from the University of California at Irvine.

Charles Howell is a consulting engineer for software assurance in the Center for Innovative Computing and Informatics at the MITRE Corporation. The center focuses on exploring, evaluating, and applying advanced information technologies in critical systems for a wide range of organizations. His current interests include tools and notations to support the development, review, and maintenance of assurance cases for software-intensive systems, and approaches to make large networked information systems more robust (i.e., less fragile). He is the principal investigator for a MITRE research project on high-confidence software. He recently chaired a DARPA panel developing a research agenda for building trustworthy systems and led an effort for the Office of the Deputy Undersecretary of Defense for Science and Technology evaluating science and technology requirements for software-intensive systems. Howell is the author of the article "Dependability" in John Wiley & Sons' second edition of the *Encyclopedia of Software Engineering* and coauthor of the book *Solid Software*. He is a senior member of the IEEE and holds an active Top Secret/SCI clearance. Howell holds a B.S. in mathematical sciences from Virginia Commonwealth University.

Robert Noel is a lead software systems engineer with MITRE's Center for Air Force C2 Systems. Noel has been working in military Air Traffic Control, Landing Systems, and Avionics certification programs since 1989. Most recently, he has been supporting the Battle Control System programs, which are modernizing the equipment used in the U.S. Air Defense Sectors for Air Battle Management. Prior to that, Noel was the lead software engineer for the USAF office responsible for Global Air Traffic Management (GATM) and Navigation Safety certification of USAF aircraft (1997-2002). Noel received a B.S. in math from the University of Lowell in 1984 and an M.S. in system engineering from Boston University in 1994.

PANEL D: ORGANIZATIONAL CONTEXT, INCENTIVES, SAFETY CULTURE, AND MANAGEMENT

Richard Cook is a physician, educator, and researcher at the University of Chicago. His current research interests include the study of human error, the role of technology in human expert performance, and patient safety. Cook graduated from Lawrence University in Appleton, Wisconsin, where he was a Scholar of the University. He then worked in the computer industry in supercomputer system design and engineering applications. He received the M.D. degree from the University of Cincinnati in 1986, where he was a general surgery intern. Between 1987 and 1991 he was a researcher on expert human performance in anesthesiology and industrial and systems engineering at the Ohio State University. He completed an anesthesiology residency at Ohio State in 1994. Since

November 1994 he has been a faculty member in the Department of Anesthesia and Intensive Care of the University of Chicago. He is an associate director for the GAPS (Getting At Patient Safety) project sponsored by the Veterans Health Administration. Cook has been involved with the National Patient Safety Foundation since its inception and sits on the foundation's board. He is internationally recognized as a leading expert on medical accidents, complex system failures, and human performance at the sharp end of these systems. He has investigated a variety of problems in such diverse areas as urban mass transportation, semiconductor manufacturing, and military software systems. He is often a consultant for not-for-profit organizations, government agencies, and academic groups. Cook's most often cited publications are "Gaps in the Continuity of Patient Care and Progress in Patient Safety," "Operating at the Sharp End: The Complexity of Human Error," "Adapting to New Technology in the Operating Room," and the report *A Tale of Two Stories: Contrasting Views of Patient Safety*.

Gene Rochlin received his Ph.D. in physics from the University of Chicago in 1966. Following his retraining in political science at MIT and Harvard in 1973-1974, his research interests in the cultural, social, political, and organizational implications and consequences of technology have extended to studies of nuclear power and nuclear proliferation, advanced information technologies, and the politics and political economy of energy and environmental policy. He was a principal of the Berkeley High Reliability Project, a multidisciplinary team that has studied the organizational aspects of safety-critical systems such as nuclear power operations and air traffic control. His recent book about the short-term effects and long-term consequences of the increasingly widespread "embedding" of computers as structural elements or organization, and the attendant creation of new modes of dependence and vulnerability, has led to a growing involvement in studies of potential threats not only to IT systems per se, but also to the many critical systems in society that have come to depend on them for operational reliability and security. He also teaches courses on the principles, theories, and methods of social studies of science and technology, as well as courses on social theories of risk.

William Scherlis is a professor in the School of Computer Science at Carnegie Mellon University and a member of CMU's International Software Research Institute (ISRI). He is the founding director of CMU's Ph.D. program in software engineering. He is a co-principal investigator of the 5-year High Dependability Computing Project (HDCP) with NASA, in which CMU leads a collaboration with five universities to help NASA address long-term software dependability challenges. His research relates to software assurance, software evolution, and technology to support software teams. Scherlis is involved in a number of activities related to technology and policy, recently testifying before Congress on innovation, government information technology, and roles for a federal CIO. He interrupted his career at CMU to serve at DARPA for 6 years, departing as a senior executive responsible for the coordination of software research. While at DARPA he had responsibility for research and strategy in computer security, high-performance computing, information infrastructure, and other topics. Scherlis chaired a National Research Council study on information technology, innovation, and e-government, and has led or participated in national studies related to crisis response, analyst information management, Department of Defense software management, and health care informatics infrastructure. He has served as program chair for a number of technical conferences, including the ACM Foundations of Software Engineering Symposium. Scherlis received an A.B. from Harvard University and a Ph.D. in computer science from Stanford University.

PANEL E: COST-EFFECTIVENESS OF SOFTWARE ENGINEERING TECHNIQUES: WHAT EVIDENCE EXISTS?

Kent Beck is the founder and director of Three Rivers Institute (TRI). His career has combined the practice of software development with reflection, innovation, and communication. His contributions to software development include patterns for software, the rediscovery of test-first programming, the xUnit family of developer testing tools, and Extreme Programming. He currently divides his time between writing, programming, and coaching. Beck is the author or coauthor of *Contributing to Eclipse*, *Test-Driven Development: By Example*, *Extreme Programming Explained*, *Planning Extreme Programming*, *The Smalltalk Best Practice Patterns*, and the forthcoming *JUnit Pocket Guide*. He received his B.S. and M.S. in computer science from the University of Oregon.

Matthias Felleisen is a professor at Northeastern University's College of Computer Sciences. His areas of interest include PLT Scheme (DrScheme and friends), How to Design Programs (HtDP), and How to Use Scheme (HtUS). Their development drives the analysis of current versions of Scheme (the language) and DrScheme (the programming environment). The goal is to support the entire spectrum of program development, from scripting to large complex systems. When problems are noticed, the language or the environment (or both) are modified. Changes are evaluated with respect to language design, analysis, and implementation as well as software engineering. Results are modeled and published so that others in the PL and SE community can adapt them as desired. Felleisen's primary educational project is the TeachScheme! project. Its purpose is to change the introductory curriculum at the high school and college levels. Instead of students being exposed to languages with a heavy syntax and commercial programming environments, they are introduced instead to a series of simple languages (small subsets of Scheme with a few additional constructs) and a programming environment tailored to beginners. This project thus creates a pool of users who stress-test our languages and environment.

Anthony Hall is a principal consultant with Praxis Critical Systems Ltd. He is a specialist in requirements and specification methods and the development of software-intensive systems. He has worked for many years on the development of critical operational systems. During this time he has pioneered the application of formal methods to industrial practice. He was chief designer on CDIS, a successful air traffic information system, and a certification authority developed to ITSEC E6 standards. Together with colleagues in Praxis Critical Systems, Hall has brought together extensive practical experience and the latest research findings to develop REVEAL, a principled yet practical approach to requirements engineering, and Correctness by Construction, a process for cost-effective development of critical software. Hall received an M.A. and a Ph.D. from Oxford University. He is a fellow of the Royal Academy of Engineering, a chartered engineer, and a fellow of the British Computer Society.

PANEL F: CASE STUDY: ELECTRONIC VOTING

David Dill is a professor of computer science and, by courtesy, electrical engineering at Stanford University. He has been on the faculty at Stanford since 1987. His primary research interests relate to the theory and application of formal verification techniques to system designs, including hardware, protocols, and software. He also has an interest in voting technology and related policy issues and has done research in asynchronous circuit verification and synthesis, and in verification methods for hard real-time systems. He was the chair of the Computer-Aided Verification Conference held at Stanford University in 1994. From July 1995 to September 1996, he was chief scientist at 0-In Design

Automation. Dill's Ph.D. thesis, "Trace Theory for Automatic Hierarchical Verification of Speed Independent Circuits," was named as a Distinguished Dissertation by ACM and published as such by MIT Press in 1988. He was the recipient of a Presidential Young Investigator award from the National Science Foundation in 1988, and a Young Investigator award from the Office of Naval Research in 1991. He has received Best Paper awards at the International Conference on Computer Design in 1991 and the Design Automation Conference in 1993 and 1998. He was named a fellow of the IEEE in 2001 for his contributions to verification of circuits and systems. Dill served on the California Secretary of State's Ad Hoc Task Force on Touch Screen Voting in 2003, and he is currently on the IEEE P1583 Voting Standards Committee and the Santa Clara County DRE Citizens Oversight Committee. Dill holds an S.B. in electrical engineering and computer science from Massachusetts Institute of Technology and an M.S. and a Ph.D. from Carnegie Mellon University.

Douglas Jones is currently an associate professor of computer science at the University of Iowa, where his teaching focuses on the intersection of computer architecture and operating systems. He is a member of the Association for Computing Machinery (ACM), the U.S. Public Policy Committee of the ACM, the National Committee on Voting Integrity, the American Association for the Advancement of Science, and Computer Professionals for Social Responsibility. Jones is currently vice president and chief technical officer of the Open Voting Consortium and a member of the Advisory Board of VerifiedVoting.org, and he has served for a decade on Iowa's Board of Examiners for Voting Machines and Electronic Voting Systems, of which he is past chair. In the wake of the 2000 election, he testified before the U.S. Commission on Civil Rights and the House Science Committee. He also gave the keynote address at the Second Interamerican Conference on Voting Technology, and he contributed Chapter 1 to the book *Secure Electronic Voting*. He received his B.S. in physics from Carnegie Mellon University in 1973 and his M.S. and Ph.D. in computer science from the University of Illinois at Urbana-Champaign in 1976 and 1980.

Avi Rubin is a professor of computer science as well as technical director of the Information Security Institute at Johns Hopkins University. Prior to joining Johns Hopkins he was a research scientist at AT&T Labs. Rubin is the author or coauthor of several books, including *Firewalls and Internet Security*, second edition (with Bill Cheswick and Steve Bellovin, Addison-Wesley, 2003), *White-Hat Security Arsenal* (Addison-Wesley, 2001), and *Web Security Sourcebook* (with Dan Geer and Marcus Ranum, John Wiley & Sons, 1997). He is associate editor of *ACM Transactions on Internet Technology* and an Advisory Board member of Springer's Information Security and Cryptography Book Series. Rubin serves on the board of directors of the USENIX Association and on the DARPA Information Science and Technology Study Group. He received his B.S. (computer science), M.S.E., and Ph.D. in computer science and engineering from the University of Michigan.

Ted Selker, at the MIT Media and Arts Technology Laboratory, is the director of the Context Aware Computing Lab, which strives to create a world in which people's desires and intentions cause computers to help them. This work creates environments that use sensors and artificial intelligence to create so-called "virtual sensors," adaptive models of users to create keyboardless computer scenarios. Selker is also director of a counterintelligence special-interest group on design and domestic life, a forum for discussing kitchens and domestic technology, lifestyles, and supply changes as a result of technology. He is creating an industrial design intelligence forum to discuss the need to understand cognitive science and quantitative experiments in doing product design. As part of the Caltech/MIT voting project, Selker has contributed important papers that have been useful to creating legislation. He has also helped the Carter/Ford voting project and is part of the IEEE voting standards committee. A large part of Selker's work in voting concerns inventing and testing new technology for voting. Examples include new approaches to user interfaces, registration database testers, ballot design systems, secure online architectures, and new approaches for using simulation to evaluate political platforms. Prior to joining MIT's faculty in November 1999, Selker directed the User Systems Ergonomics Research Lab at the IBM Almaden Research Center, where he became an IBM Fellow in 1996. He has served as a consulting professor at Stanford University, taught at Hampshire, the University of Massachusetts at Amherst, and Brown University, and worked at Xerox PARC and Atari Research Labs. His research has contributed to products ranging from notebook computers to operating systems. His work takes the form of prototype concept products supported by cognitive science research. He is known for the design of the "TrackPoint III" in-keyboard pointing device now found in Compaq, Fujitsu, HP, IBM, Sony, TI, and other computers; for creating the "COACH" adaptive agent that improves user performance (Warp Guides in OS/2); and for the design of the 755CV notebook computer that doubles as an LCD projector. He is the author of numerous patents and papers in refereed journals and conference proceedings. He received his B.S. in applied mathematics from Brown University, his M.S. in computer/information sciences from the University of Massachusetts at Amherst, and a Ph.D. in computer science from the City University of New York.

C

Committee Member and Staff Biographies

COMMITTEE MEMBER BIOGRAPHIES

Daniel Jackson (*Chair*) is an associate professor of computer science at the Massachusetts Institute of Technology. He received an M.A. from Oxford University (1984) in physics and his S.M. (1988) and Ph.D. (1992) from MIT in computer science. He was a software engineer for Logica UK Ltd. and an assistant professor of computer science at Carnegie Mellon University, and has been an associate professor at MIT since 1997. He has sat on the editorial boards of ACM's *Transactions on Programming Languages and Systems* and *Transactions on Software Engineering and Methodology*, and of Springer's *Software Tools for Technology Transfer*, and he has served on the program committee of more than 20 international conferences, including FSE, ISSTA, OOPSLA, and CAV. He has broad interests in several areas of software construction, including development methods, automatic analysis of designs and specifications, and reverse engineering of code.

Joshua Bloch is a principal software engineer at Google. Previously he was a distinguished engineer at Sun Microsystems, where he was an architect in the Core Java Platform Group. He wrote the bestselling book *Effective Java* (Addison-Wesley, 2001), winner of the 2002 Jolt Award. He led the design and implementation of many parts of the Java platform, including the collections framework, Tiger language enhancements (JSR-201), annotations (JSR-175), multiprecision arithmetic, preferences (JSR-10), and assertions (JSR-41). Previously he was a senior systems designer at Transarc Corporation, where he designed and implemented many parts of the Encina distributed transaction processing system. He holds a Ph.D. in computer science from Carnegie Mellon University and a B.S. in computer science from Columbia University.

Michael DeWalt is chief scientist, aviation systems, for Certification Services, Inc., a Seattle-area aviation consultancy. DeWalt is authorized by the FAA, as a consultant Designated Engineering Representative (DER), to approve software for any aircraft system, at any software level. In addition to his DER duties, he helps clients who have unusual project requirements to develop acceptable software-approval techniques. For 11 years, he was the FAA's National Resource Specialist (NRS) for aircraft software. He was responsible for starting the international committee that created DO-178B and served as its secretary. He was also secretary of the committee that created DO-248B and DO-278. DeWalt has been involved with both civil and military software avionics and certification for 26 years, working for airframe manufacturers and avionics suppliers. In addition to his DER certificate, he has a B.S.E.E., a master's in software engineering, and a commercial pilot's license.

Reed Gardner is a professor and chair of the Department of Medical Informatics at the University of Utah. He has been a codirector of medical computing at LDS, Cottonwood, and Alta View Hospitals in Salt Lake City. He is one of the principal developers and evaluators of the medical expert system known as HELP (Health Evaluation through Logical Processing). Gardner's primary academic and research interests are evaluating the benefits of medical expert systems as they relate to quality and cost-effectiveness; development of software oversight committee methods for evaluation of safety and effectiveness of medical software and systems; public health informatics; applying computers in intensive care medicine; and developing devices and communications methods to acquire patient data at the bedside. He is the author or coauthor of more than 300 articles in the fields of medical informatics and engineering. Gardner has been a journal editor and on the editorial boards of *Critical Care Medicine* and other critical care journals as well as the *Journal of the American Medical Informatics Association* (JAMIA). He is a fellow of the American College of Medical Informatics and past president of the American Medical Informatics Association. Gardner holds a B.S.E.E. from the University of Utah (1960) in electrical engineering and Ph.D. from the University of Utah (1968) in biophysics and bioengineering.

Peter Lee is a professor of computer science at Carnegie Mellon University. He joined the faculty of Carnegie Mellon's School of Computer Science in 1987, after completing his doctoral studies at the University of Michigan. He is known internationally for his research contributions in areas related to information assurance, especially the application of programming language technology to operating systems design, networking, and computer security. Lee is best known for his co-invention of the "proof-carrying code" technology for ensuring the security of mobile code. Today, proof-carrying code is the subject of several DARPA- and NSF-sponsored research projects and forms the basis for the products and services provided by Cedilla Systems Incorporated, a Java technology start-up company he cofounded in 1999. Lee is also the associate dean for undergraduate education in Carnegie Mellon's School of Computer Science. In this capacity, he has been involved in the administration of Carnegie Mellon's undergraduate programs in computer science. His tenure as associate dean has seen the undergraduate program rise to national prominence, both for its intensive problem-oriented curriculum and for its success in attracting and retaining women in the field of computer science. He has published extensively in major international symposia and is the author of two books. He has been invited to give distinguished lectures and keynote addresses at major universities and symposia and has been called on as an expert witness in key judicial court cases such as the *Sun v. Microsoft* "Java lawsuit." Lee has also been a member of the Army Science Board since 1997, for which he has served on four major summer studies, and a Technology Panel cochair for the 2001 Defense Science Board study on Defense Science and Technology. In addition to holding M.S. and Ph.D. degrees in computer and communication sciences, Lee earned a B.S. in mathematics from the University of Michigan in 1982. He has been a principal investigator on several DARPA, NSF, and NASA grants and contracts.

Steven B. Lipner is director of security engineering strategy at Microsoft. He was previously the head of Microsoft's Security Response Center. He will be responsible for defining Microsoft's security development processes and plans for their application to new product generations. His team will also define and execute new programs to help Microsoft customers deploy and operate their systems securely. Lipner, who was previously the director of security assurance, has been at Microsoft since 1999. He joined the company after working at the MITRE Corp. and Digital Equipment Corp., among others. He has almost 30 years' experience in computer and network security as a researcher, development manager, and business unit manager. He holds eight patents in computer and network security and is a member of the National Computer Systems Security and Privacy Advisory Board. He holds an M.S. (1966) in civil engineering from the Massachusetts Institute of Technology.

Charles (Chick) Perrow is a professor emeritus of sociology at Yale University. He is a past vice president of the Eastern Sociological Society; a fellow of the Center for Advanced Study in the Behavioral Sciences, 1981-1982, 1999; fellow of the American Academy for the Advancement of Science; resident scholar, Russell Sage Foundation, 1990-1991; fellow, Shelly Cullom Davis Center for Historical Studies, 1995-1996; visitor, Institute for Advanced Study, 1995-1996; and a former member of the National Research Council's Committee on Human Factors, of the Sociology Panel of the National Science Foundation, and of the editorial boards of several journals. An organizational theorist, he is the author of six books—*The Radical Attack on Business* (1972), *Organizational Analysis: A Sociological View* (1970), *Complex Organizations: A Critical Essay* (1972; 3rd ed., 1986), *Normal Accidents: Living with High Risk Technologies* (1984; revised, 1999), *The AIDS Disaster: The Failure of Organizations in New York and the Nation* (1990) with Mauro Guillen, *Organizing America: Wealth, Power, and the Origins of American Capitalism* (2002)—and over 50 articles. His current interests are in managing complexly interactive, tightly coupled systems (including hospitals, nuclear plants, power grids, the space program, and intelligent transportation systems); the challenge and limits of network-centric warfare; self-organizing properties of the Internet, the electric power grid, networks of small firms, and terrorist organizations; and the possibilities for restructuring society to reduce our vulnerability to increasing disasters, whether natural, industrial/technological, or deliberate. These grow out of his work on "normal accidents," with its emphasis on organizational design and systems theory, and reflect current consultations and workshops with NASA, the FAA, Naval War College, DaimlerChrysler, NIH, and NSF.

Jon Pincus works at Microsoft Research on software reliability tools and technologies, concentrating on static analysis. As founder and chief technology officer of Intrinsa Corporation, he was one of the original developers of PREfix and continues to be involved in its development and deployment inside Microsoft. Before that, he worked on CAD and document management systems and collected the usual degrees from the usual institutions.

John Rushby is program director for formal methods and dependable systems at SRI International. He worked at the Atlas Computer Laboratory (now part of the Computation and Information Department of the Central Laboratory of the UK Research Councils) from 1974 to 1975, as a lecturer in the Computer Science Department at Manchester University from 1975 to 1979, and as a research associate in the Department of Computing Science at the University of Newcastle upon Tyne from 1979 to 1982, before joining SRI in 1983. At SRI, he was successively promoted to computer scientist, senior computer scientist, program manager and, from 1986 to 1990, the acting director of CSL. In 1991 he assumed his current role as program director. He is interested primarily in the design and assurance of "critical systems," including properties such as security and safety, mechanisms such as kernelization and fault tolerance, and formal methods for assurance. He considers the main value of formal methods to lie in their use for constructing mathematical models whose properties can be analyzed and verified by computational means. This has led him to focus on the development of effective tools for formal methods. Rushby holds his Ph.D. in computer science from the University of Newcastle (1977).

Lui Sha is a professor of computer science at the University of Illinois at Urbana-Champaign. Before joining UIUC in 1998, he was a senior member of the technical staff at the Software Engineering

Institute at Carnegie Mellon University, which he joined in 1986. Sha's accomplishments are many, including critical assistance on NASA's Mars Pathfinder project, the application of rate monotonic theory to Global Positioning System software, and design assistance with the Air Force's F-22 Raptor project. His knowledge and application of theory and software designs to real-time computing platforms have made him an indispensable resource for numerous efforts. He is a leader in the real-time computing community, was the chair of the IEEE Real-Time Systems Technical Committee from 1999 to 2000, and received that committee's Outstanding Technical Contributions and Leadership Award in December 2001. He has consulted on many national high-technology projects, and his work has been recognized by several national leaders. He holds a Ph.D. and an M.S. in electrical and computer engineering from Carnegie Mellon University and a B.S.E.E. from McGill University.

Martyn Thomas graduated as a biochemist in 1969 from University College, London, and immediately entered the computer industry. From 1969 to 1983, he worked in universities (in London and the Netherlands), in industry (designing switching software for STC), and at the South West Universities Regional Computer Centre in Bath. In 1983 (with David Bean), he founded a software engineering company, Praxis, to exploit modern software development methods. In December 1992, Praxis was sold to Deloitte and Touche, an international firm of accountants and management consultants, and Thomas became a Deloitte Consulting international partner while remaining chair and, later, managing director of Praxis. He left Deloitte Consulting in 1997. Thomas is now an independent consultant software engineer, specializing in the assessment of large, real-time, safety-critical, software-intensive systems, software engineering, and engineering management. He is a member of the Expert Witness Institute and serves as an expert witness where complex software engineering issues are involved. He is a visiting professor in software engineering at the University of Oxford and a visiting professor at the University of Bristol and the University of Wales, Aberystwyth. He has advised the UK government and the Commission of the European Union on policy in the fields of software engineering and VLSI design. He has had close links with the academic research community throughout his career, as a member of two University Funding Council Research Assessments in Computer Science, numerous international conference program committees, and several UK government and Research Council panels and boards. He has been a member of the IT Foresight Panel of the UK Government Office of Science and Technology, a member of the Advisory Board for the DERA Systems and Software Engineering Centre, and a member of the Research Advisory Council of the UK Civil Aviation Authority. He is a fellow of the British Computer Society and of the Institution of Electrical Engineers. He currently serves on the Management Committee of the Engineering and Technology Forum of the British Computer Society, the IT Sector Panel of the IEE, the Industry Advisory Board for IEEE Software, the Advisory Group to the Foresight Cyber Trust and Crime Prevention Project, the Executive of the UK Computing Research Committee, and as a member of the Advisory Council of the Foundation for Information Policy Research. He is chair of the steering committee for the UK Interdisciplinary Research Collaboration on Dependable Systems (DIRC) and a member of the Council of EPSRC, the UK Engineering and Physical Sciences Research Council.

Scott Wallsten is a fellow at the AEI-Brookings Joint Center for Regulatory Studies and a resident scholar at the American Enterprise Institute. Before joining the Joint Center, he had been an economist at the World Bank, a scholar at the Stanford Institute for Economic Policy Research, and a staff economist at the U.S. President's Council of Economic Advisers. His interests include industrial organization and public policy, and his research has focused on regulation, privatization, competition, and science and technology policy. His work has been published in journals including the *RAND*

Journal of Economics, the *Journal of Industrial Economics*, and the *Journal of Regulatory Economics, and Regulation*.

David Woods is a professor in the Institute for Ergonomics at the Ohio State University. He was president (1998-1999) and is a fellow of the Human Factors and Ergonomic Society, and is also a fellow of the American Psychological Society and the American Psychological Association. He has received the Ely Award for best paper in the journal *Human Factors* (1994), the Kraft Innovators Award from the Human Factors and Ergonomic Society for developing the foundations of cognitive engineering, a Laurels Award from *Aviation Week and Space Technology* (1995) for research on the human factors of highly automated cockpits, and five patents for computerized decision aids. He was on the board of the National Patient Safety Foundation from its founding until 2002 and was associate director of the Midwest Center for Inquiry on Patient Safety (GAPS Center) of the Veterans Health Administration from 1999 to 2003. He is coauthor of *Behind Human Error* and has written over 40 book chapters and over 45 journal articles on problems such as human error and how complex systems fail, how to make intelligent systems team players, how to support anomaly response and diagnosis, cooperative cognition, and automation surprises in application areas such as space operations, automated flight decks, nuclear power plant safety, and critical-care medicine. His current work examines the themes of data overload, how complex systems fail, human-robot coordination, and how distributed teams modify plans in progress. Based on this body of work he has been an advisor to various government agencies and other organizations on issues pertaining to human performance and error, including the Federal Aviation Administration, Nuclear Regulatory Commission, National Patient Safety Foundation, Veterans Health Administration, and National Science Foundation. Most recently he served on a National Academy of Engineering/Institute of Medicine study panel applying engineering to improve health care systems, and on a National Research Council panel that defined the future of the national air transportation system. Woods earned a Ph.D. from Purdue University in 1979.

STAFF BIOGRAPHIES

Lynette I. Millett is a program officer and study director at the Computer Science and Telecommunications Board of the National Research Council and has been with CSTB since 2000. She is currently involved in several CSTB projects, including a comprehensive exploration of privacy in the information age, an examination of radio frequency identification technologies, and a study on biometrics, in addition to this project on certification and dependable software systems. She recently completed a CSTB project that produced the reports *Who Goes There? Authentication Technologies and Their Privacy Implications* and *IDs—Not That Easy: Questions About Nationwide Identity Systems*. Before joining CSTB, she was involved in research on static analysis techniques for concurrent programming languages as well as research on value-sensitive design and informed consent online. She has an M.Sc. in computer science from Cornell University along with a B.A. in mathematics and computer science with honors from Colby College.

Phil Hilliard was a research associate with the Computer Science and Telecommunications Board until May 2004. He provided research support as part of the professional staff and worked on projects focusing on telecommunications research, supercomputing, and dependable systems. Before joining the National Academies, he worked at BellSouth in Atlanta, Georgia, as a competitive intelligence analyst and at NCR as a technical writer and trainer. He has a master's in library and information science from Florida State University (2003), an M.B.A. from Georgia State University (2000), and a B.S. in computer and information technology from the Georgia Institute of Technology (1986).

Penelope Smith worked temporarily with the Computer Science and Telecommunications Board between February and July 2004 as a senior program assistant. Prior to joining the National Academies, she worked in rural Angola as a health project manager and community health advisor for Concern Worldwide. She also worked for Emory University as a project coordinator and researcher on reproductive health and HIV, and for the Centers for Disease Control as a technology transfer evaluator for HIV/AIDS programs. She earned an M.P.H. from Emory University and a B.A. in medical anthropology from the University of California at Santa Cruz. She is also a certified health education specialist.

What Is CSTB?

As a part of the National Research Council, the Computer Science and Telecommunications Board (CSTB) was established in 1986 to provide independent advice to the federal government on technical and public policy issues relating to computing and communications. Composed of leaders from industry and academia, CSTB conducts studies of critical national issues and makes recommendations to government, industry, and academia. CSTB also provides a neutral meeting ground for consideration of complex issues where resolution and action may be premature. It convenes discussions that bring together principals from the public and private sectors, ensuring consideration of key perspectives. The majority of CSTB's work is requested by federal agencies and Congress, consistent with its National Academies' context.

A pioneer in framing and analyzing Internet policy issues, CSTB is unique in its comprehensive scope and its effective, interdisciplinary appraisal of technical, economic, social, and policy issues. Beginning with early work in computer and communications security, cyber-assurance and information systems trustworthiness have been a cross-cutting theme in CSTB's work. CSTB has produced several reports that have become classics in the field, and it continues to address these topics as they grow in importance.

To do its work, CSTB draws on some of the best minds in the country and from around the world, inviting experts to participate in its projects as a public service. Studies are conducted by balanced committees without direct financial interests in the topics they are addressing. Those committees meet, confer electronically, and build analyses through their deliberations. Additional expertise is tapped in a rigorous process of review and critique, further enhancing the quality of CSTB reports. By engaging groups of principals, CSTB gets the facts and insights critical to assessing key issues.

The mission of CSTB is to

- Respond to requests from the government, nonprofit organizations, and private industry for advice on computer and telecommunications issues and from the government for advice on computer and telecommunications systems planning, utilization, and modernization;
- Monitor and promote the health of the fields of computer science and telecommunications, with attention to issues of human resources, information infrastructure, and societal impacts;
- Initiate and conduct studies involving computer science, technology, and telecommunications as critical resources; and
- Foster interaction among the disciplines underlying computing and telecommunications technologies and other fields, at large and within the National Academies.

CSTB projects address a diverse range of topics affected by the evolution of information technology. Recently completed reports include *Who Goes There? Authentication Through the Lens of Privacy*; *The Internet Under Crisis Conditions: Learning from September 11*; *Cybersecurity Today and Tomorrow: Pay Now or Pay Later*; *Youth, Pornography, and the Internet*; *Broadband: Bringing Home the Bits*; and *Innovation in Information Technology*. For further information about CSTB reports and active projects, see <http://cstb.org>.