Research Challenges in High confidence Systems:

Proceedings of the Committee on Computing, Information, and Communications Workshop, August 6-7, 1997

Committee on Computing, Information, and Communications
National Science and Technology Council

Table of Contents

INTRODUCTION

On August 6–7, 1997, the National Science and Technology Council's Committee on Computing, Information, and Communications (CCIC) sponsored an invitational workshop, Research Challenges in High Confidence Systems (HCS). The workshop was held at the Institute for Defense Analyses, Alexandria, Virginia, and chaired by Ms. Teresa Lunt, Program Manager, Information Survivability, Information Technology Office/Defense Advanced Research Projects Agency.

The objectives of the workshop were to explore research topics that would enable the creation of new technologies for developing and assessing high confidence systems, to recommend integrated research challenges to stimulate and focus high confidence systems research, and to suggest national goals and benefits to encourage U.S. Government interagency commitment to a national research agenda in high confidence systems.

To achieve its objectives, the workshop considered the findings of the 1995 Committee on Information and Communications (CIC)[1], and explored in more detail the gap between the needs for high confidence and the HCS state-of-the-art for high confidence, as well as the research needed to begin to fill in that gap.

BACKGROUND

The 1995 CIC report defines a high confidence system as one in which the designers, implementers, and users have a high degree of assurance that the system will not fail or misbehave due to errors in the system, faults in the environment, or hostile attempts to compromise the system. Such a system can be expected to behave appropriately within an operational context envisioned by its creators. High confidence, then, is a measure of predictability that a system will behave within established expectations. Highlights from the 1995 CIC report include:

- There must be clear benefit from such systems: their use "must exceed the costs of not using them, failure, and misuse." Initially, only those systems for which the consequences of failure are very serious are likely to receive the attention and extraordinary efforts of assuring correct and safe operation. However, the research goal should be to not only enable the achievement of high confidence systems but also to raise the possibility that the technologies developed to create such systems are eventually usable to increase the levels of confidence that can be placed in all operational computing systems.

[1] The July 1995 workshop was sponsored by the Committee on Information and Communications (CIC) (succeeded by the Committee on Computing, Information, and Communications (CCIC)) under the National Science and Technology Council of the National Coordination Office for High Performance Computing and Communications. A report on this 1995 workshop, *America in the Age of Information*, is available at http://www.ccic.gov/ccic/cic_forum_v224/cover.html.

- "…high confidence systems of the future require integration of many properties including functional correctness and safety, fault tolerance, time-critical response, and security." The report recommended that a national research agenda be developed by using high confidence research as an integrative, inter-disciplinary incentive for these related areas.
- Research is needed to develop measures of confidence, methods for predicting confidence, and measures for understanding the costs of high confidence.
- The 1995 CIC report also recommended research on system and component engineering methods, including assurance methods, for building useful high confidence systems in cost-effective ways, as well as large-scale experiments to demonstrate that high confidence systems can be built cost effectively.

WORKSHOP ORGANIZATION

The 1997 CCIC workshop was conducted in two parts: Problem Exploration and Solution Exploration. It used two groups of panelists to help stimulate these explorations. The first group comprised participants who possessed high confidence systems problems; the second comprised participants who possessed technologies or research ideas that could contribute to addressing the problems. The panelists for each session are identified in Appendix A. The panelists each briefly presented either the problems faced by their agency (or field) or the solutions that might contribute to addressing such problems.

At the end of the first day, the problems were further explored to develop the challenges, issues, and priorities for HCS research. During the second day, the solution was further explored to identify possibilities, approaches, and goals for HCS research and development (R&D).

HCS PROBLEM EXPLORATION - PANELIST VIEWPOINTS

This workshop started from a problem perspective, rather than starting with proposed solutions. The viewpoints expressed primarily by the workshop problem holders are presented in this section. The huge range of concerns expressed reflects the magnitude of the problem from both personal and organizational perspectives.

HCS PROBLEMS

DEMONSTRATING HIGH CONFIDENCE

The task before our agencies is establishing research that will provide the means that lead to high confidence in systems. We need technologies and methods that allow us to achieve high confidence in systems, to know when high confidence has been achieved, to convince someone else that high confidence has been achieved, and to integrate assurance methods that lead to high confidence into standard system development, evolution, and operations. All of this must be done in a cost-effective manner. How to measure how much confidence can be achieved through application of specific assurance techniques remains an open issue. Safe system design approaches are needed, as are means of recognizing when the use of high confidence technology has worked. As things now stand, when accidents have been prevented, no one knows it or—worse—no one believes it.

COMPLEX SYSTEM INTEGRATION

Why is it important to address high confidence now? Participants felt that it is the increased integration of systems that makes addressing high confidence systems a compelling and urgent matter. A high confidence system has to be one in which the consequences of its behavior are well understood and predictable. The trend is towards increased complexity: achieving high confidence is becoming more difficult as systems become more complex. Highly complex systems may have unknown, unpredictable, or emergent behaviors that may exhibit themselves at undesirable moments. Complex integrated and continuously evolving systems mean the simpler forms of analysis previously used to achieve high confidence may be reaching their limits. New analysis techniques are urgently needed.

New analysis techniques must take a system-level view of the problem to include not only the digital components but also the electro-mechanical and human components. Also, new technology changes the operator's behavior, consequently inducing new kinds of errors. It is important to identify—early in the design cycle—features of the software black-box behavior that will induce user errors and then remove and/or modify those features. Technology should accommodate the user rather than try to change the user.

BARRIERS TO TRANSITION OF METHODS AND TOOLS

Because of time pressures, industry does not use the methods and tools available today for achieving higher confidence. One panelist mentioned that typically four months are spent on quality assurance; systems must be safety critical to spend six months or more on assurance. The time budget for assurance is an attractive target for cutting as a development schedule slips. It takes too much investment to learn new tools and methods. Solutions are not perceived to provide enough benefit to compensate for this cost. Any new tool must give a system designer/builder a cost-effective capability to achieve something that cannot be done now, must require little training and not a tremendous amount of effort to use, and must be what is needed to do the job.

For some industries, adopting new tools may be accomplished much more readily. For example, the computer chip industry needs new tools to address critical issues (e.g., feature size reduction) that cannot otherwise be addressed. New tools are much easier to adopt when the problem they are addressing is clear and the demand is urgent. In other industries, new software tools are adopted less readily because the quantifiable benefit, in terms of cost for the increased reliability achieved through their use, cannot be identified. However, one participant noted that industry should really consider the cost-per-product-item sold; for example, it may actually cost relatively little per copy of popular consumer software to apply an assurance technique, whereas to do so for a one-time system development could be perceived as more costly. Yet because of the regulatory environment, assurance techniques are used primarily in systems (generally safety-critical systems such as aviation or space vehicles) for which very few copies are built. Another economic factor in the use of assurance tools and techniques: supplying high-quality software components to the safety-critical industries is not a big business—the volumes are low.

Not all parts of a system can be evaluated formally, and so a variety of flexible analysis tools are needed for large complex systems. Problem holders need comprehensive and usable methods that combine both informal and formal techniques. Different individuals use different problem-solving strategies and may switch strategies midway through the problem, but today's tools typically support only one strategy.

A characteristic of complex systems is that they are difficult to understand. Visualization methods could help to present complex information in multiple ways, depending on the question being asked. Analysis processes may need special tools tailored for different domains, as opposed to more general tools. Domain-specific languages, supported by common reasoning engines, may provide some efficiencies in tool development.

Working on the Right Problems

Researchers must engage with industry to make sure they work on the right problems. For example, there are too many research papers that improve on each others' recovery block algorithms: These algorithms are rarely, if ever, used in practice. Researchers must be directed toward real-world problems.

Scaling Up

Validating approaches on real systems is critical. Research approaches may not scale easily to real-world problems. A researcher cannot know the scalability of a proposed solution until it is tried on something real. Unfortunately, small research problems are different from large real-world problems. It is not simply a matter of scaling up the techniques that were developed and tested on small examples—these often were not solving the right problems anyway. HCS research must consider problems of realistic scale from the beginning. Otherwise, we may find that we are working on wrong parts of the problem.

Data Gathering

Research must focus on areas for which it is possible to obtain the data needed for the analysis. For example, analysis tools for system models do not help if the domain information to build adequate models cannot be gathered efficiently.

Designing in Safety

Reduction in accident rates results from past accident analysis and making things "fail-safe" against what has happened previously. Ten percent of past errors are technology related; however, new technology introduces new ways that things can go wrong, making it difficult to predict future accident rates. Panelists pondered whether a system can be made safe once it has been built without safety in mind in the original design. This has been tried before, for example, putting a protection system around a legacy system. But participants felt that such an approach may be too costly to get ultra-high reliability and is not as effective as an inherently safe design. This is still a controversial topic in security, where research is currently being pursued on how to "add on" security in massively deployed critical infrastructure systems, such as telecommunications and power generation and distribution, where it is too late to "do it right" in the first place.

Panelists agreed that one can't "test in" or "validate in" a quality. One cannot decide after the fact whether a system is safe. High confidence must be designed into a system and examined through a variety of confidence-generating means. In addition, some critical properties may conflict with each other, and attention must be paid to this from the beginning.

Sometimes automating the design analysis process loses the opportunity for the designers to reason about that process and understand it. We need ways to help humans to reason about a process so they can bring all their expertise into understanding that process. Fault tree analysis[2] serves as an example. We need to discover common-sense engineering principles for establishing properties such as safety and security in systems. Learning from the causes of past accidents, builders need to impose a discipline on their designs to minimize the introduction of additional complexity.

PANELIST VIEWPOINTS

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

NASA's system development programs have long time horizons. These systems belong to the category of critical systems requiring high confidence because of the harsh remote environment of space in which autonomous (e.g., deep space robots) and semi-autonomous systems must function and because of man-rating requirements of the manned space program. Satisfying these requirements means that the system meets specified criteria for human safety. Tolerance for loss of life in civilian aviation or in the manned space program is very low. Tolerance for mission failure in high cost space programs is also very low. These low tolerances and increasingly complex mission requirements present difficult challenges to system builders. Extraordinary—though cost-constrained—efforts must be taken to ensure that failure, malfunction, or loss incidents are maintained well below such tolerance levels. Manned missions to Mars are impractical today due to extremely high costs of developing and integrating technologies that would provide high confidence that such missions could succeed.

NASA defines high confidence as knowing to some degree of predictability about a system's behavior in a given environment under specified usage conditions. Such understanding can be achieved through design analysis, demonstrated behavioral performance (simulation and testing), and detailed rigorous analytical review (verification and validation (V&V)). The confidence one has is the result of:

- The belief in the production process (design and implementation, including in-process evaluation)
- Observed system behaviors over time, including independent V&V

[2] Fault Tree Analysis (FTA) is a form of safety analysis widely used in the aerospace, electronics, and nuclear industries. The technique was originally developed at Bell Laboratories in 1961 to evaluate the Minuteman launch control system for unauthorized (inadvertant) missile launch. Leveson and colleagues have extensively explored this technique in software safety analysis and requirements specifications and modeling.

- Separate system qualification program (criteria and process).

The high confidence system goal, from an engineering point of view, is behavioral predictability: no surprises.

What is new in NASA systems is the increased scale of integration¾highly complex systems with many undiscoverable potential behaviors. Space programs are enabled through automated systems, and efficiencies of such systems are achieved through increased integration. In civilian aviation, flight safety concerns are increasing as air traffic control, navigation, and aircraft avionics become more highly integrated. Novel software architectures using artificial intelligence and neural networks are stretching the behavior understanding that can be achieved through state-of-the-art software V&V techniques.

Current software engineering practices are human-centered activities. They are informal and often create “communication air gaps” that foster misunderstandings between requirements setters and designers that may lead to errors. Such practices do not adequately address very large or complex systems. New system engineering productivity improvements are also needed. An incredible amount of tedium in the software engineering process must be addressed by more automation. The software engineering environment has a very low level of automated tool support compared to other engineering disciplines. For example, high-level languages and their associated tools (compilers/debuggers) have provided the biggest productivity improvement in forty years. There is a continuing need for productivity-enhancing engineering design and development tools, such as compilers that are flight qualified. These could include tools that provide higher levels of abstraction and tools to automate software synthesis.

V&V technology, with the exception of extensive work in system testing, has largely been ignored by industry. There is still a relatively high tolerance of software bugs that would not be tolerated in hardware, and at the same time a hesitation to trust software in critical applications. NASA designs for testability and then tests. It would like to extend V&V to be a more integral part of the early design process, and NASA wants tools supporting automated V&V throughout the system life cycle. Currently, NASA is working to shift from informal communications and models to formal syntactic processes capable of manipulation by machines to produce traceable, reusable, computer-based artifacts.

FEDERAL AVIATION ADMINISTRATION

The FAA currently uses a set of heuristic dependability standards oriented around sixty-six objectives. More than 250 industry representatives voted on these standards. These standards address different levels of confidence in dependability, but are not explicitly safety standards.

Some of the future drivers for FAA high confidence systems will be:

- Reducing the number of accidents per flight hour as the number of flights and overall passenger throughput increases
- Reducing the costs of aviation safety by a factor of five

- Increasing system efficiency through increased system integration

Industry wants the dollar-cost-per-airplane to drop greatly over time. Part of achieving such cost efficiency is putting more airplane functions into software. As the complexity of those functions increases, the problem of attaining confidence gets worse. However, no civilian aviation accidents have yet been attributed directly to software. Pilot error is the largest attributed cause. It is unknown how many aviation accidents were aided by problems in software.

The air travel system is currently near capacity. Current accident rates per flight hour, after having fallen due to corrections to planes made as the result of failure analysis following accidents, have been stable over the past ten years. Over the same time, air travel (number of flights) has increased faster than linearly. Even if the rate of accidents per flight hour stays stable along the current trend, we can expect to see one aircraft accident per week as the number of flights increase over the next twenty years. The public is unlikely to tolerate such a high frequency of aircraft accidents.

Increased integration of disparate systems will increase the complexity of such systems. In the future, avionics, air traffic control, and navigation will be tightly coupled. To gain increased air traffic throughput, current safety margins (e.g., the "cocoon" in airspace around an aircraft) will be reduced. The current trend in aviation is to design human sensors out of the system. Such design approaches (e.g., fly-by-wire) increase the potential for human-generated error when the "human machine" is not appropriately considered as part of the system. For example, the European consortium Airbus designed a system that was non-intuitive to the experienced pilot in that it had non-moving throttles in the aircraft. The design introduced the potential for human error because human pilots have learned to subconsciously see the moving throttles out of the corners of their eyes.

"Free flight," in which the aircraft automatically negotiate their desired airspace with each other as well as occasional ground controllers, is viewed as a replacement to conventional air traffic control, and is already being demonstrated in experimental programs. Such aircraft have increased dependence on various inter-communicating automated systems for flight from takeoff to landing. This increasing dependency on communications with free flight leads to concerns about the opportunity for terrorists to interfere with flight safety. At the same time, it also raises questions about the safety and predictability of interacting autonomous systems.

Systems safety is different from component safety. Air safety cannot simply be satisfied by certifying the aircraft. For example, navigation depends upon the Global Positioning System (GPS), and flight safety depends on air traffic control. The HCS challenges faced by the FAA include achieving a higher degree of air traffic system dependability (i.e., the system correctly does what is expected) that subsumes safety. The ground component must have increased reliability. The airborne component must focus on safety (from the regulator's view) and on dependability (from the aircraft producer's view). The FAA is studying how system confidence should be gained and safety measured, considering the dependence on ground systems that have several million lines of code. Safety for the composite system may be different from what is determined for its constituent components.

Complexity resulting from increased functionality and integration will lead to increased costs unless a means to reduce these cost pressures is developed. Industry thinks certification costs too much now. The FAA thinks that an order of magnitude improvement occurred in the Boeing 777 aircraft flight certification through the integrated product team (IPT) approach and the automated design and development support. This is significant since the 777 is a more complex system than earlier aircraft designs. Cost-effective approaches that support *a priori* behavioral provability—not after-the-fact provability—are desired. We need a means to assure behavioral properties at design time so as to incur less risk in building the system (e.g., so we will not have to return to an earlier phase of the development process and make significant changes).

FOOD AND DRUG ADMINISTRATION

The FDA faces the problems of determining:

- What devices or products are or should be high confidence systems
- When a device or product actually is a high confidence system
- Who decides that a device or product meets the requirements to be treated as a high confidence system

Operating on a budget of about $1 billion per year, the FDA is responsible for regulating tens of thousands of different products in many different domains¾this equates to about one FDA employee per product manufacturer. There are only two software engineers within the FDA to address HCS issues for products asking for approval or reapproval as medical devices. Moreover, the FDA does not regulate products that are not sold as medical products, and it cannot prevent a physician from using consumer products (that are not regulated) in practicing medicine. One problem is that about 50 percent of the manufacturers who develop products that are classified as safety critical have fewer than fifty employees. Most people in these companies have not had any safety training, and they do not know they should be doing safety-related development work.

The FDA is composed of five centers: (1) Drugs, (2) Biologic Products, (3) Food, (4) Animal Feed/Medications, and (5) Medical Devices. Drugs deals with statistical analyses of clinical results and qualification of software tools used in manufacturing of drugs. Biologic Products deals with software that runs blood banks. Food deals with all aspects of food processing. Animal Feed/Medications deals with medications that go into animals and how to prevent undesirable side-effects in the human food supply. Medical Devices deals with qualifying CAT scanners, laser surgery devices, MRI devices, pacemakers, and other medical devices. In all of these areas, software devices may be submitted for approval. The FDA is faced with consequence-management questions such as: Is society better off to have people die from heart failure in the absence of pacemakers than to have a one percent failure of pacemakers? Is such a failure rate good or bad?

The FDA also has no regulatory authority over hospital informatics.[3] These systems are being interconnected to medical devices. Such integration can make a low-confidence technology, such as a digital thermometer, into a critical system component requiring high confidence qualification when it is plugged into a hospital information system to control the administration of a drug. This sort of integration will be an increasingly significant issue for health care safety.

Confidence in medical devices has depended critically on the expertise of the user. Manufacturers may claim that a device does not have to be high confidence because any doctor using it will catch errors. However, the next generation of nurses and doctors will not have this expertise because they have been trained using this device and rely upon it completely. Devices in the past that weren't high confidence systems (e.g., digital thermometers) are now becoming part of a system requiring high confidence; devices are being approved for one purpose and then being marketed or used for another.

U.S. medical technology is a significant component of the U.S. share of the global economy. Our country has 62 percent of the global medical technology market. Regulations that affect the costs of production of such technology or its time of entry into the marketplace are of significant concern to technology producers. There are economic pressures to get safety evaluations out of the way. Such regulation is problematic when there is an inadequate understanding of what contributes to device safety. A set of expected "best practices" and complete hazard analyses of information-based safety-critical systems are needed.

The FDA would like to get out of the certification business and instead require the use of established, open, standards and protocols with third-party certifiers that would be regulated by FDA. This third-party evaluation and qualification process should also reduce the time to get products approved. The international IC-1508 (draft) standard, Functional Safety: Safety Related Systems, is moving in that direction and will lead the way. Individual sectors will be required to extend IC-1508 (i.e., through the use of sub-standards). The IC-1508 standard requires documentation and responsible engineering (e.g., engineering accountability). A supplier has to make a safety case and bring it to a third-party reviewer. Europeans are leading this standard development.

The IC-1508 standard is contained in seven volumes¾so absorbing, understanding, and applying the requirements of this standard is quite a hurdle. The standard is not open: the FDA had to fight for eighteen months to get an FDA person on the committee producing the standard. Hundreds of products have been through the IC Standard's certification process. In

[3] Medical informatics encompasses the application of computers and automated information to support medical practice, research, training, and education.

contrast, the FDA, with its 510(k) standard[4], asks a company to write a plan for how it will demonstrate safety as opposed to mandating a particular process. There are few checks on what goes into such plans. Some who are experienced in safety issues think this is a better approach than a fixed certification process. The FDA also feels strongly that the certification and professional licensing of system engineers, software engineers, and safety engineers is important.

The two major areas in which the FDA would like to see increased confidence are the Internet and Microsoft's Windows. The hazards that can result from operating system or Internet failures must be addressed: Today's practitioners within the health care industries assume that they can use these technologies safely in applications such as remote surgery, transferring X-rays and CAT-scans to remote doctors, and operating medical devices (e.g., one for turning surgical laser on and off).

Addressing these concerns would help protect the U.S. market share in the world-wide drug and device market. Currently, U.S. producers have the market advantage, and addressing the issues of high confidence in medical devices would help preserve U.S. competitiveness and increase overall U.S. export market share of such devices.

NATIONAL INSTITUTES OF HEALTH

"First, do no harm!" This credo of medical practice is becoming more difficult to honor in today's technology-based medical practice.[5] The medical community has no formal safety mechanisms to ensure that technology does not induce harm in their patients. This is becoming increasingly critical as medical devices move from stand-alone analog devices to more highly integrated digital systems. This problem is exacerbated when medical practitioners use commercial operating systems, such as Windows, that were never intended for use in safety-critical systems.

---

4 Medical devices are regulated under the authority of the medical device amendments to the Federal Food, Drug, and Cosmetic Act of 1938 [21 Code of Federal Regulations (CFR)]. The 1938 Act required devices to be safe but placed the burden of proof to remove unsafe products on the Government. The 1976 Medical Device Amendment and 1990 Safe Medical Devices Act established a comprehensive scheme of regulation. They defined the term "device," provided for classification of all medical devices into three classes, required device manufacturer registration and listing of products, and set up procedures for clinical investigations (Investigational Device Exemption), premarket notification (510(k)) , and premarket approval. In addition they included the basic prohibition on misbranding and adulteration, required adherence to good manufacturing practices and provided for post market surveillance (of select devices).

Section 510(k) of the Food, Drug, and Cosmetic Act requires those device manufacturers that must register to notify FDA, at least 90 days in advance, of their intent to market a medical device. This is known as Premarket Notification —called PMN or 510(k). It allows the FDA to determine whether the device is equivalent to a device already placed into one of the three classification categories. Thus, "new" devices (not in commercial distribution prior to May 28, 1976) that have not been classified can be properly identified.

5 Hippocratic oath (hÓp¥e-kr‡tπÓk ÷th) *noun.* An oath of ethical professional behavior sworn by new physicians, attributed to Hippocrates. Varying versions have been taken for 2,000 years by physicians entering the practice of medicine. At one time the oath was thought to come from ancient Greek physician Hippocrates, but modern research has shown that it most probably originated in a Pythagorean sect of the 300s BC. (From the Microsoft Bookshelf98.)

The medical industry accounts for one-seventh of the nation's Gross National Product. Managed care is pushing the risk onto the individual patient who has the least knowledge and least leverage in the system.[6] One study showed that 28 percent of patients are affected by "medical errors."[7] These errors surfaced in medical rounds, the meetings in which nurses and doctors discuss things that went wrong. The rate of negligent errors may be higher today than before … opportunities have been increased by the complexity of modern technology…[8]The near future could bring large malpractice suits aimed at managed health care companies. The American Medical Association (AMA) now admits to the existence of medical error and is establishing a foundation to fund studies on medical error and research to systematically reduce it.[9] The foundation will be funded through voluntary contributions from individuals.

The Consortium of Hospital Informatics Manufacturers (CHIM) is an organization of vendors of medical information systems manufacturers. The vendors have developed a policy for medical information system regulation.[10] American Medical Informatics Association (AMIA), an organization of hospital informatics managers, also developed a policy for medical information system regulation, including criteria and a methodology for evaluation.[11] However, these policies do not deal with engineering or technical data and have no bearing on high confidence in systems.

NUCLEAR REGULATORY COMMISSION

The mission of the Nuclear Regulatory Commission (NRC) is to protect the health and safety of the public and the environment from hazardous consequences resulting from the use of nuclear materials in the U.S., and to provide for the national security of nuclear materials. The Commission addresses system assurance through system safety reviews of applicant submissions and license modifications. Nuclear Regulation 0800, Section 7, Standard Review Plan, Instrumentation and Controls, provides a standard review plan. NRC's Office of Research develops tools and the basis for technical assessment to support the reviews. The Safety Evaluation Report documents the results of these reviews assessing a licensee's conformance to standards. The report is peer reviewed by NRC management and the Reactor Committee for Safeguards before a license or a modification to a license is granted.

There are 109 nuclear power plants in the U.S. These plants produce about 20 percent of the Nation's electricity. The NRC is concerned with two particular types of systems in these

[6] Smith, Richard, Editor, "The Future of Health Care Systems," *BMJ*, Vol. 314, 24 May 1997, pp. 1495-1496.

[7] Feinstein, Alvan, R., MD, "System, Supervision, Standards, and the 'Epidemic' of Negligent Medical Errors," Commentary, *Archives of Internal Medicine*, Vol. 157, June 23, 1997, pp. 1285-1288.

[8] Leape, Lucien, L., MD, et al., "Systems Analysis of Adverse Drug Events," *The Journal of the American Medical Association*, Vol. 274, No. 1, July 5, 1995, pp. 35-43.

[9] American Medical Association, *Prospectus*, "The National Patient Safety Foundation," [n.d.].

[10] Center for Healthcare Information Management, position paper on *FDA Draft Policy to Regulate Computer Products*, November 22, 1996. See also the CHIM Web site for a list of its publications: http://www.chim.org/Publications.html.

[11] See the AMIA web site for a list of its publications: http://www.amia.org/pubsbroc.htm.

power plants. The first is the Operational Process Control System that controls the operation of the plant (e.g., feedwater control). The second is a separate and independent safety system that takes over in case of an operational failure and provides safe shutdown (e.g., shuts down fission process and initiates cooling if there is a threat to the safe operation of the plant). Security is associated with these systems. They use one-way, out-only communications, usually under administrative control. Such systems may also have access control imposed.

Nuclear power plants have metrics of safety; for example, how often a safety trip ("scram") happens. Safety is defined only in terms of the safety trip (scram) system, not the control system. One out of every three trips is initiated by plant operators. Curiously, automatic trips are considered bad whereas operator trips are considered good. A high confidence goal could be to safely reduce the number of scrams. This could also provide economic benefits through increased safety plus lower cost of operation.

The operational control and safety systems have been largely analog hard-wired systems. These systems were extremely sensitive to operate and generally took the best operator in the plant to start them. Digital control systems have been introduced to replace the older analog systems. Digital systems have increased the functionality of such control systems and have resulted in improved system operations. However, digital systems introduce new issues in the review process. For instance, safety standards were designed for the analog technology and current safety standards do not easily extend to digital control systems. What should the review plan look like for such systems? Lawrence Livermore National Laboratory (LLNL) recently completed a standard review plan to conduct reviews of digital systems. Until now, most of the standard review plans had conformance linked to analog hard-wired standards.

To address these new issues in nuclear power digital control systems, the National Research Council was asked to review the safety and reliability issues associated with nuclear power plants. This study was initiated in October 1994 and a final report has been recently issued.[12] It identified inadequate systems engineering as the source of most problems. The study proceeded in two phases: first to define issues (six technical and two strategic issues were identified); and second, to identify criteria for review and acceptance of digital instrumentation and control (I&C) technology. The six technical issues were:

- *Systems aspects of digital instrumentation and control systems technology*: Four recommendations were made, including that the Nuclear Regulatory Commission obtain systems review guidance from other government agencies.
- *Software quality assurance¾process and product*: Four recommendations were made, including increasing the emphasis on early aspects in early phases of design; the National Research Council noted that 50 percent of errors are introduced in the requirements stage. To implement this particular

[12] National Research Council, *Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability*, National Academy Press, 1997. This may not be available on the Internet; the bibliographic description came from the Open Communications, Inc., Web site: http://www.opengroup.com/open/books/030/0309056470.shtml.

recommendation requires knowing how to assess the completeness of requirements and how to prevent design errors.

- *Common-mode software failure potential*: Four recommendations were made, including having the Nuclear Regulatory Commission (1) revisit its guidelines on the use of techniques to preclude common-mode failures in redundant systems, and (2) examine the adequacy of software diversity, functional diversity, different operating systems, and hardware diversity in its systems.
- *Safety and reliability assessment methods*: These are metrics and methods for estimating the failure of systems, (e.g., probabilistic risk assessment).
- *Human factor and human-machine interfaces*
- *Dedication of commercial-off-the-shelf hardware and software*

The strategic issues were:

- Case-by-case licensing
- Adequacy of the technical infrastructure (e.g., qualifications of Commission review personnel)

The National Research Council's recommendations were controversial, and not all of the study participants endorsed the final study report. For further information regarding safety and reliability issues for digital instrumentation and control systems in nuclear power plants (including the standard review plan), see the Commissions' web site at www.nrc.gov.

FEDERAL RAILROAD ADMINISTRATION AND STATEPUBLIC UTILITY (RAILROAD) COMMISSIONS[13]

The Federal Railroad Administration (FRA), an operating unit within the U.S. Department of Transportation, is the safety regulatory authority for the nation's freight, intercity passenger, and commuter railroads. Its regulations cover railroad operating practices, equipment, track, and train control. The freight railroads are, with the exception of a few short lines, privately owned. Amtrak, the intercity passenger operator and a contract operator of some commuter lines, is by statute a for-profit corporation, although heavily subsidized by the Federal government. It owns about 500 route miles of track in the Northeast Corridor and about 100 route miles in Michigan, and elsewhere operates on tracks owned by freight railroads or by states. Commuter railroads are generally owned by state agencies. Some of their trackage is owned outright, and some of their operations take place on track owned by freight railroads or Amtrak.

The freight railroad industry was deregulated in 1980 by the Staggers Act and, as a result, has had the incentive to make substantial capital investments to improve the quality of its rolling

[13] This section reflects post-workshop corrections made pursuant to FRA review. The updated material was provided by Steve Ditmeyer, FRA.

stock and physical plant over the past 18 years. Nevertheless, train collisions and overspeed accidents continue to occur, and the National Transportation Safety Board (NTSB) has, for over a decade, been recommending that railroads install new communications-based positive train control (PTC) systems, and has been recommending that FRA issue regulations requiring that PTC be installed on the nation’s railroads. The NTSB has placed PTC on its “most wanted” list of transportation safety improvements.

Starting in the 1880s, U.S. railroads used track circuits to detect the presence of trains in a physical block of track, and gravity relays to pass information between blocks to preclude the presence of more than one train in a particular block. The track circuits also provide a mechanism to determine track integrity. This technology and philosophy, called “automatic block signaling” (ABS) remains in use on about 15 percent of the track in the U.S.

In the 1920s railroads started to adopt “centralized traffic control” (CTC) on more heavily trafficked lines. CTC is overlaid on automatic block signals and involves the remote control of switches and signals by dispatchers at control centers. CTC is in place on about 35 percent of the track in the U.S. today.

Starting in the 1930s, some railroad lines were equipped with cab signaling, in which the wayside signal indication is also displayed in the cab, and/or “Automatic Train Control” (ATC), in which there is on-board enforcement of the signal indication. Coded messages are transmitted through the rails and picked up inductively by coils mounted on the locomotive to permit the signal information to be displayed and acted on in the locomotive cab. A very small portion (less than 5 percent of total national mileage) of the ABS and CTC territory is currently equipped with such systems.

In the 1970s, railroads began to use microprocessors to emulate the relays in their signal systems. Communications between relays historically took place over open wire pole lines, but more recently has used coded track circuits. About 50 percent of the railroad track in the U.S., generally with lighter traffic density, still has no signaling system in place at all and is controlled by voice radio instructions. It is referred to as “dark territory” or “Track Warrant Control” (TWC) territory.

The NTSB has noted that, even though ABS and CTC signal systems are very reliable, it is still possible for individuals to make mistakes that cause collisions or overspeed accidents. In TWC and ABS territory the mistake can be made be either an engineer or dispatcher, while in CTC territory, the system prevents dispatcher-caused accidents, but a mistake by an engineer can still cause an accident.

Since the early 1980s, the railroad and railroad supply industries have been conducting research and testing of PTC systems. PTC systems are made up of digital data links connecting locomotives, maintenance-of-way equipment, wayside base radios, and control centers; onboard computers, positioning systems, data radios, and display screens on locomotives and maintenance-of way equipment; and control center computers. Positioning systems on the vehicles inform the control centers of their location via the data link. The dispatcher and the control center computer issue movement authorities to the vehicles via the data link. On-board computers compare actual location with the movement authority, and if there could be a violation of the authority, the on-board computer stops the train.

With traditional signal systems, the logic and “vitality” of the system reside with the relays and microprocessors along the track. With PTC, the intelligence of the system is moved to the control centers and the vehicles. Detailed track map data bases are carried in both the vehicle on-board computers and control center computers to permit comparison of real-time location with movement authorities. PTC can allow locomotive engineers to request that a switch be thrown in front of the train. The control center, before issuing the instruction to the switch, must be certain that this action will not cause derailment. Track circuits would probably be retained with PTC to provide broken rail protection. Wayside interface units provide radio communications to and from switches, and from wayside detectors and the track circuits.

The PTC digital radio data links between locomotives, maintenance-of-way equipment, and wayside base radios use either VHF or UHF radio frequencies assigned to the railroad industry by the Federal Communications Commission. Messages move from the base radios along the wayside to control centers along backbone networks consisting, generally, of microwave radios, copper cable, and/or fiber optic cable that are owned by railroads or by commercial telecommunications companies.

PTC systems that have been developed and tested have used a variety of positioning systems: dead reckoning using an odometer, transponders between the rails, inertial platforms, Global Positioning System (GPS), and differential GPS (DGPS). Current interest is focusing on DGPS augmentation of dead reckoning for cost and efficiency reasons. DGPS provides sufficient precision (2 meters) to differentiate between trains on parallel tracks that are typically four meters apart. FRA is cooperating with the U.S. Coast Guard and the Federal Highway Administration to expand the Coast Guard’s current Maritime Differential GPS network into a Nationwide DGPS system using soon-to-decommissioned U.S. Air Force Ground Wave Emergency Network (GWEN) towers. FRA and the Coast Guard have converted a GWEN site at Appleton, WA to provide DGPS correction signals for a PTC demonstration project in the Pacific northwest.

The braking system used on freight trains today uses compressed air both to provide the braking force and to provide the medium for transmitting brake application and release commands along the length of the train. In 1993, the first practical electronically-controlled pneumatic freight train braking systems were developed to provide faster, more reliable train braking and shorter braking distances. Two different approaches are currently being tested; one uses an electrical cable with connectors between each car to transmit the application and release commands, and the other uses a spread-spectrum radio on each car to transmit the commands. In both cases air hoses remain to supply air for recharging the braking system. Also, in both cases, the communications medium provides a way to transmit information about the “health” of the braking system, the car, and its contents to the locomotive where it can be sent to other locations on the railroad over the PTC digital data link.

The railroad industry, using the freedoms and incentives provided by the Staggers Act, also downsized their physical plant and workforce over the past 18 years. However, the booming

U.S. economy of the past several years has changed the situation on many railroad lines from one of over-capacity to one of significant capacity constraints. PTC offers the possibility of increasing the capacity on existing railroad lines by shifting from a fixed-block to a moving-block operating strategy. Studies done for the railroad industry and FRA indicate that PTC can also improve transit time, transit time reliability, and asset utilization.

Major impediments to the implementation of PTC on the Nation's railroads appear to be the amount of capital required for PTC ($3 to 4 billion for the U.S. railroad industry) especially in view of other capital requirements, lack of confidence by senior executives about the reliability of the new (to railroads) PTC technology that will change the paradigm under which railroads are operated, and concern about liability if it is acknowledged that PTC systems are safer.

FRA is funding PTC demonstrations on high-speed passenger corridors in Michigan and Illinois jointly with the respective state department of transportation, the development of interoperable PTC locomotive hardware by three major eastern railroads, and the first phase of a PTC implementation project on the Alaska Railroad. FRA has also initiated a rulemaking on PTC under the auspices of the Railroad Safety Advisory Committee, a collaborative rulemaking body consisting of railroads, railroad unions, railroad suppliers, and other interested parties. In conjunction with that activity, FRA is also carrying out corridor risk and business benefit analyses of PTC.

FEDERAL TRANSIT ADMINISTRATION AND STATE PUBLIC UTILITY (PUBLIC TRANSIT) COMMISSION

The Federal Transit Administration (FTA) is a funding agency, not a regulatory agency. Its primary role is to choose which public transit projects to fund and has a small research budget. FTA works with state and local authorities as well as contractors during project formulations to ensure the project requirements meet Federal funding standards. FTA also monitors the project implementation to ensure it stays within cost and schedule constraints. An example of such a project is the Los Angeles Green Line, which is the light rail line that runs east-west through Los Angeles County and is administered by the Los Angeles County Metropolitan Transportation Authority.

The California State Public Utility requirements documented for the Los Angeles Green Line Project include the following quantitative requirements, which are characteristic of future automated train control systems:

- Safety: MTBHE (mean time between hazardous events) for the entire system of greater than 100,000 years for a maximum size system (corresponds to the same kinds of safety numbers seen in aeronautics).
- Availability: Less than thirty minutes of downtime per year.
- Performance: 100,000 application operations per second. The program to pick up magnetic signals from the track is computationally expensive.

- Maintainability: MTTR (mean time to repair) of less than one hour.
- Reliability: MTTF (mean time to failure) of at least 20,000 hours.
- Flexibility: Technology independence.
- Modularity: User programmable and user friendly. Cost effective and efficient.
- Transparency: Integrate safety assurance and fault tolerance techniques so that it is a natural part of the system.
- User programmable: The user's responsibility is that the algorithm is correct, the system builder's responsibility is that the algorithm executes correctly on the computer.

The goals of this Green Line Project include:

- Maximize the use of commercial hardware and software
- Minimize custom hardware and software
- Support long-term maintenance (twenty-five years)
- Minimize time-to-market
- Minimize sole sources of components
- Minimize cost
- Minimize reengineering from project to project and re-proving of the safety over and over

A critical aspect of infrastructure projects such as this is the long-term maintenance issues and the speed of technology advances that lead to technology obsolescence. For example, maintenance must be concerned about minor variations in the same model number of microprocessor that come out every few months. The current approach is to stockpile parts and get, for example, Motorola to agree to provide the microprocessor masks to the builder/maintainers if Motorola ever stops manufacturing that part.

In pursuing one train control project, researchers developed a model of hardware that the software was going to execute on, then they executed the actual software on that model to identify problems. They simulated unknown errors using fault injection techniques to exercise the exception handling and diagnostics in the software. However, still lacking is a means to obtain quantifiable safety for the complete hardware/software system.

Federal Highway Administration and State Public Utility (Highway) Commissions[14]

The Federal High Administration (FHA) and the State Public Highway Commissions regulate transportation on the Nation's highways. Their regulatory issues are evolving toward computer-based control technology as a means to increase throughput over these highways while reducing accidents attributable to human error. Estimates of human error as a contributing factor in the Nation's 10.7 million automobile accidents each year are as high as 90 percent. Human error is attributed to more than 30 percent of the 40,000 annual fatalities that result from those automobile accidents.

The U.S. Congress, through The Intermodal Surface Transportation Efficiency Act (ISTEA) of 1991, created the National Automated Highway System Consortium as a public-private effort with $200 million in funding to coordinate research in the field of automated highways. Several of its experimental projects were first demonstrated on August 6, 1997, over a seven-plus mile stretch of San Diego highway in the controlled-access car pool lanes. One experiment used inexpensive video cameras to control the hands-off driving of a bus at 45 mph, including speed changes and obstacle avoidance. Another experiment, with the goal of doubling or tripling lane capacity at high speeds, used radios and ceramic-magnets embedded in the roadway to control the distance between vehicles to create high-speed convoys with vehicle separation of as little as twelve feet.

HCS Problem Exploration - General Discussion

The HCS problem is one that should be seen with a renewed sense of urgency as needing increased government funding attention. The most significant change since the 1995 CIC report is the *degree* and *rate* of change in scale and integration in critical information-based systems. This leads to many concerns, the most important of which are:

- The inability to define and predict interactively complex behaviors of highly integrated, large-scale systems that are life critical, safety critical, enterprise critical, or national security critical.
- The growing exposure of critical systems, previously operated in a closed environment, to an open, widely interconnected environment that is much less predictable and increasingly hostile.
- Increasing societal risks resulting from unanticipated and often unpredictable behaviors in large-scale, highly integrated, critical information-based systems.

[14] Because the 1997 workshop made note of automated highways requiring high confidence systems during its discussions, this section was added during the editing process. Details not actually discussed were incorporated in the spirit of providing a more complete perspective of the HCS problem space. The source of this material was *The San Francisco Chronicle*, page A5, August 12, 1997.

We can no longer accurately predict the potential consequences of misbehavior in such systems.

Pressures to achieve cost-saving or profit-enhancing system efficiencies through larger scales of integration are outrunning our engineering capabilities to design and assure predicable behaviors in complex systems. The tools and techniques in use today are primarily products of 1960s fundamental research that was focused on smaller-scale information technology components and federated approaches to system design, and are no longer adequate. The research results we purchased back then have long been used up; new fundamental results are needed.

This exploration of the HCS problem attempts to balance both the enabling-capability and the risk-oriented views. The danger of over-dramatizing the risk perspective is that such characterizations become largely ignored in the absence of catastrophic events that demonstrate such risks to society at large. The danger of overselling the enabling opportunities resulting from new system-level engineering approaches is that such approaches rarely achieve significant changes in the short term and often take much longer than expected. Regardless of the motivation, today's technical shortcomings are in engineering increasingly complex systems.

High confidence systems are those systems whose consequences of undesired operational behavior are great. Use of the term "high" implies degrees of confidence can be established. We want to accurately predict with a high degree of confidence how the systems will behave, rather than calculate from historical accident records how the systems did behave. What yardsticks do we have? These yardsticks may include degree of predictable behavior, freedom from surprise (the dual of predictability), tolerance of anomalous situations (bad inputs, bad states), and accumulation of evidence (both from the system development process and its operational behavior over time). Today, we largely depend on "good" engineering practices and sometimes collect data on those practices that work. However, process attributes do not directly inform one as to system operational properties. We need measures related to system requirements and design¾factors that requirements and designs must address (e.g., enforceable behaviors, measures of development defects).

NATIONAL CHALLENGES FOR HCS

Catastrophic events have not been sufficient to gain the national attention that is needed to foster high confidence research and development. For example, Intel's Pentium problem was costly and gained some attention, but addressing the problem remained relatively local to Intel and largely limited to better debugging that occurs late in the product life cycle. The catastrophic failure of the Arianne-5's first flight test was attributed to software, but received relatively little attention in the U.S.

We must articulate a set of challenges that will establish a national imperative for achieving high confidence in systems. The workshop participants discussed an approach that would:

- State the desired outcomes in meeting the challenge

- Develop a way to measure the outcomes with associated milestones
- Identify the technical breakthroughs necessary to meet the challenge
- Identify the strategies and barriers to meeting the challenge
- Identify what industry will do on its own and what government must do for the public good

In establishing this national imperative, we must keep in mind that we are seeing a growing dependence on computing throughout industries that have life-critical and safety-critical responsibilities. A new national sense of urgency must be brought to the issues of high confidence systems due to this increased use and rate of integration of computing and networking components in medicine, and in the transportation, financial, and energy industries. We want higher-confidence for existing applications while also being able to generate new capabilities such as pilotless aircraft and telesurgery.

Just as critical is the need to prevent the collapse of essential but saturated infrastructure systems such as air traffic control and an aging power grid designed for the early 1970s. A number of major systems that have been canceled—the FAA's Advanced Automation System, the military's World-Wide Military Command & Control System (WWMCCS) Automation Modernization Program (WAM), the IRS Tax Modernization System, for example. Such cancellations are an indication of our inability to deal with the complexity of next-generation replacements for aging infrastructure systems (see, for example, the book *Fatal Defect*[15].)

OUTCOME-BASED QUANTIFIABLE CHALLENGES

The workshop participants thought that HCS challenges should be stated in terms of desired outcomes and associated milestones. Examples of desired outcomes include higher transportation throughput with lower accident rates, or the Internet with higher numbers of transactions and lower risks.

A set of domain-specific national challenges could serve a purpose similar to that of the High Performance Computing national challenge problems and the goals established by NASA (see example 1, below). One such challenge for NIH could be to reduce the 28 percent medical error figure to 10 percent within five years. Another challenge for the Automated Highways Project could be to increase safety (by some factor) by getting the driver out of the loop while simultaneously increasing the traffic throughput (by another factor) on the nation's highways. Other challenges could include reducing fraud in financial systems by some factor; or increasing the ability to do electronic commerce over public networks by some factor. There is a spectrum of domains to address, from medicine to transportation, and from safety-critical to general quality-of-life, with ubiquitous computerized appliances. It is not necessary that we know how to achieve solutions to these challenges now. Rather, the challenges should help identify the technical breakthroughs that are necessary to achieve them. We will need a way to measure the outcomes.

[15] Ivar Peterson, *Fatal Defect: Chasing Killer Computer Bugs*.

The outcome-based articulation gives a picture of the world that is significantly different from that of today (e.g., pilotless commercial aviation). We need challenges that let us envision how to change the world. Like the HPC grand challenges, this articulation should appeal to the general public who will pay for this research. We must be wary of articulating challenges that generate fear (e.g., a pilotless airplane is a scary idea, as are self-driving cars).

We should also stress affordability challenges. Methods to achieve high confidence systems can potentially lead to less costly technology while improving their operational efficiency. For example, in the 1996 Information Technology Management Reform Act (ITMRA), such goals include a Sense of Congress provision that "Executive Agencies should achieve a 5% per year decrease in the cost incurred by the agency for operating and maintaining information technology, and a 5% per year increase in the efficiency of the agency operations."

We need to establish strategies for each articulated challenge. What are the barriers to achieving that challenge? What research is required? We will probably want a combination of focused research with relatively near-term results and longer-term, higher risk research with potential for high payoff.

Identifying nearer-term technical milestones makes progress on these challenges measurable. One example that participants discussed was to reduce software defects by a factor of *n* in *x* years—in particular, requirements defects by a factor of two in five years; specifications defects by a factor of five in ten years; and similarly for design, coding, and testing. Of course, this is not a simple set of objectives. Each one leads to many additional issues. For example, in reducing requirements defects by 50 percent:

- How should one count errors of omission?
- How are such errors detected?
- Can such errors be masked?
- How is completeness assessed?
- Are high confidence relevant requirements uniquely identifiable and separable from other system requirements?

Example 1: NASA's Aeronautics Enterprise. One example of a successful challenge-setting undertaking is NASA's Aeronautics Enterprise.[16] NASA's strategy is to exploit digital technology to increase aviation safety. NASA is developing an outcome-based research agenda across multiple industries and agencies (e.g., FAA and DoD), identifying industry partners to absorb the developed technology. The research agenda is intended to contribute to ten long-term, outcome-oriented challenges that NASA has established for the next twenty years in aeronautics. These are national challenges with a relevance that can capture the attention of the public, including:

[16] See the NASA Aeronautics Enterprise Web site for more details: http://www.hq.nasa.gov/office/pao/NASA/aeronautics.html.

- Reduce the cost of air travel by 25 percent in ten years and 50 percent in twenty years
- Reduce the aircraft accident rate by 80 percent in ten years and 90 percent in twenty years

All ten NASA challenges are designed to “stretch” technology. For some of the challenges, NASA does not know how they will actually be solved. But the point is to create a challenge that will pull the research along. For example, meeting some of the challenges involving dramatic reduction in coast-to-coast air travel times may require eliminating the pilot. This means that there must be a change in the way flight crews interact with the aircraft. This leads to new behavior requirements for computing and networking associated with commercial aviation.

Example 2: The NGI Initiative. The President's Next Generation Internet (NGI) initiative is intended to lead into "terabit-per-second network speeds over wide area advanced capability networks." There are three goals for the NGI: (1) Experimental Research for Advanced Network Technologies, (2) Next Generation Network Testbed, and (3) Revolutionary Applications. In the NGI implementation plan (IP) many performance metrics are defined within each of the three NGI Goals. Examples of these include:

- Demonstrate 25 percent utilization improvement in the 100 node, 100+ Mbps demonstration network (Goal 2.1) over 3 months and 100 percent utilization improvement in the 10 node, 1+ Gbps demonstration network (Goal 2.2) over three months.
- Demonstrate 15-msec response capability.

The above are not directly tied to applications. To plan the development of NGI applications, the NGI IP identified key "affinity groups." The NGI IP placed "disciplinary affinity groups" (e.g., health care, crisis management, basic science) in the rows of a matrix and within them looked for enabling "applications technology affinity groups" common across those rows (e.g., collaboration technologies, distributed computing, and privacy and security) that would support those disciplinary groups. It is through these affinity groups that additional applications measures will be developed. The disciplinary affinity groups would identify applications within the scope of that group that require NGI bandwidth and services; especially those applications that, while impossible without NGI, would improve mission success using NGI technologies. This leads to the following questions for HCS:

- What are the most appropriate ways to define and measure HCS contributions (technology specific, application specific)?
- How can we map HCS success down to more technical dimensions that can be measured? For example, could we have the top of a matrix have security, safety, reliability, and functional correctness?
- What are the significant HCS performance measures (challenges) we can demonstrate or target? Can we show a reduction in the cost of producing high confidence software

by 50 percent while increasing the safety and/or security by some factor? Could we develop product-based criteria that would reduce the cost per line of code or per delivered function for field-certified systems by half without any increase in safety related incidents?

ISSUES

There are many issues to address in meeting the challenge problems for high confidence systems. One set of issues deals with process, including tools, while another deals with the product. These two sets are tightly interrelated. A third set deals with measurement. Table 1 lists the issues identified by the participants.

Table 1. HCS Issues

I. Process Issues

- Getting the system requirements right
- System design. Systems engineering viewpoint: commonsense engineering principles for safety, for dependability, for security. Learning from past mistakes and incorporating this knowledge into reusable design principles.
  - Integration of different properties—the "ilities"
  - Understanding core principles and practices common across the different properties, and understanding the tradeoffs among properties
  - Composition: dealing with component technologies. If we are mixing and matching from a set of reusable components, how to get confidence in the parts, and how do we get confidence in the composition?
  - Evolution. How to maintain high confidence in extensible systems, user-modifiable systems, customizable systems, dynamically adaptable systems
  - Usability - treating the user as part of the system to reduce interface errors
- User-centered design
  - Coding standards and design practices
- Tools
  - Tools for considering the entire system
  - Tools to reduce/understand complexity.
  - Informal methods and lightweight tools
  - Tools that support different problem solving strategies
  - Tools tailorable to specific domain applications.
    - Seek a general purpose underpinning (e.g., decision procedures) common to all
    - Package in a way usable across multiple domains. Different domain-specfic front ends
    - Fundamental research in term rewriting, decision procedures. Need breakthrough research to address current brick walls, e.g., verification of programs that use integer arithmetic. Need decision procedures that are fast , efficient available in reusable form.
  - Testing tools and techniques
    - Specification-based testing. Develop test cases directly from specifications automatically. Find ways of replacing some of the current extremely expensive testing, structural tests. Reducing cost of testing. Alternatives to hardware testing (because of the high cost of unit test).
  - Modeling tools and techniques
- Reducing the time and cost of certification. Incremental evaluation cost for incremental change.

II. Product Issues:

- Lack of a conceptual framework, models
  - Establishing the boundaries of a system
- No unintended side effects
- Commonalities may reduce certification cost
- Improvements to new designs; establish well-known "safe" designs or architectures
- New designs for high confidence within the space of new system designs

· Introducing confidence into legacy systems
III. Measurement Issues:
· Assessing the effectiveness of processes and practices
· Metrics for degree of confidence in system
· National standards for certifying tools

HCS SOLUTION EXPLORATION - PANELIST VIEWS

On day two of the workshop the participants moved from problem exploration to solution exploration, suggesting and discussing the following solutions for the yet to be identified HCS challenges:

- Formal methods
- Improved use of programming languages, for example, proof-carrying code and type-checked compilation
- Software safety and improved specification tools
- Engineering process integration
- Lowering the cost of entry: Higher-confidence systems from low-confidence parts
- Research-supported education

SOFTWARE SAFETY AND IMPROVED SPECIFICATION TOOLS

One possibility for HCS development is to pursue systems development in a manner analogous to what Nancy Leveson, a professor at the Department of Computer Science and Engineering at the University of Washington, Seattle, describes in her book, *Safeware: System Safety and Computers*.[17] Using the concepts of software safety and continuing research along those lines would contribute towards developing improved specifications and an integrated methodology for high confidence.

Leveson has looked at many design-for-safety issues, e.g., experiments with *n*-version programming, experimentation with self-checks, comparative evaluation of alternative methods (fault elimination and fault tolerance), provably correct code generation, design of safe human-computer interfaces (HCI), design analysis, verification, and software fault tree analysis. She has evaluated these methods on real systems using prototype tools.

Leveson is using what she terms "software deviation analysis" to see what combination of events can lead to hazardous outputs. This approach enables a better understanding of what is needed for safety-related robustness. It is an adaptation of hazardous operations (HazOp)

[17] Nancy G. Leveson, *Safeware: System Safety and Computers*, Addison-Wesley, New York, 1995. See Leveson's Web site for an outline of this book: http://www.cs.washington.edu/research/projects/safety/www/book.html.

analysis from chemical plant applications. The approach considers software requirements completeness and safety analysis.

In addition, Leveson has developed RMSL, a requirements modeling and specification language. The language is essentially state charts embedded in stylized English. The goals of RMSL are to provide guidance in building specifications; completeness is enforced in the language. The language could also be model-checked for some criteria. Other goals of RMSL are the elimination of error-prone system features and improved specification readability. RMSL is being designed for a specific type of application, namely, process control models, so the language is domain specific.

The foundation of design-for-safety is the structuring of specifications based on human problem solving, cognitive psychology, and systems theory. Leveson wants to produce "human-centered specifications," i.e., specifications that incorporate a focus on the human in the loop of system control. She is attempting to add "intent" into the specification to amplify the "why" component of design rationale. This will be an integration of systems and software specifications including the HCI, procedures, and physical components. The specifications will provide for traceability to better enable system evolution and re-analysis for required changes. The language will also integrate with system hazard analyses, e.g., fault tree analysis.

RMSL was used to specify TCAS-II, the FAA's aircraft collision avoidance system. This work included the development of forty-seven criteria for completeness of specifications (see details in her book*)*. Matts Heimdahl developed a "D-completeness" (mathematical completeness) and consistency analysis for two of the forty-seven criteria. The approach enables backward and forward executions of specifications to identify events leading to hazardous states.

Leveson is continuing work on RMSL extensions including timing, integration of human procedure models, and mode-confusion software analysis (i.e., where the human is confused about which mode the system is in). For example, ongoing work includes analysis for restarting real-time systems. A number of accidents have been attributed to real-time software that has been taken off-line and then returned to service without synchronizing the software's internal state with the system's current state. RMSL now requires any restart to begin in an "unknown" state. Ongoing work also focuses on the integration of HCI and system operating procedures in RMSL to address the problem of operators not understanding the state of the system and, therefore, expecting different behavior. A number of accidents (including Airbus A320 problems) have been attributed to this problem.

Future RMSL work will include specification-based test generation to ensure completeness of test coverage, specification-based code generation and integration with autocode systems, further extensions to HCI designs and human-computer architectures, and work on organizational safety factors. An area of investigation is the application of safety techniques to security.

FORMAL METHODS (1)

Connie Heitmeyer asserted that continued work in formal methods was an essential solution to high confidence systems. Her work at the Naval Research Laboratory (NRL) continues earlier efforts in improving systems specifications. She is using static analysis techniques to identify unused inputs and spontaneous outputs in system specifications. She is also building tools to analyze and help cope with incomplete specifications.

Heitmeyer is extending David Parnas's software cost reduction (SCR) method of requirements analysis. She has developed a formal semantics to support this method and is building tool support for the more tedious, error-prone portions of specification analysis. Heitmeyer is focusing on automatic detection of errors, with executable specifications, and detailed feedback when an error occurs. She has a combination of lightweight and heavyweight tools, so that she can match the tool to the problem, using the simplest and lightest-weight tool possible.

The formal semantics are state based: the system is modeled as a state machine. These semantics also enable formal modeling of the system environment, much of which is naturally continuous. The project goal is to support reasoning about various classes of properties (security, safety, reliability, real time, etc.). Her tools include a specification editor, a dependency graph browser, a consistency checker, a simulator, and verification tools. There are also tools to automatically simplify the state machine so that it can be model checked.

Heitmeyer outlined where additional research is needed. One area is a sound foundation for model checking. The models currently used are almost always *ad hoc* and are a frequent source of errors. Needed is the capability to automatically derive the abstract models to be analyzed, and their derivation should be based on the property one is trying to prove. Then one would have a mapping from the state space of the abstract machine to the state space of the original machine. The reasoning about the abstract machine should be equivalent to the reasoning about the original machine, so that counterexamples in the abstract machine correspond to counterexamples in the original machine. This specification set should be the same set that the system engineers are working from, rather than a separate (disconnected) parallel formal process. Theorem proving will be required to show completeness and soundness.

A second research issue is automatic invariant generation. Needed are sound algorithms that generate invariants from specifications. Such invariants are extremely useful in reasoning about specifications; they are useful as auxiliary invariants in proving more general system invariants, and useful in themselves as system properties. Examples are Zohar Manna's localized invariants, and the mode invariants of SCR specifications.

Another research area is the formal representation of the system/software environment. This is usually ignored by model checkers. The environment should be modeled as nondeterministic and continuous.

There is also a need for applied research, for example, better engineered model checkers. Counterexamples can be hundreds of states long and difficult to analyze, often cryptic to anyone not intimately familiar with the model. And it is difficult to map this back to the original specification. Errors are made in going from the concrete machine to the abstract model. We should simulate the model where model checking is too computationally expensive. Better engineered model checkers will provide automatic translation of specifications into the language of the model checker as well as easily understood feedback when an error is detected (for example, run the produced trace through a simulator).

Another research area is more efficient, well-integrated decision procedures. Specifications of high confidence systems usually include variables of varied types. The formulas analyzed in such specifications can include propositional tautologies, Presburger arithmetic, linear arithmetic, etc. We will want to use all these at once.

We need comparisons of different analysis techniques; comparing model checkers, for example. But we must be cautious about comparing apples and oranges because some model checkers are better for some problems than for others. Such comparisons would allow us to give guidance to system developers. (e.g., some are good for protocols, some for hardware).

Trustworthy code is a requirement of high confidence systems. We need specialized models and tools for specialized domains. We should generate code from specifications and produce automatic test cases. We need to develop theorem proving systems for specialized domains. Once the user develops the overall proof strategy, the system could suggest needed lemmas and complete the trivial proof parts automatically.

FORMAL METHODS (2)

Ricky Butler discussed NASA's concern with identifying missing areas of HCS research. NASA uses formal methods to reduce errors in aviation systems. However, a mission-oriented agency such as NASA has difficulty getting basic research funding for fundamental computer science investments. Instead, NASA funded several projects to pair a formal methods team and industry partner to transition the technology. NASA has also leveraged DARPA investments as well as those by other parts of the Government. This same focus on applied research is seen in ohter Government agencies—thus running the risk of neglecting the fundamental research essential to achieving high confidence systems.

Fundamental research is needed in theorem proving technology (e.g., automated reasoning). NASA wants DARPA to fund this research. There is a spectrum of needs for verification tools, ranging from domain specific to general purpose, and from mathematician friendly to engineer friendly. NASA wants improved automated reasoning, with algorithms underneath (transparent to the user) that decide chunks of mathematics for the user. There are dozens of these formal methods tools and they all exploit the fundamental techniques developed more than a decade ago: decision procedures, term rewriting, model checking, resolution. We have no research programs aimed at advancing this technology. What challenge problems could, as a side effect, lead to funding of some of this fundamental research?

One challenge problem may be to produce a next generation verification environment providing a seamless integration of different tools and strategies that are effective for different parts of the life cycle. Such an environment could support a wide variety of applications, result in a hundred-fold improvement in user productivity, provide interoperable libraries for fundamental concepts of computer science and discrete mathematics, and enable effective modeling techniques.

Recent research funded by NASA has focused on requirements analysis tools and requirements specifications. This is probably because we have been trying to get industry to use these things. This is a good approach (i.e., finding errors), but we have completely neglected the refinement of requirements into executable code. If we want safe systems, then we need design verification, code verification, and certifiable and qualifiable program synthesis. We want to be able to assert that there are no safety-related errors in the system. We do not want to use formal methods just to find bugs, but to give us confidence that the product is safe. Existing code synthesis has not been designed with certifiability in mind.

We need new ideas on how to advance automated deduction. We need research to identify parts of arithmetic we can reason about (e.g., a theory of bit vectors), and we need decision procedures for modular arithmetic. We could use more tools to exploit graph theory and other aspects of mathematics. We also need to identify new relevant domains and the limitations in the current formal methods tools.

There will continue to be a trade-off between funding new fundamental research and transitioning newly developed technology into engineering practice. The concern is how scarce resources should be best focused. One barrier to use of current formal methods technology is the sheer horsepower needed to do the analysis. However, the situation has greatly improved in the past several years. Rather than looking at the past attempts and costs of formal methods application, we need a significant, visible demonstration of what can be done today. At the same time, we must be cautious in our claims. We need to calibrate some of the differences between what was and what is now. For example, the time to produce a mathematics book was about five years in 1960; today we could use SRI International's PVS verification system to produce that same book within a few months. Can we further quantify the changes in how long it takes to do a particular verification? There could be a quantum improvement demonstrated simply because computers have gotten faster and more capable by a larger factor.

FORMAL METHODS (3)

John Rushby of SRI International asserted several research challenges for formal methods. These extend from refutation (debugging) to verification (assurance). We need to abstract rather than to downscale, as Heitmeyer said, and the abstractions must be mechanized. In the past, researchers didn't have the horsepower to do this, but with today's tools they can do it. We need a repertoire of mechanized deduction methods, not just model checking, to construct, justify, and use these abstractions.

Today's model checking's simplification of problems is too crude, the simplified system models used as input are arbitrarily downscaled, and we can't provide assurance that nothing is wrong with the real system as opposed to the system model that may, for example, have fewer and shorter buffers than the real system. We need justifiable simplifications/abstractions. Heitmeyer's abstraction methods are good, and we need still other abstraction techniques. This will make it possible to address large problems. Much of the work in automated deduction in the near future will be in using computation to simplify the theorems that must be proven.

Rushby suggested composing locally checked designs and properties. For example, one might model check small fragments and glue these together in some way. A combination of model checking and theorem proving is possible, but it raises issues such as language issues. Formal methods research that is being applied to hardware now must focus on systems issues that are at least seven years out from now. Requiring verification systems to hook up to the Very High Speed Integrated Circuit (VHSIC) Hardware Design Language (VHDL), for example, shackles us to the transient present. Rushby believes that Intel has a better vision for this research than the research funding agencies.

Not only do we need to develop suitable methods, we need to integrate algorithmic and interactive methods. We need to integrate formal methods with testing and safety analysis techniques such as fault tree analysis and reachability analysis.

One goal for formal methods is to be able to provide a calculation of properties of models. This should be done as part of the engineering process in a fashion similar to computational fluid dynamics except the calculation is done by mechanized deduction (model checking, decision procedures, interactive theorem proving).

The ability for model checking to find bugs in complex systems has been adequately demonstrated and has led to widespread acceptance in certain fields including hardware and protocols. It has successfully helped hardware engineers avert many bugs in processor and cache designs. Model checking is being exploited to find problems in protocols (communication, authentication), distributed algorithms, chemical and nuclear engineering control systems, and fault-management in spacecraft. Its application to additional problem domains should be pursued.

We have witnessed a tremendous increase in capabilities over the last ten years, and we can foresee tremendous opportunities in next ten. The kind of systems people are building is changing rapidly and drastically in a way that is inimical to providing any guarantees, so we could provide major enablers for what people want to do. We need to use formal methods to dispense with what is trivially computable, and allow the human to focus attention on discovering other problems. Rushby believes that we must focus research more on paradigmatic examples than on big industrial problems; it is this approach that will help avoid our overselling formal methods.

LOWERING THE COST OF ENTRY: HIGHER-CONFIDENCE SYSTEMS FROM LOW-CONFIDENCE PARTS

Ed Felton from Princeton University asserted that we need to lower the cost of entry in achieving higher confidence systems. He believes there are several key ingredients to making this happen: education, technology transfer, intuitive means of increasing confidence, and empirically based “economic coping.”

The biggest impediment to building system with high confidence is that people in the field (i.e., programmers) do not see the usefulness of thinking carefully and systematically about the code they write. We need to address this in education early on by teaching respect for addressing a coding task more formally and by using these methods in a way that meshes with the rest of the curriculum. More disciplined and formal thinking should pervade the curriculum: if it is in a separate course, students will think it is a separate thing from system building. This education is necessary if we are to achieve high confidence programming. Today, most students do not know that their programs are not correct—they do not try them on enough test cases. If, however, they are told that they will get a zero for the assignment unless the program gives reasonable error messages on a number of test cases, they may try harder.

Universities and industry have to join in partnership to transfer university developed technology (i.e., the research prototypes) into usable industry technology. We must also transfer the research prototype to commercial systems developers where the cost per individual failure is low, but there are many copies of the product (e.g., hardware failures). We then would be able to amortize the total cost across all these copies; the collective cost may actually be quite high. The key to this solution is forging the appropriate partnerships between the tool developers and the problem holders. As part of technology transfer, researchers have an obligation to learn about the pressures underlying product development.

The methods we are developing should not have a huge learning curve, and it should not require a lot of time to build specialized models. We should maximize the use of "semi-formal" and informal methods, and have a focused use of formal methods on the most critical or difficult part of the product. We must consider human factors for users for developers. Such human factors can be applied to notations and tools (e.g., to make it easier for a developer to translate a specification into code). Human factors can also be applied to administrator interfaces. Most administration interfaces allow users to do bad things without any warning.

We must seek compromise, not victory. For example, if you use this methodology, you can get a certain kind of help. Understanding the utility of compromise is important; even in critical systems it is often the simple components that people are getting wrong; they need better type checkers, for example. Give them the simple components now and you are better situated to give them more later.

We must learn from the mistakes of others. How else are we to know, in advance, where to focus attention for the application of formal methods? We must understand where systems break down. This may require looking across many cases and developing meaningful statistics. Since today's systems are low-confidence systems, there should be enough cases to generate statistics. For example, we might not have known in advance that buffer overflows would turn out to be such a common security problem, but after we see so many examples of this, we should know to focus attention on buffers.

It is becoming more likely that many of our critical systems will be deployed with Windows. We will feel its effects (e.g., new software may make the job take longer, or when the computer is down the lines will get longer at the airport). It is not commonly understood how weak these commodity products are. Products used in critical systems must be simple enough so that we can analyze their failure modes.

Can you build high confidence systems from low-confidence parts? We can learn from practices developed for trusted systems. A lot of the functionality is probably not safety oriented, so we should concentrate our efforts on the areas that are safety critical and protect those areas. For example, only put Windows where it is not safety critical.

The increasing prevalence of virtual machines may make high confidence easier. If the Java Virtual Machine (JVM) really does run on every platform, it could help in two ways: (1) increasing the uniformity of the execution environment would allow people to better focus their efforts on creating a high confidence execution environment; and (2) if it has higher assurance than what people were using before, it is an immediate win.

RESEARCH-SUPPORTED EDUCATION

Today's systems are highly complex, especially embedded control systems for ships, aircraft, automobiles, factories, weapons, nuclear power plants, and medical devices. They have a lot of software, legacy components, and commercial hardware and software, all pulled together by a domain-defined fabric. These systems are often distributed, highly interconnected, highly integrated, and operationally time-constrained. They have a significant cost of failure, and such high costs mandate the need to achieve high confidence in their design, construction, maintenance, and operation.

Systems fail because of operational wear-and-tear, operator error, and human-induced defects introduced during system design and development. Failure results from incomplete definition of requirements, defects in software or hardware, operational error (e.g., lengthy, complicated, broken procedures), security violations, implementation defects, and human-computer interface designs that lead to operator confusion. They may also fail due to their inherent complexity. There are many interesting recent failures such as the Internet's Domain Name Service (DNS) failure due, in part, to the distribution of bad tables; the America On-Line (AOL) upgrade error; and AT&T's WORLDNET net-wide propagation of a single error in a C program which caused failure of AT&T's long distance service due to a register overflow that, in turn, was compounded by having the backup database overwritten by a corrupted database during recovery.

John Knight of the University of Virginia asserted that research-supported education has to be one of the key solution directions. We need education for practitioners that teaches them existing techniques and we also need new techniques for practitioners. The computer science research community has not reached out to engineering. Obtaining new techniques for engineering practitioners requires that we educate researchers in the context of practical issues and practical constraints. Such education requires a multi-disciplinary, coordinated research program.

Bad technology continues to be used by practitioners. We must inform practitioners about the good technology that exists but is not widely used, such as "semi-formal" specifications, strongly typed languages, test tools and techniques, static analysis and inspection, and rigorous and analyzable design techniques. Predicate calculus should be taught and understood by any engineer. Promising new technology needs to be tried more widely, such as formal specification, application generators, component-based reuse, test case synthesis, etc. But some of these things are not quite ready for prime time. We can learn how to make them more robust from experimenting with their use.

Researchers must be educated so that they understand what they need to work on. They must understand the real problems and issues, the environment, and the priorities. If researchers do not understand these things, they will be working on the wrong problems. People do not jump into new technologies rapidly; researchers need to understand the concepts and tools that the engineers use to accomplish their work. Researchers need to give the engineers specification constructs and tools that the engineers can understand. The concepts and tools that are produced and used should focus a practitioner's energy where it should be focused. For example, engineers prefer to use switching diagrams, LaPlace diagrams, etc., because they are skilled in their use and the notation allows them to focus attention on the problem rather than on the notation. This then leads to the requirement to better integrate tools across the disciplines, and more research is needed to accomplish such integration.

Another participant pointed out that we vitally need practitioner education to reduce over-simplified ad hoc solutions in fielded critical systems, such as *n* version, unsynchronized fault tolerance. Design problems (e.g., meeting both security and reliability needs) are fairly sophisticated, and ad hoc solutions are being used on life-critical systems such as the Boeing-777 channel control systems and train control systems, developed by engineers lacking necessary knowledge about advances in high confidence systems techniques. Almost no articulated knowledge exists in these fields that is written down; most knowledge is passed on verbally within an organization. Such verbal lore is highly localized. We must bring such knowledge to higher levels of visibility.

ENGINEERING PROCESS INTEGRATION

Shiu-Kai Chin from Syracuse University proposed that solutions will come from reducing theory to practice for high confidence design. Chin is working on an educational program whose goal is to have a lasting impact on our high confidence systems through change and focused improvements to our systems engineering processes. He believes this lasting impact will result from teaching engineering students abstract conceptualization and applying it to small examples in the classroom. Chin noted that the 1995 CIC report has been a strong influence on the directions of his program at Syracuse University. He is developing and teaching masters-level courses in engineering, and his students are using formal techniques for high confidence system design. He wants to move formal methods for high confidence into undergraduate education. He believes that predicate calculus is appropriate for undergraduate engineering students and that they can absorb it if there are sufficient examples for them to work through.

Chin is engaging high confidence systems engineering along two dimensions: (1) developing support for reasoning, evaluation, and conformance; and (2) developing a science of composability and its application to engineering. What he hopes to achieve is an enhanced, multi-disciplinary design process that uses several formal methods and tools in an integrated fashion.

In developing his integrated approach, Chin asks several key questions:

- What are the design (and analytical) processes that support high confidence?
- What is the theoretical basis for the design process? (We don't teach engineering by just throwing tools at the students.)
- How is the process applied to produce implementations?

Chin thinks that if the more formal process can be articulated (with experimental justification), the engineer can understand the benefits of using the theory and will try it. An example of this, inserted by Ricky Butler, is that Rockwell engineers were able to read PVS specifications within a month or two.

System engineering requires integration of many design methods, tools, and techniques; connections should be encouraged among them. Chin referred to David Kolb's 1984 experiential learning cycle (characterized as "concrete experience -> reflective observation thinks -> abstract conceptualization -> active experimentation") and noted that our research processes are out of synch with our software development processes. Industrial engineering has generally been on only two parts of the cycle (active experimentation and concrete experience), while academic researchers have been in the other two parts of the cycle. We must connect the two loops¾feeding problems to the research community and solutions back to the development community.

Chin wants to show that formal methods and tools are of practical value in an enhanced design process. His approach is to provide one or more concrete challenge problems (similar to the DARPA/NSA/DISA Joint Technology multi-level security (MLS) formal methods challenge problem) to demonstrate the utility of formal methods. A challenge problem must be a real system with properties to be established.

Chin intends to incorporate into his enhanced design process the support necessary for assurance and synthesis through formal verification and compilation. He is interested in showing the potential for starting from a top-level requirement and achieving a synthesized set of code. He is including higher order logic and functions to support parameterized design and use of hardware and software components. Compilation from higher order logic theories and algebraic specifications is used to produce C++ code. He separates the data path and control path operations for analysis. This separation sets up the verification path and the synthesis path. It is easy to show the relationship of these paths (e.g., using Kestrel's *Specware* category theory-based tool). There are appropriate points of correspondence that can be used to join them. Chin has used *Specware* to generate code for a secure e-mail system.

Unifying various logics could mean a potential wait of twenty years. But there are more practical approaches. In a year, Chin established a correspondence mapping between synthesis (refinement in *Specware*) and verification (type definition in HOL (high order language)).

Chin mentioned the 21st Century Engineering Consortium, which will focus on education and how to move formal methods for high assurance design into mainstream engineering. An initial two-day workshop is being held in Melbourne, Florida, in March 1998. The workshop is intended to gather insights from industry and government, identify what skills are needed, and to examine current practices in education. Two questions are to be answered: What are we teaching and why? How well are we doing? The answers are intended to be the source of recommendations for both graduate and undergraduate education.

IMPROVED USE OF PROGRAMMING LANGUAGES

Peter Lee from Carnegie Mellon University brought two related examples of potential solutions to the table from his research focus on improving our use of programming languages to achieve higher confidence systems. Lee asserted that the science of programming languages is a bridge between research and practice. We should have no distinction between programming and specification languages. Such languages provide the structure for communicating, either to another human or to a computer or both. Programming languages also give you a way of mediating the boundary between formal and informal methods. Programming languages provide an essential foundation for high confidence computing.

Lee illustrated the contribution of programming languages to high confidence using the analogy of maze puzzles. A good maze must have a solution; and both the maze producer and the user want confidence in that condition. Now, suppose a maze extends infinitely in two dimensions. It is extremely difficult to know if such a maze has a solution. But if a maze producer wants to convince a maze "consumer" that the maze is a good maze, he can provide the solution path along with the maze, and the consumer can simply check it. It is much easier to check a solution than to generate it. The consumer does not care how the line trace was generated or derived.

Lee presented a similar approach for high confidence software. At Carnegie Mellon, Lee is directing the Fox project, that is conducting research in the composition of software. The programming language solution to composition is to use interfaces, that is, one can check whether two interfaces can match up. But the problem is we can not know if an imported software component (say an applet) will do bad things to another component. So the solution is to require that imported code come with some "witness" to the fact that it follows our rules. This witness approach Lee calls Proof-Carrying Code (PCC). PCC is a technique that packages the proof of a software module's properties along with the module's executable code. This makes it possible to check the module against an application's specification before including the module. PCC has several advantages:

- No external authentication or cryptography is used.
- The code consumer does not care who does the certification or how.
- Excessive runtime checking is avoided.

- The method is tamper-detectable.
- The Trusted Computing Base (TCB) is simple, small, and fast. The core proof-checker is about five pages of C code. The same checker is used for many applications. Simple logic has been used, but in principle an implementer could use more expressive logic to obtain a more precise specification of safety policies.

PCC is not without problems. It is difficult to prove arithmetic. The need for arithmetic decision procedures would make the code much larger. This is a difficult issue for this approach.

Checking imported modules against application-specified pre-conditions provides higher confidence of correct operation. Lee, along with George Necula, has been applying this to the context of extensible operating systems. They are implementing PCC in a Unix kernel. Their formal safety policy is a local (kernel) definition of what should happen or should not happen. But they still have the problem of how to know if they have formally stated what they wanted to state or what they should have stated.

They are working on several program proofs in the areas of high-speed networking and extensible operating systems. These include bounds checking; memory safety; time limits (actually instruction count limits); and resource constraints such as guaranteed memory deallocation, locks, and network I/O (guarantee write constraints of no more than *n* packets per second). They are also working on a proof that target code produced by an optimizing compiler preserves the type safety properties of the source code (for Java) and a proof that a machine language program meets the type safety properties of machine language (ML). Example proofs were checked five to fifteen times faster than software (pretty good privacy—PGP) encryption.

Lee stressed the role of types and type theory in providing high confidence. An example from his research shows that in using modern programming languages (e.g., ML) a safety predicate can be encoded as a type. With the proof encoded as an expression of that type, proofchecking is reduced to typechecking. Type checking then becomes a means to guarantee the (type) safety property. In his work, the proofs are represented in a type system (line feed (LF)) because LF typechecking is decidable. Standard techniques from type theory are used to prove the soundness of all of this and to obtain better algorithms for fast checking of proofs and more compact encodings. And because it is a programming language, there is a bridge from the formal analysis to the actual implementation.

To date, these proofs have been generated by hand, but it is hard work. Lee and Necula want to do proof generation. Several research questions arise: Can we have a compiler generate these proofs, and can we avoid having to do theorem proving on the compiler? Can we preserve properties all the way down to machine code? Can we prove that the compiler has not fouled something up? Experience has shown that simple properties (memory safety, instruction counts, etc.) for simple programs (e.g., packet filters) seem to be no problem, but scaling to more interesting properties remains an issue.

Peter Lee thinks that we can do type-directed compilation. His approach of propagating types and staged type checking may provide an answer. Carrying type information across intermediate levels of translation and checking it at each pass of compilation is a possible approach to certifying compilers. This could serve as assurance for the correctness of the compiler without formal theorem proving; it can propagate easily checked safety information all the way to native code.

Lee thinks that programming language theory should be part of any high confidence program. While providing potential solutions, much remains for further research. These include advanced programming languages, type theory, formal semantics, and applications of logic. Basic research on foundations is needed. Higher assurance in operating system kernels has been achieved through the application of type theory (LF type theory in the Open Group's OSF/1 kernel). Research on the application of programming language theory is spreading. It is being applied to networks and operating systems as well as to security, real-time computing, and distributed computing. Programming language theory may be entering a golden age: theorists are becoming aware of problems in related areas of computer science and it is sparking a lot of research.

However, we have also seen a heavy shift away from basic research. The U.S. is still behind Europe in this theoretical area. We have a serious shortage of fresh talent. And there is the growing perception by researchers (possibly real) of no interest from Government organizations and funding agencies.

HCS SOLUTION EXPLORATION - DISCUSSION

ESTABLISH THE COMPELLING HCS NEEDS

To obtain new funding for HCS research, we must establish that the HCS needs are compelling and that clear achievable goals exist. We must convey the urgency generated by the trend toward ubiquitous computing, the increasing level and accelerating rate of integration, and the resulting increased personal risk exposures for every citizen. More important will be addressing such risks in transportation, health care, and critical national infrastructures.

The High Performance Computing and Communications (HPCC) Program argued that research investments lead to economic advantage. This approach should be considered here as well. The goal is to be able to build good systems well. *Building good systems* means building in properties such as reliability, and *building them well* means achieving the result with the most effective and efficient tools. Thus, in addition to high quality, we should articulate the added benefits of improved productivity. The result is better and more affordable systems. Establishing this HCS research agenda today will prepare us for the day when industry turns to computer scientists for help with large-scale, highly complex projects, especially for those that are life critical.

Without such a research program, we could be blindsided by another country that invests in this area to build a capability for higher quality software. The U.S. must develop and maintain a reputation for high quality systems to maintain the health of the our computer industry. Such a research program can also enable the “rocket science”—e.g., the equivalent of putting people on the moon—that our Government and industry needs. Finally, we want to ensure that life-critical and other safety-critical systems are built with the requisite degree of confidence.

BUILDING A BETTER BASE

Efficiency in the use of scarce research dollars points to the need to integrate or interoperate our tools for high confidence, for example, libraries of components of mathematical software. This will not be easy. In the experience of one researcher, two years of research in logic were needed to figure out what the common theory was to put two systems together. Different theorem provers are based on different logic. Each theorem prover has its own procedures, separately coded and separately verified. Integrating these systems will require these provers to talk to each other in a sound way—it is not simply a syntax problem. We need common sets of tactics and tactics-style theorem provers¾essentially a theorem prover’s workbench. We should be able to share these components, and research is required on how to do this well. To start with, we need to share decision procedures, mathematical software libraries, models, and provers; and it should be possible to connect them. Such sharing will result in monetary savings (i.e., it will provide the leverage to achieve savings), but it will also require near-term increases in costs to build the necessary infrastructure to enable sharing. We could follow the model of the National HPCC Software Exchange that has been funded by NASA and other CIC agencies at more than $1 million per year. This exchange does not currently support research, however.

Sound results require a solid science base. Many people take for granted the tools that were built up over years of hard work (e.g., higher order logic, constructive theorem provers). But there has not been nearly enough investment to support today’s needs. Meeting our goals also demands significant improvements in systems engineering. Efficient tools and processes are needed for increased understandability of complex systems. Improved integration of formal methods as well as lighter-weight methods into the systems engineering process and improved efficiency of these tools are needed. We need tools that can be widely used on the World Wide Web. For example, we need to integrate with tools such as Java. We should exploit Web mechanisms to make provers available and to provide lighter-weight tools.

Achieving these goals will require stronger partnerships between the researcher and the engineers who build systems. We must find out what they want and need. The experience of at least one academic research organization has shown that in working closer with industry through partnering, it discovered things that developers really want that the researcher had not previously realized.

CLEARER ARTICULATION OF NEEDS AND OPPORTUNITY POTENTIAL

Why should we begin a new research initiative now? Systems are getting larger, more complex, interactive, integrated, and they fail from the sheer complexity of their own interactions. "Normal accidents"[18] resulting from interactive complexity will become more prevalent in the future. We need to instill a sense of urgency in industry, government, and the general public if we are to head off the potential for increasingly catastrophic results of such accidents.

We are reaching the limit of what we know how to do safely. System design considerations used to be separable, but we can no longer build systems one piece at a time and do local analysis. It is increasingly difficult to predict overall behavior. Performance requirements mandate more integrated systems. An example of such integration and its resultant problems is the DARKSTAR stealth product at Boeing that is designed without a tail (rudder) surface. It was not possible to design flight controls along one axis at a time, because everything interacts with everything else. Engineers concentrated hard on the yaw axis, but much less on the pitch axis; as a result, the DARKSTAR crashed on takeoff on its second flight.

Another reason for doing this now is that today's higher-powered computing infrastructure—as well as significant successes in applying formal techniques, most notably model checking—allow us to do much more with our tools. For example, through model checking, hundreds of bugs were found in the Intel P7 processor. Engineering organizations are beginning to see the value of formal methods. Intel has made a commitment to verify all of its P8 processors and is already absorbing much of the world's supply of formal methods graduates (about 60 per year).

The U.S. is falling behind in the area of high confidence systems, both in research and in industrial products. We need cheaper building blocks for high confidence systems that can be inexpensive and commercially viable. The SAFEBUS of the Boeing-777 costs about $10 million. The Europeans intend to develop products that cost from $10 to as low as 50 cents per item. For example, there are new HCS opportunities that have been largely ignored in the U.S. such as Time-Triggered Architectures[19] (TTAs). TTAs provide a much simpler and cheaper means to achieving control than current expensive control system products. They are well designed, very simple versions of more expensive products, and they have been thoroughly analyzed. Because of their low cost and higher reliability, they can out-compete most of the U.S. control system suppliers. Driving down costs will allow us to move into new application areas.

In summary, a new national research initiative in High Confidence Systems should be started. It should be driven by several high-value societal challenges. It is urgent that the initiative be started now, because technology is becoming increasingly dependent on highly complex

[18] Charles Perrow, *Normal Accidents*, 1984: A "normal accident" is one in which interactive complexity and tight coupling of system characteristics will inevitably produce an accident. "It is meant to signal that, given the system characteristics, multiple and unexpected interactions of failures are inevitable. This is an expression of an integral characteristic of the system, not a statement of frequency."

[19] Herman Kopetz, "Real-Time Systems, Design Principles for Distributed Embedded Applications," Chapter 8, The Time-Triggered Protocols. Kluwer Academic Publishers, 1997.

systems for our safety and national well-being. Moreover, the U.S. is falling behind other countries in technology investments in this area, which could contribute to an eventual decline in U.S. leadership in the computing industry.

APPENDIX A. AGENDA

Wednesday, August 6, 1997

0830	Purpose of the workshop - Introductions	Teresa Lunt (DARPA-ITO)
0845	Review of recommendations for Computing, Information, and Communications (CCIC) R&D Forum for High confidence Computing contained in *America in the Age of Information: A Forum*.	Mani Chandi (CalTech)
0930	Panel I - "Those who possess the HCS problem"	Panelists: Jeff Van Baalen (NASA-Ames); Mike Dewalt (FAA); Nancy Leveson, (U.WA)
1030	Break	
1045	Panel II - "Those who possess the HCS problem"	Panelists: John Murray (FDA); Leo Beltracchi (NRC); Henry Heffernan (NIH)
1230	Working Lunch	
1330	Problem Exploration - Focus on Challenges, Issues, and Priorities	Co-Moderators: Teresa Lunt, Nancy Leveson
1530	Break	
1545	Continue Problem Exploration	
1730	Adjourn Working Session	

Thursday, August 7, 1997

0830	Reconvene - Summary of Day 1	Teresa Lunt
0845	Panel III - "Those who possess solutions"	Panelists: Nancy Leveson, Connie Hietmeyer (NRL); Ed Felton (Princeton); Ricky Butler (NASA); Barry Johnson (U.VA); John Rushby (SRI)
1045	Break	
1100	Panel IV - "Those who possess solutions"	Panelists: John Knight (U.VA); Bob Constable (Cornell); Shiu-Kai Chin (Syracuse); Peter Lee (CMU)
1230	Working Lunch	
1330	Solution Exploration - Focus on Goals, Possibilities, and Approaches	Co-Moderators: Terry Mayfield (IDA); Mani Chandi
1530	Break	
1545	Continue Solution Exploration	
1630	Summary Discussions	
1700	Adjourn	

APPENDIX B. ATTENDEES

Bob Aiken
DoE
19101 Germantown Road
Germantown, MD 208740-1290
ATTN: ER-31
Tel: 301-903-9960
Fax: 301-903-7774
E-mail: aiken@er.doe.gov

Mark Balkovich
National Research Council, NAS
2101 Constitution Ave., NW
Washington, DC 20418
Tel: 202-334-2558
Fax: 202-334-2318
E-mail: mbalkovi@nas.edu

Leo Beltracchi
NRC
12112 Triple Crown Road
Gaithersburg, MD 20878
Tel: 301-415-6558
E-mail: lxb@nrc.gov

Robert Brill
NRC
Office of Research
M/S T10-E33
Rockville, MD 20852
Tel: 301-415-6760
Fax: 301-415-5160
E-mail: RWB2@NRC.GOV

Wayne Bryant
NASA
NASA Langley Research Center
Hampton, VA 23681-0001
Tel: 757-864-1690
E-mail: w.h.bryant@larc.nasa.gov

Ricky Butler
NASA-LARC
Mail stop 130
1 South Wright St.
NASA Langley Research Center
Hampton, VA 23681-0001
Tel: 757-864-6198
Fax: 757-864-4234
E-mail: r.w.butler@larc.nasa.gov

Mani Chandi
California Institute of Technology
Computer Science Department
Mail Code 256-80
Pasadena, CA 91125
Tel: 818-395-6559
Fax: 818-792-4257
E-mail: mani@cs.caltech.edu

Shiu-Kai Chin
Syracuse Univ., Associate Professor
Electrical Eng. & Computer Science
123 Link Hall
Syracuse, NY 13244
Tel: 315-443-3776
Fax: 315-443-2583
E-mail: skchin@syr.edu
or
Rome Laboratory
525 Brooks Road
Rome, NY 13441
Tel: 315-330-3241
E-mail: chin@cat.syr.edu

Bob Constable
Computer Science Department
4147 Upson Hall
Cornell University
Ithaca, NY 14853
Tel: 607-255-9204
Fax: 607-255-4428
E-mail: rc@cs.cornell.edu

Mike DeWalt
FAA
FAA ANM-106N
1601 Lind Ave. SW
Renton, WA 98055
Tel: 425-227-2762
Fax: 425-227-1181
E-mail: michael.dewalt@faa.dot.gov

Tice DeYoung
NASA
4312 Alta Vista Drive
Fairfax, VA 22030
Tel: 202-358-1363
Fax: 202-358-3519
E-mail: tdeyoung@mail.arc.nasa.gov

Ed Felten
Department of Computer Science
Princeton University
35 Olden Street
Princeton, NJ 08544-2087
Tel: 609-258-5906
Fax: 609-258-1771
E-mail: felten@cs.princeton.edu

Helen Gill
DARPA/ITO
3701 N. Fairfax Dr.
Arlington, VA 22203-1714
Tel: 703-696-7461
Fax: 703-696-2202
Fax: 703-696-6416
E-mail: hgill@darpa.mil

Henry Heffernan
NIH
19 Eye St., N.W.
Washington, DC 20001
Tel: 202-336-7208
Fax: 202-789-1880
E-mail: henry_heffernan@nih.gov

Connie Heitmeyer
Naval Research Laboratory
Code 5546
4555 Overlook Ave., SW
Washington, DC 20375-5337
Tel: 202-767-3596
Fax: 202-404-7942
E-mail: heitmeye@itd.nrl.navy.mil

Barry Johnson
Department of Electrical Engineering
Thornton Hall
Univesity of Virginia
Charlottesville, VA 22903-2442
Tel: 804-924-7623
Fax: 804-924-8818
E-mail: bwj@virginia.edu

Dick Kieburtz
Division of Computer and Computation Research
National Science Foundation
4201 Wilson Blvd., Room 1145
Arlington, VA 22230
Tel: 703-306-1910
Fax: 703-306-1947
E-mail: rkieburt@nsf.gov

John Knight
Department of Computer Science
Thornton Hall
University of Virginia
Charlottesville, VA 22903
Tel: 804-982-2216
Fax: 804-982-2214
E-mail: knight@virginia.edu

Gary Koob
DARPA/ITO
3701 N. Fairfax Dr.
Arlington, VA 22203-1714
Tel: 703-696-7463
Fax: 703-696-2202
Fax: 703-696-6416
E-mail: gkoob@darpa.mil

Rick Kuhn
NIST
North Building, Room 509
Gaithersburg, MD 20899
Tel: 301-975-3290
Fax: 301-926-3696
E-mail: kuhn@sst.ncsl.nist.gov

Peter Lee
Department of Computer Science
Carnegie-Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
E-mail: petel@cs.cmu.edu

Nancy Leveson, Boeing Professor
Computer Science & Engineering Dept.
Box 352350
Univ. of Washington
Seattle, WA 98195
Tel: 206-685-1934
Fax: 206-543-2969
E-mail: leveson@cs.washington.edu

Jeffrey Van Baalen
NASA-AMES
Computer Science Department
P.O. Box 3682
University of Wyoming
Laramie, WY 82071
Tel: 307-766-6231
Fax: 307-766-4036
E-mail: jvb@ptolemy.arc.nasa.gov

Teresa Lunt
DARPA/ITO
3701 N. Fairfax Dr.
Arlington, VA 22203-1714
Tel: 703-696-4469
Fax: 703-696-2202
Fax: 703-696-6416
E-mail: tlunt@darpa.mil

Terry Mayfield
IDA/CSED
1801 N.Beauregard St.
Alexandria, VA 22311
Tel: 703-845-6602
Fax: 703-845-6848
E-mail: mayfield@ida.org

Reg Meeson
IDA/CSED
1801 N.Beauregard St.
Alexandria, VA 22311
Tel: 703-845-6619
Fax: 703-845-6848
E-mail: rmeeson@ida.org

Robert Meushaw
NSA/R2
9800 Savage Road
Ft. Meade, MD 20755-6000
Tel: 301-688-0840
E-mail: rvmeush@clark.net
E-mail: rvm@tycho.ncsc.mil

John F. Murray, Jr.
FDA
Center for Devices and Radiological Health
(HFZ-141)
9200 Corporate Blvd
Rockville, MD 20850
Tel: 301-443-2536 x63
E-mail: jfm@cdrh.fda.gov

Hilarie Orman
DARPA
3701 N. Fairfax Dr.
Arlington, VA 22203-1714
Tel: 703-696-2234
E-mail: horman@darpa.mil

Rob Rosenthal
DARPA
3701 N. Fairfax Dr.
Arlington, VA 22203-1714
Tel: 703-696-2264
E-mail: rrosenthal@darpa.mil

John Rushby
SRI International
Computer Science Laboratory
333 Ravenswood Avenue
Menlo Park, CA 94025-3493
Tel: 415-859-5456
Fax: 415-859-2844
E-mail: rushby@csl.sri.com
E-mail: burgess@csl.sri.com
(Judith Burgess, secretary)

Fritz Schulz
DISA-JEXF
5600 Columbia Pike
Falls Church, VA 22041
Tel: 703-681-2350
Fax: 703-681-2813
E-mail: schulzf@ncr.disa.mil

John Silva
DARPA
3701 N. Fairf3701 N. Fairfax Dr.
Arlington, VA 22203-1714
Tel: 703-696-2221
E-mail: jsilva@darpa.mil

Brian Snow
Director, NSA/R2
National Security Agency
ATTN: ISSO Technical Director
Ft. Meade, MD 20755-6000
Tel: 301-688-8112
Fax: 301-688-3090
E-mail: bsnow@radium.ncsc.mil
E-mail: bsnow@dockmaster.ncsc.mil

Dolores Wallace
NIST
NIST NORTH, Bldg 820, Room 517
Gaithersburg, MD 20899
Tel: 301-975-3340
Fax: 301-926-3696
E-mail: dwallace@nist.gov

REFERENCES

American Medical Association. *Prospectus*, “The National Patient Safety Foundation.” [n.d.].

American Medical Infomatics Association (AMIA). AMIA Web site: http://www.amia.org/pubsbroc.htm.

Center for Healthcare Information Management. Position paper on *FDA Draft Policy to Regulate Computer Products*. (November 22, 1996).

Committee on Information and Communications. *America in the Age of Information*. Available at http://www.ccic.gov/ccic/cic_forum_v224/cover.html.

Consortium of Hospital Informatics Manufacturers (CHIM). CHIM Web site. http://www.chim.org/Publications.html.

Feinstein, Alvan R. “System, Supervision, Standards, and the ‘Epidemic’ of Negligent Medical Errors.” Commentary, *Archives of Internal Medicine* 157 (June 23, 1997): pp. 1285-1288.

Kopetz, Herman. “Real-Time Systems: Design Principles for Distributed Embedded Applications,” Kluwer Academic Press, 1997.

Leape, Lucien L., et al. “Systems Analysis of Adverse Drug Events.” *The Journal of the American Medical Association* Vol. 274, No. 1 (July 5, 1995): pp. 35-43.

Microsoft. Bookshelf98.

National Aeronautics and Space Administration. Aeronautics Enterprise Web site. http://www.hq.nasa.gov/office/pao/NASA/aeronautics.html.

National Research Council. National Research Council, *Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability*, Washington, DC, National Academy Press, 1997.

National Science and Technology Council http://www.whitehouse.gov/WH/EOP/OSTP/NSTC/html/NSTC_Home.html

Next Generation Internet Initiative. NGI Initiative Web site: http://www.ngi.gov/.

Nuclear Research Commission. Plant Information Books: http://www.nrc.gov/AEOD/pib/disclaimer.html.

Nuclear Regulatory Commission’s list of available documents: http://www.nrc.gov/NRC/NUREGS/indexnum.html

Perrow, Charles. *Normal Accidents: Living with High Risk Technologies*, Basic Books, NY. 1984.

Peterson, Ivars. *Fatal Defect: Chasing Killer Computer Bugs*. Reprint edition, Vintage Books, 1996.

Seward, John. "Restoring the Ethical Balance in Health Care." Perspective, *Health Affairs* Vol. 16, No. 3 (May/June 1997): pp. 195-197.

Smith, Richard, editor. "The Future of Health Care Systems." *BMJ* Vol. 314, 24 (May 1997): pp. 1495-1496.

The San Francisco Chronicle, August 12, 1997.

U.S. Congress: 1996 Information Technology Management Reform Act (ITMRA) (Clinger-Cohen Act).

LIST OF ACRONYMS

AMA	American Medical Association
AMIA	American Medical Association
AOL	America On Line
CCIC	Committee on Computing, Information, and Communications (successor to the CIC)
CHIM	Consortium of Hospital Informatics Manufacturers
CIC	Committee on Information and Communications
CMU	Carnegie Mellon University
DARPA	Defense Advanced Research Projects Agency
DISA	Defense Information Systems Agency
DNS	Domain Name Service
FAA	Federal Aviation Administration
FDA	Federal Drug Administration
FHA	Federal Highway Administration
FRA	Federal Railroad Administration
GPS	Global Positioning System
HCI	human-computer interface
HCS	high confidence systems
HOL	high order language
HPCC	High Performance Computing and Communications
IC	Intelligence Community
I&C	[digital] instrumentation and control
IDA	Institute for Defense Analyses
I/O	input/output
IPT	Integrated Product Team
IRS	Internal Revenue Service
ITMRA	Information Technology Management Reform Act [1996]
LF	line-feed
LLNL	Lawrence Livermore National Laboratory
JVM	Java Virtual Machine
ML	machine language
MLS	multi-level security
MRI	Magnetic Resonance Imaging
MTBHE	mean time between hazardous events
MTTF	mean time to failure
MTTR	mean time to repair
NASA	National Aeronautics and Space Administration
NGI	Next Generation Internet
NIH	National Institutes of Health
NRL	Naval Research Laboratory
NRC	Nuclear Regulatory Commission; National Research Council
NSA	National Security Agency
OSF	Open Software Foundation
PCC	Proof-Carrying Code
PGP	Pretty Good Privacy
R&D	research and development
RMSL	Requirements Modeling and Specification Language
SCR	[David Parnas method of requirements analyis]

TCB	Trusted Computing Base
TTA	Time-Triggered Architecture
U.S.	United States
V&V	verification and validation
VHDL	Very High Speed Integrated Circuit (VHSIC) Hardware Design Language
WAM	World-Wide Military Command and Control System (WWMCCS) Automation Program
WWMCCS	World-Wide Military Command and Control System