

Computer Systems Technology

U.S. DEPARTMENT OF
COMMERCE
Technology Administration
National Institute of
Standards and
Technology

NIST

Proceedings of the Digital Systems Reliability and Nuclear Safety Workshop (NUREG/CP-0136)

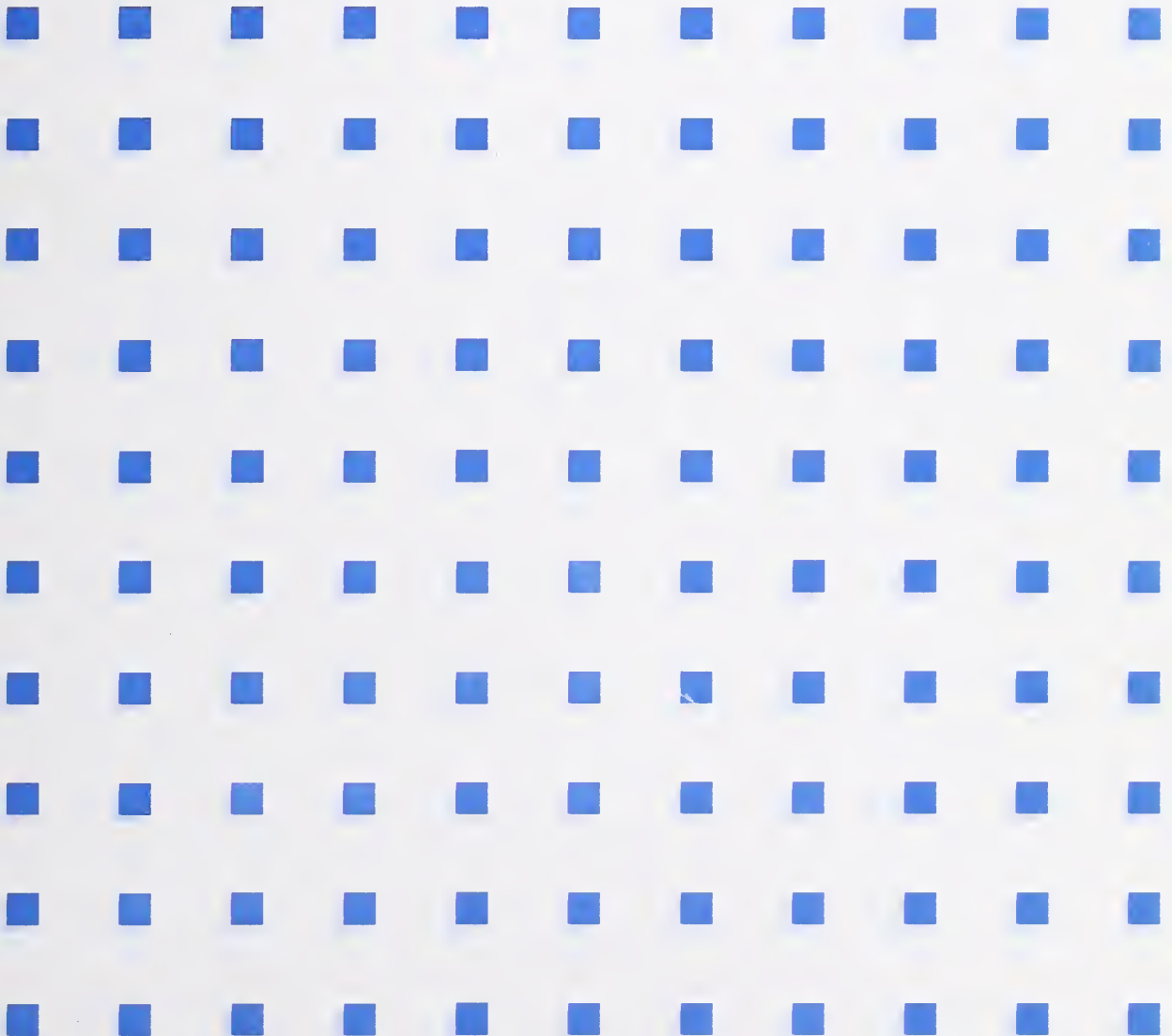
Editors:

D. R. Wallace, B. B. Cuthill
L. M. Ippolito, L. Beltracchi

NIST
PUBLICATIONS



A11104 288163



QC

100

.U57

1994

#500-216

The National Institute of Standards and Technology was established in 1988 by Congress to “assist industry in the development of technology . . . needed to improve product quality, to modernize manufacturing processes, to ensure product reliability . . . and to facilitate rapid commercialization . . . of products based on new scientific discoveries.”

NIST, originally founded as the National Bureau of Standards in 1901, works to strengthen U.S. industry’s competitiveness; advance science and engineering; and improve public health, safety, and the environment. One of the agency’s basic functions is to develop, maintain, and retain custody of the national standards of measurement, and provide the means and methods for comparing standards used in science, engineering, manufacturing, commerce, industry, and education with the standards adopted or recognized by the Federal Government.

As an agency of the U.S. Commerce Department’s Technology Administration, NIST conducts basic and applied research in the physical sciences and engineering and performs related services. The Institute does generic and precompetitive work on new and advanced technologies. NIST’s research facilities are located at Gaithersburg, MD 20899, and at Boulder, CO 80303. Major technical operating units and their principal activities are listed below. For more information contact the Public Inquiries Desk, 301-975-3058.

Technology Services

- Manufacturing Technology Centers Program
- Standards Services
- Technology Commercialization
- Measurement Services
- Technology Evaluation and Assessment
- Information Services

Electronics and Electrical Engineering Laboratory

- Microelectronics
- Law Enforcement Standards
- Electricity
- Semiconductor Electronics
- Electromagnetic Fields¹
- Electromagnetic Technology¹

Chemical Science and Technology Laboratory

- Biotechnology
- Chemical Engineering¹
- Chemical Kinetics and Thermodynamics
- Inorganic Analytical Research
- Organic Analytical Research
- Process Measurements
- Surface and Microanalysis Science
- Thermophysics²

Physics Laboratory

- Electron and Optical Physics
- Atomic Physics
- Molecular Physics
- Radiometric Physics
- Quantum Metrology
- Ionizing Radiation
- Time and Frequency¹
- Quantum Physics¹

Manufacturing Engineering Laboratory

- Precision Engineering
- Automated Production Technology
- Robot Systems
- Factory Automation
- Fabrication Technology

Materials Science and Engineering Laboratory

- Intelligent Processing of Materials
- Ceramics
- Materials Reliability¹
- Polymers
- Metallurgy
- Reactor Radiation

Building and Fire Research Laboratory

- Structures
- Building Materials
- Building Environment
- Fire Science and Engineering
- Fire Measurement and Research

Computer Systems Laboratory

- Information Systems Engineering
- Systems and Software Technology
- Computer Security
- Systems and Network Architecture
- Advanced Systems

Computing and Applied Mathematics Laboratory

- Applied and Computational Mathematics²
- Statistical Engineering²
- Scientific Computing Environments²
- Computer Services²
- Computer Systems and Communications²
- Information Systems

¹At Boulder, CO 80303.

²Some elements at Boulder, CO 80303.

**Proceedings of the
Digital Systems
Reliability and Nuclear
Safety Workshop**
September 13-14, 1993;
Rockville Crowne Plaza Hotel;
Rockville, Maryland
(NUREG/CP-0136)

Editors:

D. R. Wallace, B. B. Cuthill

L. M. Ippolito, L. Beltracchi

Systems and Software Technology Division
Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

March 1994



U.S. Department of Commerce

Ronald H. Brown, Secretary

Technology Administration

Mary L. Good, Under Secretary for Technology

National Institute of Standards and Technology

Arati Prabhakar, Director

Reports on Computer Systems Technology

The National Institute of Standards and Technology (NIST) has a unique responsibility for computer systems technology within the Federal government. NIST's Computer Systems Laboratory (CSL) develops standards and guidelines, provides technical assistance, and conducts research for computers and related telecommunications systems to achieve more effective utilization of Federal information technology resources. CSL's responsibilities include development of technical, management, physical, and administrative standards and guidelines for the cost-effective security and privacy of sensitive unclassified information processed in Federal computers. CSL assists agencies in developing security plans and in improving computer security awareness training. This Special Publication 500 series reports CSL research and guidelines to Federal agencies as well as to organizations in industry, government, and academia.

National Institute of Standards and Technology Special Publication 500-216
Natl. Inst. Stand. Technol. Spec. Publ. 500-216, 348 pages (Mar. 1994)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1994

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

ABSTRACT

The United States Nuclear Regulatory Commission (NRC), in cooperation with the National Institute of Standards and Technology, conducted the Digital Systems Reliability and Nuclear Safety Workshop on September 13-14, 1993, in Rockville, Maryland. The workshop provided a forum for the exchange of information among experts within the nuclear industry, experts from other industries, regulators and academia.

The information presented at this workshop provided in-depth exposure of the NRC staff and the nuclear industry to digital systems design safety issues and also provided feedback to the NRC from outside experts regarding identified safety issues, proposed regulatory positions, and intended research associated with the use of digital systems in nuclear power plants. Technical presentations provided insights on areas where current software engineering practices may be inadequate for safety-critical systems, on potential solutions for development issues, and on methods for reducing risk in safety-critical systems.

This report contains an analysis of results of the workshop, the papers presented, panel presentations, and summaries of discussions at this workshop.

KEY WORDS

Configuration Management, Digital Systems, Formal Methods, Human Operations, Nuclear Power Plants, Nuclear Safety, Software Engineering, System Reliability, Verification and Validation

Certain trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.

ANALYSIS OF RESULTS

The Digital Systems Reliability and Nuclear Safety Workshop was held on September 13-14, 1993, in Rockville, Maryland. This workshop, sponsored by the United States Nuclear Regulatory Commission (NRC) in cooperation with the National Institute of Standards and Technology (NIST), was co-chaired by Mr. Leo Beltracchi of the NRC and Ms. Dolores R. Wallace of NIST. The workshop provided a forum for the exchange of information among experts within the nuclear industry, experts from other industries, regulators and academia.

The 263 participants came from the District of Columbia, 30 states within the United States, and 10 other countries. While most participants represented nuclear industry vendors, utilities, or the NRC, many came from other power industries (e.g., fossil power) national laboratories, and other U.S. government agencies.

This workshop was designed to respond to the need for a forum on the conversion of nuclear power plant (NPP) safety systems to digital technologies. Many analog hard-wired process control systems and safety systems within NPPs are wearing out and frequently are being replaced with systems using digital technology. There are unique design and safety issues in the development and assurance of safety systems using digital instrumentation and control (I&C) systems. The NRC is developing regulatory positions and review guidelines to address these safety issues.

This workshop continued an in-depth exposure of the NRC staff and the nuclear industry to digital systems design safety issues, and also provided feedback to the NRC from outside experts regarding identified safety issues, proposed regulatory positions, and intended research associated with the use of digital systems in NPPs. While this workshop principally addressed the software engineering issues for safety-critical digital systems, the opening speakers provided a system overview and perspective to promote an understanding of the broader issues of safety systems and of the relationship between digital system design issues and the overall development and assurance of safety systems.

The workshop began with welcoming remarks¹ from Commissioner Kenneth C. Rogers (NRC), Mr. Eric S. Beckjord (Director, Office of Nuclear Regulatory Research, NRC) and Dr. J. Ernest Wilkins, Jr. (Chairman, Advisory Committee on Reactor Safeguards, NRC). These speakers identified the benefits of using digital technology in safety systems while acknowledging the risks and the need for regulation and guidance from the NRC.

The next set of speakers provided regulatory, research, and industry perspectives on digital upgrades. Mr. William T. Russell (Associate Director for Inspection and Technical Assessment, Office of Nuclear Reactor Regulation, NRC) presented the regulatory perspective. Mr. Leo

¹Each speaker's complete presentation is included in the body of this document.

Beltracchi (Senior Project Manager, Office of Nuclear Regulatory Research, NRC) stated the research perspective. Mr. Richard J. Blauw (Commonwealth Edison Company) and Mr. Paul K. Joannou (Ontario Hydro) provided the industry perspective.

Three technical sessions followed the opening sessions. The first technical session gave an overview of the state of practice and research in the three major components of digital safety systems: hardware, software and human operations. The second technical session focused on the software engineering issues inherent in developing high integrity systems. The third technical session focused on the methods for reducing risks in the software development process.

The final workshop session was a panel presentation followed by an open question and discussion period. The panel consisted of academic and industry experts on the risks of safety-critical digital technology. The NRC asked the experts to address these four questions:

1. Are the proper issues being addressed?
2. What other issues need to be addressed?
3. Are proposed NRC regulatory positions complete and correct?
4. What are the considerations for further research?

Following the panel, the audience asked questions, challenged the panelists' positions, and provided their own opinions. The result was a valuable dialogue on the future use of digital technology within the nuclear industry, and the open regulatory and technical issues.

At the close of the meeting, Mr. Robert Mullens presented a short position paper written by Mr. Wayne Glidden (Duquesne Light Company) on behalf of the Nuclear Utilities Software Management Group (NUSMG). Mr. Franklin Coffman (Chief, Human Factors Branch, Office of Nuclear Regulatory Research, NRC) and Dr. Cecil Thomas (Deputy Director, Division of Reactor Controls & Human Factors, Office of Nuclear Reactor Regulation, NRC) summarized the workshop issues.

Welcome and Opening Statements

Commissioner Kenneth C. Rogers (NRC), in his opening remarks, observed that while the use of digital technology in I&C systems has proliferated, the existing NRC criteria for analog systems are inappropriate for digital technology. Regulators and industry must apply knowledge from other industries' experiences and from academia about digital I&C systems to the nuclear industry. The nuclear industry has been slow to adopt digital technology, compared with other industries (e.g., avionics, chemical, transportation, and defense). Commissioner Rogers encouraged the nuclear industry to propose and implement, subject to NRC approval, new digital I&C systems. He ended by suggesting that the NRC may have delayed use of digital technology by not having standards and acceptance criteria in place.

Mr. Eric S. Beckjord (Director, Office of Nuclear Regulatory Research, NRC) called on the NRC to address the public safety issues, form complete and proper regulatory positions, and undertake

appropriate research as these digital upgrades occur. While digital systems may be more reliable and perform more functions than analog systems, the purpose of using digital systems is to improve operator performance. Specifically, the use of digital technology should help in I&C systems by reducing the false trip rate, improving process control, and improving the man-machine interface. For example, digital systems have the ability to present information in a more clear and coherent manner, which may help the operator make a quicker and more reliable decision.

Some of the techniques used to ensure analog safety systems are inappropriate or need adaptation for digital systems. For example, the principle of diversity as used in analog systems is based on the belief that the elimination of common-mode failure is not possible by quality alone. For software, the effectiveness of diversity is hard to measure and may affect more than the design. Software testing, configuration control, security control, and formal methods need more study and use in the design of digital safety systems.

Dr. Ernest J. Wilkins, Jr. (Chairman, Advisory Committee on Reactor Safeguards, NRC) has been interested in the nuclear industry's utilization of digital technology for some time and has commented to the NRC on the obvious advantages and disadvantages of digital technology for I&C systems. Dr. Wilkins' position is that the regulations developed before the electronic revolution are inappropriate to the regulation of computerized functions in NPPs. Currently, there is no standard review plan or guide that could help both the Commission and the industry know what is expected of them. Dr. Wilkins affirmed a need for a method to assure correctness of the specification and advocated the use of formal verification and validation (V&V) procedures. Formal V&V procedures assure the correctness of the implementation of the "correct" specification. Dr. Wilkins proposed continuing recruitment by the NRC of additional staff with appropriate digital I&C background to augment current staff capabilities.

Issue Perspectives for Nuclear Power Plants

In this session, Mr. William T. Russell (Associate Director for Inspection and Technical Assessment, Office of Nuclear Reactor Regulation, NRC), Mr. Leo Beltracchi (Senior Project Manager, Office of Nuclear Regulatory Research, NRC), Mr. Richard J. Blauw (Commonwealth Edison Company), and Mr. Paul K. Joannou (Ontario Hydro) provided regulatory, research, and industry perspectives on digital upgrades.

Mr. Russell covered highlights of NRC positions intended to facilitate ongoing reviews. The reviews include those for the advanced boiling water reactors, the Combustion Engineering System 80+, and retrofits on operating reactors. He provided a basis for evolving positions on quality and diversity, and he described a block approach for performing diversity assessments. The NRC staff recognizes the potential for enhanced safety and reliability that digital systems bring to the nuclear industry. The staff also recognizes the challenges to safety that are unique to digital systems implementation. An item used in the process for assessing quality of advanced reactors is the Design Acceptance Criteria (DAC). The NRC specifies top-level system requirements and a detailed design process from system performance requirements through V&V.

The process is broadly based on IEC 880 [IEC880]. This covers the approval of the design process, not the approval of the actual design of the system.

The four major elements, according to Mr. Russell, of the NRC's diversity position are the following:

1. The applicant shall assess the defense-in-depth and diversity of the proposed I&C system to demonstrate that vulnerabilities to common-mode failures have been adequately addressed.
2. In performing the assessment, the vendor or applicant shall analyze each postulated event in the analysis section of the safety analysis report (SAR) using best estimate methods. The vendor or applicant shall demonstrate adequate diversity within the design for each of these events.
3. If a postulated common-mode failure could disable a safety function, then a diverse means, with documented bases that the diverse means is unlikely to be subject to the same common-mode failure, shall be required to perform either the same function or a different function. The diverse or different function may be performed by a non-safety system if the system is of sufficient quality to perform the necessary function under the associated event conditions.
4. A set of controls located in the main control room shall be provided for system level actuation and control of critical safety functions. The displays and controls shall be independent and diverse from the safety computer system identified in Items 1 and 3.

Mr. Beltracchi presented an overview of the NRC research activities and stressed the need to define a technical basis for digital system requirements. A technical basis comprises the following:

1. A requirement has been clearly coupled to safe operations.
2. The scope of the requirement is clearly defined.
3. A substantive body of knowledge exists and the preponderance of evidence supports a technical conclusion.
4. A repeatable method exists for correlating relevant characteristics with performance.
5. A threshold for acceptance can be established.

Mr. Beltracchi led the audience through the history of NRC's regulations, which included an analysis of standards used within the nuclear industry. Only two standards cite digital systems. Mr. Beltracchi also identified two major regulatory areas of digital upgrades for the NRC to address: the diversity requirements for safety algorithms and computer unique requirements. He presented an outline for a framework for an NPP safety system. When completed, this framework would provide guidance in organizing the digital system requirements for the hardware, software and the human operator components.

Mr. Blauw voiced a concern in his presentation that while the top-level view or design of digital systems may appear simplistic, the implementation of safety regulations is complex. The nuclear industry must look at other industries for process control and monitoring. He will be working on the revision of American Society of Mechanical Engineers (ASME) standard on Nuclear Quality Assurance (NQA), Part 2.7 [ASMENQA2] to explain differences between design verification and V&V, differences between configuration control and configuration management (CM), and documentation issues. Mr. Blauw described his experiences with implementing digital systems in which the concern was the cost-effectiveness of providing the necessary assurance of the safety of these systems. Mr. Blauw then described the IEEE Standard 7-4.3.2 "Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations" [IEEE7432]; utilities were involved in developing this standard². This standard is intended to be used with IEEE Standard 603 "Standard Criteria for Safety Systems for Nuclear Power Generating Stations" [IEEE603].

He also indicated that the Nuclear Utilities Management and Resources Council (NUMARC) will publish a digital upgrade guideline which recommends a licensing approach, and that the Electric Power Research Institute (EPRI) will publish other guidelines. The standards and guidelines address some principal issues for system design models including project management, configuration control, failure and error analysis management, and independent review. Currently, a major problem is that plant drawings and other forms of design documentation are not under configuration control and are frequently incorrect with respect to the current plant configuration.

Ontario Hydro of Canada has vast experience in applying digital system technology in a NPP. Mr. Joannou described some of the issues encountered while licensing the Darlington Reactor Station, which used a fully computerized shutdown system. One problem was lack of a widely-accepted definition of "good enough" software; the deficiency led to joint development between Ontario Hydro and Atomic Energy Canada Limited (AECL) of a family of software engineering standards, guidelines and procedures for NPP protective, control, and monitoring software. Major issues addressed by AECL and Ontario Hydro include the reviewability of the software, safety functions, ambiguities in requirements specifications, software reliability and software maintainability. One of the problems is that the use of software analysis techniques, such as reengineering and hazard analyses, is costly. The industry needs to develop cost-effective analysis methods. The new standards and guidelines developed during this experience provide

²This standard was officially approved by the IEEE Standards Board at their meeting on September 15, 1993.

rules for documentation, test types (statistically valid, trajectory-based random tests, systematic tests), software CM (SCM), audits, qualifications of personnel and independent V&V.

Technical Session on Digital Safety Systems for Nuclear Power Plants

Mr. A.L. Sudduth (Duke Power), Dr. John C. Cherniavsky (National Science Foundation (NSF)), and Dr. Lewis F. Hanes (nuclear industry independent consultant) discussed the problem of replacing an analog with a digital control system in a NPP from the different perspectives of hardware, software, and human factors. These different views of the problem resulted in different definitions of the problem and led to different approaches to solving the defined problem.

Mr. Sudduth provided a hardware view of replacing an analog with a digital system from the perspective of a senior engineer involved in digital upgrades for control systems in fossil fuel plants. His concerns are to maintain or improve the level of safety available from the new system, but also to minimize the downtime for the plant. Mr. Sudduth proposed several alternate hardware designs using proven components to improve safety. To minimize plant downtime, Duke Power used a complete control-room simulator to speed up the process of testing the new system in a setting which closely approximates the actual control room and for training personnel to work with that new system. Digital upgrades for Duke Power now require only 3 months of downtime.

Dr. Cherniavsky provided a software view of the problem of installing a digital upgrade of an analog system from a research perspective. He discussed the research supported at NSF in the High Performance Computing Communications Initiatives and placed that within the larger context of NSF software research support. The NSF has a continuing interest in topics related to safety-critical software. In the past, NSF has funded formal methods research and, in the future, may develop a Center for Software Safety studies.

Dr. Hanes provided a human factors view of the problem of converting a control system from analog to digital displays. Dr. Hanes raised issues across the entire system lifecycle including new requirements that might be imposed on these systems, a better understanding of anthropometrics and biomechanics, designing these systems to provide the information the operator needs when he or she needs it, and intelligent aids to support the operator's decision making after the system is in operation. Dr. Hanes drew on experience in other industries to identify ways that a digital system can improve crew performance, enhance plant safety, and avoid problems encountered in those industries.

Each speaker's discussion of methods for addressing the problems reflect their experience and expertise in a specific technology. Mr. Sudduth viewed the conversion of control systems as essentially a solved problem from a hardware perspective. The hardware is available and in use in fossil plants which have an established conversion process. The use of digital control systems has supplied substantial data on the human factors issues in converting plants, but these issues are not entirely resolved and still require more research according to Dr. Hanes. In contrast,

software engineers are just beginning to grapple with some of the issues in developing safety-critical systems according to Dr. Cherniavsky.

Technical Session on Software Engineering

The technical session on software engineering focused on techniques for improving the software development process. The premise of all the speakers in this session was that software developed using the methods generally in use today does not meet the needs of high integrity systems. The speakers identified two general areas of concern. The first two speakers focused on the need to provide accurate specifications. The last two speakers moved the discussion into the need to design, implement, and test systems to meet the specifications.

Dr. John Knight (University of Virginia) focused on the need to separate the functions of systems engineers and software engineers, and to ensure an effective interface between them. His point was that software engineers do not have the system or application knowledge to make decisions about the behavior of the full system. Deciding what the system should do under each set of circumstances is the job of the systems engineer. Software engineers need to receive a precise system specification including all the assumptions and constraints that the systems engineer expects the software to maintain.

Dr. John McHugh (Portland State University) focused on the need to provide a nonambiguous communication mechanism for the systems and software engineers. This mechanism should be some form of formal specification understandable both to system and software engineers. Dr. McHugh acknowledged the difficulty of learning formal specification languages and understanding formal specifications. While the use of explanatory text can mitigate some of these problems, it introduces a new problem in defining which of the two specifications is controlling, the formal or the English language one.

Mr. Robert M. Poston (Interactive Development Environments) focused on the need for and benefits of providing tool support for developing a formal specification. Developing a formal specification is difficult and labor intensive. Mr. Poston discussed the use of tools that make the process less "user-hostile" by allowing the developer to create the specification in diagrams which can transform into an easily understood notation. The developer can then check the produced specification. Another advantage of a formal specification in a defined notation is that a test generation tool can take that specification and produce test cases.

Dr. Barbara B. Cuthill (NIST) moved the discussion from the requirements specification to the design and implementation portions of the software lifecycle. Specifically, she discussed general features of object-oriented design (OOD) and C++ development. Dr. Cuthill enumerated risks and benefits of using OOD and C++ with respect to specific criteria important to safety-critical systems such as functional diversity. While she did not form a final conclusion, she presented many of the issues that need to be addressed as OOD and C++ become more widely used in industry.

The need for traceability of the requirements, design, code and test cases back to a set of nonambiguous specifications provided a common thread that all the speakers identified as important and discussed in some form. Dr. Knight began by discussing the need for systems engineers to establish complete specifications. Dr. McHugh continued by discussing how systems engineers can provide nonambiguous specifications. Mr. Poston discussed the methods for generating test cases traceable to the specifications while Dr. Cuthill discussed methods for designing and coding the software while maintaining traceability. While each of the speakers also addressed other important issues in the development of safety-critical systems, each one returned to this theme of traceability and maintaining the link between the phases of the software development lifecycle.

Technical Session on Risk Reduction

The technical session on risk reduction provided interpretation on both the problems in achieving and assuring the safety of high integrity software and possible solutions to the problems. The speakers, Dr. Winston Royce (TRW, Inc.), Ms. Anne-Marie Lapassat (Commissariat à l'Energie Atomique), Mr. H. Ronald Berlack (Configuration Management International), Dr. Lance A. Miller (Science Applications International Corporation), Ms. Charlotte O. Scheper (Consultant), Mr. Kyle Y. Rone (IBM), Dr. William Everett (AT&T), and Mr. Roger U. Fujii (Logicon, Inc.), provided insights from their experiences in defense, nuclear, space, and communication industries.

Dr. Royce identified the following six areas of concern with regard to safety-critical systems, and elaborated on their meaning during his presentation:

1. Safety-critical systems are implemented in the *wrong languages*.
2. There are *not enough tools* for safety-critical systems development.
3. There is insufficient analysis and distribution of *error measurement* data.
4. There is *no organizational certification*.
5. There is *no people certification*.
6. There is *infrequent graspability* of the full system functionality.

Ms. Lapassat, whose presentation preceded Dr. Royce's, and the six speakers immediately following Dr. Royce, provided some techniques for addressing most of these problems.

While several speakers discussed the need for and availability of good tools, the only speaker in this session to focus on tool development was Ms. Lapassat. She discussed simulation, testing and auditing tools for independent evaluation of the software in NPPs developed for the French nuclear regulatory agency. These tools focused on testing the control system to see if it met the required timing constraints by simulating real-time, normal and abnormal operation.

Other speakers discussed tool availability and usage in relation to support for specific processes. Mr. Berlack focused on the importance of SCM and that organizations need to have both the tools and processes in place to support CM. Dr. Miller focused on the need for organizations to support V&V and to use automated V&V test tools whenever possible.

Two complementary techniques were discussed as ways to manage the complexity of large software projects: CM and reuse. Mr. Berlack strongly endorsed the use of CM as a means of communicating between the systems and software engineers and maintaining traceability. Good CM tools can provide a mechanism for tracing the impact of one change on the overall system. Dr. Miller reiterated this need for CM to trace and allocate requirements to all the development artifacts (e.g., specifications, code) as a means of supporting V&V.

Ms. Scheper described typical approaches for certifying and reusing software components. She provided an alternative approach to simplifying complex systems through the certification of reusable components. Ms. Scheper developed a certification framework for software for high integrity systems. Software components are maintained with the requirements they meet. The framework grades the requirements and component based on the level of confidence that the component meets the specification, the level of criticality of the software, and the level of assurance used to test the software.

Three of the speakers discussed the need for and use of empirical data on the software development process and software error measurement. Mr. Rone discussed the need for models of error discovery. Dr. Everett addressed the question: "Can we apply software reliability engineering techniques to safety-critical systems?" Dr. Everett discussed the test acceleration methods that isolate safety-critical functions for extensive testing to achieve higher estimates of reliability. Dr. Miller also discussed the use of models to estimate the numbers and types of errors remaining in a system. Many speakers and members of the audience agreed that public availability of error data would provide valuable information but that it is not realistic to expect companies to release this data.

While no speaker directly addressed Dr. Royce's call for certifying an organization's capability to produce safety-critical software, several speakers discussed the need for organizations to implement repeatable processes supported by CM. These capabilities are necessary to establishing a corporate ability to produce software. Both Mr. Berlack and Mr. Rone identified process definition as a required precondition for an organization to produce useful metrics data. Without a defined process, it is not clear what the collected data means across different projects. Mr. Rone linked the production of usable metrics data to quality, cost, and schedule planning.

The speakers also linked process definition, metrics collection, project planning, and V&V activities to CM. Mr. Berlack explicitly discussed the link between CM and the establishment of consistent planning, traceability, formal releases, change management, status accounting and auditing. All of these capabilities are important for an organization to define its software development process. Mr. Fujii and Dr. Miller discussed the need for CM and specifically traceability to support a V&V process.

Mr. Fujii discussed software V&V in the context of the system, i.e., software V&V is a systems engineering discipline that evaluates software as part of the entire system, including hardware,

human operators, and other interfacing software (e.g., operating systems, printers). He provided guidance on estimating the cost of software V&V based on two concerns:

1. The criticality of system-specific functions and other system parameters (e.g., security, usability, maintainability, and performance).
2. Risks of the development environment (e.g., system architecture maturity, processor technology suitability, development methodology, maturity of development tools and aids, staff skills, schedule, and software application maturity).

The criticality analysis process requires traceability from system functions to all other components to define the system behavior and to trace the implementation of the system functions through all system documentation and code.

Mr. Fujii described a system safety framework that assists in estimating how much of the software to analyze and test. In modern systems the interaction of software with the hardware, human operators, and other software elements is more complex and interwoven in the total system solution than existed previously. The system performance functions must be specified and analyzed. Software V&V must also analyze the allocation of the system requirements to ensure that critical requirements are traceable and are allocated so as to make integration and testing less difficult and time-consuming.

No speakers in this session addressed the other two issues that Dr. Royce mentioned: language selection or developer certification; however, speakers in other sessions, the panelists in the last session, and many audience members discussed and debated these issues.

In this session, like the previous technical session, two major themes were the importance of maintaining and verifying the traceability of specifications across a complex system, and maintaining the system context for the software. Mr. Berlack discussed the use of CM as a vehicle for communication between the systems and software engineers because it provides the mechanism for tracing the elements of development artifacts back to the specifications. Dr. Miller emphasized the importance of traceability to V&V activities. Mr. Fujii emphasized the need for systems engineers not only to be able to trace the specification to the design implementation but to be heavily involved in the software V&V process since it is the systems engineers who know how the full system, not just the software, should behave. Ms. Scheper also emphasized the need to understand the requirements on software in the context of a full system to be able to select reusable software components and catalog those components correctly.

Panelist Perspectives

Four experts in safety-critical software or systems areas were invited to be on a panel to discuss the application of the workshop to NRC activities. These were Dr. John Knight (University of

Virginia), Dr. John McHugh (Portland State University), Dr. Winston Royce (TRW, Inc.), and Dr. Joseph Naser (EPRI). They were asked to address the following four questions:

1. Are the proper issues being addressed?
2. What other issues need to be addressed?
3. Are proposed NRC regulatory positions complete and correct?
4. What are the considerations for further research?

The speakers addressed these questions generally. While the panelists did agree that issues addressed at the workshop are proper and important for the assurance of software in safety systems, they posed the questions differently and provided a background against which the questions should be discussed. No one questioned the appropriateness of the issues discussed by workshop presenters. Rather, the context for the discussion was the major issue. The panelists agreed that the consequences of software failure relative to system behavior should drive assurance activities and that the cost-effectiveness of those activities is also a major consideration. Among the panelists and speakers there was some dissension about the goal of achieving perfect software; many felt that there is an unacceptably tolerant attitude toward software error. The panel members presented ideas on how the context of consequence of failure and cost effectiveness will influence how the remaining questions should be addressed.

Dr. Knight used a general approach for addressing the questions. With respect to the proper issues, he stated that first the consequences of failure must be identified and then one can determine if the right issues are addressed relative to the (potential of) failure. If the consequences are significant, then the question is to identify how to avoid failure to the extent possible. Dr. Knight suggested that those topics which need research are those identified as potential failure areas, but which were not yet addressed in the workshop. He offered two considerations:

1. Ensure that the activities proposed for regulation are appropriate
2. Ensure that the people building the digital systems are doing what is intended. With current requirements, there are plenty of opportunities for ambiguity and misunderstanding of the requirements.

To help define the requirements, steps might be taken to investigate the application of the body of software engineering knowledge specifically to the problems of the nuclear industry by performing more analyses of design methods, languages, and other technologies relative to their advantages and disadvantages to fulfilling the requirements of avoidance of failure in digital systems.

Dr. John McHugh cited the lack of the use of science and mathematics in software *engineering* compared to the scientific and mathematical bases in other engineering fields. Without such a background in software engineering, demonstration of compliance to "principles" may be difficult. The disciplines of engineering need to be applied to software problems. He perceived that the

utilities are unwilling to conduct the research into the best, safest, and most appropriate ways to control plants. Yet, there is a need for research in both the safety and operating control areas, because these are likely to be based on more elaborate models than in the past with expectations of higher degrees of operational efficiency. Dr. McHugh believes someone must take a prescriptive role, not an advisory role, to tell industry what they must do to measure safety.

From Dr. Royce's perspective, the most important issue is to overcome the tolerance for errors in software. Whether the workshop and the NRC are addressing the right issues and whether the regulatory position is right are related directly to the consequence of failure. Software is error-prone; software is the stringing together of lines of logical elements. While people skilled in mathematics and science tend to be skilled in logic, there is no higher principle in software than logic. Humans develop and judge software. The effort to discover and remove the errors is costly, and no one is willing to pay for the effort. Research is expended on how to design and how to code software, not on discovering why errors exist at all, or on error finding techniques like requirements analysis. The consequence of failure must be understood to remove that tolerance for error among the buyers, sellers and builders of software and promote research into error discovery and removal.

While several workshop presenters discussed the benefits of using digital systems, Dr. Naser claims this topic is no longer an issue because there is agreement on the benefits of digital systems. The major issue to be addressed by the NRC is how to allow safe implementation of digital systems in a cost-effective manner. An approach may be to stabilize the licensing process with well-defined requirements for qualification. This entails the following steps:

1. Definition of a technical basis for the well-defined requirements and a process for achieving them. The basis should incorporate compensating factors (e.g., operating experience, software development techniques, and error reporting) against lack of a formal process.
2. Description of requirements for system acceptance.
3. Use of system behavior as the driver for the requirements.
 - Consequence of failure with plant safety as highest goal.
 - Application of defense-in-depth to work around imperfections.
4. Application of knowledge from other industries. This may lead to use of proven systems, reusable software, and use of new technology (e.g., artificial intelligence to assist operators).

In summary the panel provided the following advice:

1. Specify the consequences of failure relative to system behavior.
2. Apply software technology from other application domains which have been developed from mathematics.
3. Define acceptance criteria, which address the software process with a technical basis, software product quality, and cost-effectiveness.

Questions and Discussion after Panel Presentations

Many members of the audience took the opportunity in the discussion session to comment on the entire workshop or on presentations from other sessions.

One frequent comment from the industry participants was that the focus of the workshop had been on the problem of developing safety-critical *software* while the nuclear industry was concerned about safety-critical *systems*. These comments from the audience echoed comments from the speakers. Many of the speakers insisted that systems engineers and application experts must specify the software requirements for these systems and be involved in the entire development process.

A second group of comments from the industry participants related to the need to bring in more of the experience of other industries in converting to digital systems. Digital control hardware used in other industries may be useful to the nuclear industry; however, the embedded software for these controllers may need adaptation. There is considerable experience in the fossil fuel, chemical and other industries with safety-critical control systems, and the nuclear industry needs to draw on that experience.

A third group of comments related to the inherent complexity of the software required for safety systems in NPPs. There was debate among the speakers and the audience on this point. Many of those in the nuclear industry contended that the current safety systems and the software to run the digital safety systems were actually fairly simple. This debate existed in part because different audience members and speakers had different perspectives on the problem. One perspective was that safety systems based on analog hard wired technology are easier to understand. Some people included the entire control system while others narrowed the focus to only the smaller safety system. Another reason for the disagreement was that the system engineers tended to view the safety system at a high level as having a simple overall function, and the software engineers tended to think of the system as lines of code which is analogous to the level of resistors and capacitors for systems engineers. The software engineers also tended to assume that the simple analog systems would be replaced by more complex equipment that would perform more functions for the operator. Others argued that while this added functionality might theoretically improve safety, the problems of certifying that the more complex software was accurate made the additional functionality no longer cost-effective.

While most speakers accepted conversion of NPPs to digital systems as a given, others felt that the conversion would not produce safer plants and would not be cost-effective. The speakers and audience agreed that all upgrades were costly. Specially fabricated analog parts would be far more costly than buying off-the-shelf digital equipment; however, the hidden costs of digital equipment were a major concern. The nuclear industry needs to know how the digital hardware and software can be certified for use and what the cost of this certification will be. The digital hardware and software used in the upgrades may still have to be custom designed and built. In addition, the nuclear plants will have to retrain their work force on the new equipment.

What the discussion illustrated was the need for systems engineers who understand the nuclear industry's application needs and the software engineers who know how to embed reliable software in systems to agree on a common vocabulary. The systems engineers or applications experts and the software engineers were frequently talking about the same problem, but from different perspectives. These different perspectives tended to confuse the issues under discussion.

NRC Closing Remarks

At the close of the meeting, Mr. Franklin Coffman (Chief, Human Factors Branch, Office of Nuclear Regulatory Research) and Dr. Cecil Thomas (Deputy Director, Division of Reactor Controls and Human Factors, Office of Nuclear Reactor Regulation) summarized the workshop issues.

Mr. Coffman provided the following, tentative list of issues that resulted from the conference.

1. The means to obtain a complete and precise translation of a using organization's needs into design specifications. This included the issues surrounding the role of formal methods for specification capture and analysis.
2. The question of allocating the requirements between the hardware and the software while defining the interface requirements between the digital system, the driving software, the human operators and maintainers, the plant systems, and the power conversion phenomena. Yet it is the total system that is to be evaluated including the consequences of software failure on the total-system's performance.
3. The issues surrounding the role of hazard analysis for defining the level of detail at which fault-tolerance is required.
4. Questions on doing common-mode-error analysis and questions on the technical basis for criteria to invoke diversity as a defensive measure. The questions include considerations of the net-benefit of diversity.
5. The question of adequate reliability metrics for important systems' properties like complexity, and the relationships of the metrics to the degree of safety obtained.

6. The potential for a response-time hazard in digital systems because they are incremental (in contrast with the continuous nature of analog-hardwired systems).
7. The issue of the role of specification-based and statistical testing requirements and acceptance criteria.
8. The acceptance or certification of Commercial Off-The-Shelf software for safety-critical applications.
9. The issues associated with the transition from analog to digital including 10CFR50.59 reviews and Unreviewed Safety Questions.
10. The issues associated with the net benefits to a system's reliability from developing software using structured processes and structured languages, and improved languages.
11. The questions associated with the use of CASE tools for design specifications, testing design, and safety reviews.
12. The role of V&V and the degree to which different techniques assure reliable software, and the degree to which the V&V must be independent of the designer.
13. The issue of standards or conventions for controlling software configurations.
14. Questions concerning the need to qualify or certify compilers and operating systems.
15. Questions of adequate isolation of non-safety related software from safety-related software.
16. Questions on the degree of domain knowledge needed by developers of software for nuclear applications.
17. Generally, the need for a comprehensive framework/outline for the scope and content of the technical basis and acceptance criteria for digital I&C systems.
18. The question of the need for further research and subsequent workshops on topics such as hardware and human factors.
19. Interest in the possibility of the NRC initiating a process where error experience is collected, analyzed, characterized, and distributed.
20. The impact of the trend toward the use of blocks of experienced code versus the conventional development of code.

Summary and Conclusions

Many speakers reinforced the need for a technical basis for regulatory requirements. Speakers in the technical sessions agreed on the need for defining software requirements in the context of system requirements and for the traceability of those requirements across the entire software lifecycle. While there was some debate among the speakers and audience about the degree of complexity inherent in safety systems, the speakers agreed that systems engineers need to clearly state the requirements, constraints, and assumptions for the safety system. One difficulty in communicating the requirements is that the terminology of the nuclear systems and software engineering communities are different. These differences can lead to miscommunications about the requirements which may have a safety impact. For example, in system development, the design phase includes the development of the software requirements and the software design. In the nuclear industry, design specifications could be system design specifications, the software requirements specification, and functions allocated to the operator. This problem is evident in the study of standards and guidelines conducted by NIST for the NRC [NUREG5930]. Another problem identified in that study and discussed by speakers in this workshop is the requirement for CM because most of the guidance documents did not invoke CM during the entire life cycle, nor was the impact on SCM clear.

Several speakers emphasized the importance of precise specifications which are traceable and maintainable throughout the development process. Dr. Knight introduced the subject by stressing the importance of systems and application engineers completely specifying the software requirements, and that software engineers are not qualified to make decisions about the system. Dr. McHugh, Mr. Poston, and Mr. Berlack all discussed the means for systems and software engineers to specify nonambiguously the requirements and maintain the traceability of those requirements through the software development process. Dr. McHugh endorsed formal methods as a precise means of specifying requirements. Mr. Poston agreed that formal methods were a good choice, but advocated the use of tools to make the process easier and permit traceability between the tests and the requirements. Mr. Berlack stressed CM as a mechanism for precise continuing communication between the systems and software engineers. The CM method can allow the systems engineer to see how the software engineers allocated the requirements. Dr. Hanes also stressed the importance of maintaining the requirements allocation. Dr. Cuthill discussed maintaining traceability of the requirements through the design and coding phases. Mr. Fujii emphasized that software V&V had to refer back to the system specifications and involve system engineers. Finally, Ms. Scheper discussed the need for maintaining the system context for software components kept in a reuse library so that they can be included in future systems appropriately. Her approach for certifying software components and making them available for reuse in other systems could reduce the aggregate cost of software certification.

Another theme of all the technical sessions was the need for better acceptance and V&V testing. Mr. Fujii emphasized that software V&V can make a significant contribution to analyzing the allocation of functions, early in the system development. Mr. Poston, Ms. Lapassat and Dr. Everett stressed the need to automate testing. Measuring the reliability of software is a non-trivial problem but a measure of reliability is important for assuring safety. Mr. Rone,

Dr. Everett and Dr. Royce proposed partial solutions to the problem including incorporating specification-based testing and statistical testing. Dr. Royce also cited the value of a structured process and structured process language for effecting reliability.

Other major issues discussed by the speakers concerned the need for (system and software) hazard analysis requirements, fault tolerant requirements, and common mode error and diversity requirements.

Tables 1, 2, and 3 present a summary of the issues addressed from another perspective. This summary divides the topics discussed into the problems that the speakers identified, the theoretical solutions they proposed and any experience based solutions offered. These topics are further grouped by life cycle process. Table 1 contains topics raised in the software requirements and design processes. Table 2 contains topics for the processes of implementation and integration and installation, operations, and maintenance. Table 3 includes topics in the V&V and quality assurance areas.

While the speakers generally agreed on the importance of these issues, there was, at times, a discrepancy between what the speakers believed to be the issues facing the nuclear industry and what the audience (nuclear industry and regulatory personnel) believed to be the important issues.

Table 1. Requirements and Design Lifecycle Phases

	Requirements	Design
Problems Identified	Dr. Knight <ul style="list-style-type: none"> ■ Incomplete specifications ■ Formal methods not enough ■ Software engineers not qualified to deal with systems issues 	Dr. Royce <ul style="list-style-type: none"> ■ Few tools ■ System too complex to understand Mr. Rone <ul style="list-style-type: none"> ■ System too complex
Theoretical Solutions	Dr. McHugh <ul style="list-style-type: none"> ■ Use formal, precise specifications Mr. Poston <ul style="list-style-type: none"> ■ Use tools to generate specification & test cases 	Dr. Cuthill <ul style="list-style-type: none"> ■ OOD is potentially useful Dr. Miller <ul style="list-style-type: none"> ■ Allocate functions to operators
Application Experience	Mr. Joannou <ul style="list-style-type: none"> ■ Formal specification (Parnas) ■ Defined process ■ Developed OASES Framework ■ Tools for consistency checking Mr. Blauw <ul style="list-style-type: none"> ■ Standard IEEE P-7-4.3.2 ■ NUMARC Digital Upgrade Guideline 	Mr. Joannou <ul style="list-style-type: none"> ■ Fail safe and self check features ■ Defined process Mr. Sudduth <ul style="list-style-type: none"> ■ Probabilistic Risk Assessment ■ Alternate fault tolerant architectures ■ Fault tree and Failure Modes and Effects Analysis (FMEA) ■ Markov Models

Table 2. Implementation/Integration and Operation/Maintenance Lifecycle Phases

	Implementation & Integration	Installation/Operation/Maintenance
Problems Identified	Dr. Royce <ul style="list-style-type: none"> ■ No error measurement ■ Languages not designed for safety 	Dr. Hanes <ul style="list-style-type: none"> ■ No guidelines for reviewing human interfaces ■ Define extent of automation
Theoretical Solutions	Dr. Cuthill <ul style="list-style-type: none"> ■ C++ is potentially useful Ms. Scheper <ul style="list-style-type: none"> ■ Levels of criticality assurance for reusable software artifacts 	Dr. Cuthill <ul style="list-style-type: none"> ■ OO & C++ simplify maintenance Mr. Russell <ul style="list-style-type: none"> ■ Self-diagnostics and testing Dr. Hanes <ul style="list-style-type: none"> ■ Intelligent displays and aids ■ Computerized procedures
Application Experience	Mr. Berlack <ul style="list-style-type: none"> ■ Software Configuration Management 	Mr. Berlack <ul style="list-style-type: none"> ■ Software Configuration Management Mr. Joannou <ul style="list-style-type: none"> ■ Defined acceptance criteria

Table 3. Verification/Validation and Quality Assurance

	Validation	Quality Assurance
Problem Identification		Dr. Royce <ul style="list-style-type: none"> ■ No people certification ■ No organization certification
Theoretical Solutions	Mr. Poston <ul style="list-style-type: none"> ■ Tool generated tests from requirements Dr. Miller <ul style="list-style-type: none"> ■ Fault specific verification ■ Cost benefit tradeoff for V&V ■ Automation of V&V methods 	Mr. Rone & Ms. Olson <ul style="list-style-type: none"> ■ Define development process ■ Develop quality plan ■ Use cost, schedule & error detection models ■ Configuration management Ms. Scheper <ul style="list-style-type: none"> ■ Levels of certification for reusable s/w
Application Experience	Mr. Joannou <ul style="list-style-type: none"> ■ Software hazard analysis (Leveson) ■ Guided inspection of software Mr. Fujii <ul style="list-style-type: none"> ■ Estimation of V&V necessary ■ Process to select V&V methods Mr. Sudduth - ■ Simulation of system for testing Dr. Everett - ■ Isolation of safety-critical components for test acceleration Ms. Lapassat <ul style="list-style-type: none"> ■ Tools for independent verification ■ Software simulation & modeling tools 	Mr. Berlack <ul style="list-style-type: none"> ■ Configuration & change management ■ Status accounting and auditing ■ Subcontractor control Mr. Fujii - ■ System safety framework Mr. Joannou <ul style="list-style-type: none"> ■ Train personnel in methodologies and retrain as necessary Mr. Sudduth <ul style="list-style-type: none"> ■ Use proven components Ms. Lapassat <ul style="list-style-type: none"> ■ Tools for independent auditing

The speakers and audience seemed to disagree on the following:

1. The degree to which software may be a safety concern.
2. Simplicity versus complexity of the software needed in digital systems.
3. The perceived cost of assuring software versus the cost affordable by the nuclear industry.

The speakers and audience seemed to agree on the following:

1. Standards and criteria should develop from a defined technical basis, which is not currently available.
2. There are plenty of benefits from using digital systems, such as self-diagnostics and less chance of human error, but operators must remain in charge.
3. There are opportunities to make improvements in some known failure classes (omitted function, unintended function).
4. A definition of diversity for software is needed.
5. Future workshops should address other components of NPPs (e.g., hardware, human operators).
6. The software engineering concepts presented at the workshop were not strongly connected to the problems faced by the nuclear industry, i.e., their vendors would be more suitable to deal with these issues.

The aggregate of technical presentations and issue perspectives leads to the conclusion overall that many of the management and technical problems of digital systems are not sufficiently mature for regulation. Research to define technical solutions and research into existing solutions in other industries is necessary.

References

[ASMENQA2]

ASME NQA-2a-1990, Part 2.7, "Quality Assurance Requirements for Nuclear Facility Applications," The American Society of Quality Engineers, 1990.

[IEC880]

IEC 880, "Software for Computers in the Safety Systems of Nuclear Power Plant Stations," International Electrotechnical Commission, 1986.

[IEEE603]

IEEE Std 603-1980, "Standard Criteria for Safety Systems for Nuclear Power Generating Stations," The Institute of Electrical and Electronics Engineers, Inc., 1980.

[IEEE7432]

IEEE Std. 7-4.3.2-1993, "Application Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear power Generating Stations," American Nuclear Society, 1993.

[NUREG5930]

"High Integrity Software Standards and Guidelines," Wallace, Dolores R., Laura M. Ippolito, D. Richard Kuhn, NUREG/CR 5930, U. S. Nuclear Regulatory Commission, September 1992. (Also published as National Institute of Standards and Technology NIST SP 500-204.)

TABLE OF CONTENTS

		<u>Page</u>
1	INTRODUCTION	1
2	THE NUCLEAR REGULATORY COMMISSION AND THE ADVISORY COMMITTEE ON REACTOR SAFEGUARDS	3
2.1	Welcome: Commissioner Kenneth C. Rogers	5
2.2	Welcome and Opening Statement: Mr. Eric S. Beckjord	9
2.3	Welcome and ACRS Perspective: Dr. J. Ernest Wilkins, Jr.	13
3	ISSUE PERSPECTIVES FOR NUCLEAR POWER PLANTS	17
3.1	Presentation on NRC Regulatory Positions and Guidelines: Mr. William T. Russell	21
3.2	NRC Research Activities: Mr. Leo Beltracchi	31
3.3	Industry Perspective on Digital Upgrades: Mr. Richard J. Blauw	47
3.4	Experiences from Application of Digital Systems in Nuclear Power Plants: Mr. Paul K. Joannou	61
3.4.1	Questions: Mr. Paul K. Joannou	78
4	DIGITAL SAFETY SYSTEMS FOR NUCLEAR POWER PLANTS	79
4.1	Hardware Aspects of Safety-Critical Digital Computer Based Instrumentation and Control Systems: Mr. A.L. Sudduth	81
4.1.1	Questions: Mr. A.L. Sudduth	105
4.2	Software Aspects for Safety-Critical Systems: Dr. John C. Cherniavsky ..	107
4.3	Human Aspects for Safety-Critical Systems: Dr. Lewis F. Hanes	109
4.3.1	Questions: Dr. Lewis F. Hanes	130
5	SOFTWARE ENGINEERING FOR HIGH INTEGRITY SYSTEMS	131
5.1	Interaction Between Systems and Software Engineering in Safety-Critical Systems: Dr. John Knight	133
5.1.1	Questions: Dr. John Knight	136
5.2	The Role of Formal Specifications: Dr. John McHugh	139
5.2.1	Questions: Dr. John McHugh	145
5.3	Specification-Based Testing: What is it? How Can It Be Automated?: Mr. Robert M. Poston	149
5.3.1	Questions: Mr. Robert M. Poston	160
5.4	Applicability of Object-Oriented Design Methods and C++ to Safety- critical Systems: Dr. Barbara B. Cuthill	163
5.4.1	Questions: Dr. Barbara B. Cuthill	191

6	METHODS FOR REDUCING RISKS IN SOFTWARE SYSTEMS	193
6.1	Automated Tools for Safety-Critical Software: Ms. Anne-Marie Lapassat	197
6.1.1	Questions: Ms. Anne-Marie Lapassat	211
6.2	The Risks of Safety-critical Systems: Dr. Winston Royce	213
6.2.1	Questions: Dr. Winston Royce	216
6.3	Integrated Modeling of Software Cost and Quality: Mr. Kyle Y. Rone and Ms. Kitty M. Olson	219
6.3.1	Questions: Mr. Kyle Y. Rone	224
6.4	Software Reliability for Safety-Critical Applications: Dr. William Everett and Mr. John Musa	225
6.4.1	Questions: Dr. William Everett	227
6.5	Software Configuration Management for Safety-critical Systems: Mr. H. Ronald Berlack	229
6.6	How Much Software Verification and Validation is Adequate for Nuclear Safety?: Mr. Roger U. Fujii	247
6.6.1	Questions: Mr. Roger U. Fujii	254
6.7	Fault-Specific Verification (FSV)--An Alternative VV&T Strategy for High Reliability Nuclear Software Systems: Dr. Lance A. Miller	257
6.8	Certification of Software for Reuse in Safety-Critical Applications: Ms. Charlotte O. Scheper	267
6.8.1	Questions: Ms. Charlotte O. Scheper	274
7	PANEL: APPLICATION OF WORKSHOP TO NRC ACTIVITIES	275
7.1	Presentations	277
7.1.2	Presentation by Dr. John Knight	277
7.1.2	Presentation by Dr. John McHugh	278
7.1.3	Presentation by Dr. Winston Royce	279
7.1.4	Presentation by Dr. Joseph Naser	280
7.2	Questions and Discussion	283
8	PREPARED STATEMENTS	305
8.1	NUSMG Presentation: Mr. Wayne Glidden (presented by Mr. Robert Mullens)	305
9	NRC CLOSING REMARKS	309
9.1	Closing Remarks: Mr. Franklin Coffman	309
9.2	Closing Remarks: Dr. Cecil Thomas	313
10	SUMMARY AND CONCLUSIONS	315
11	REFERENCES	321

APPENDIX A	WORKSHOP AGENDA	323
APPENDIX B	AUTHOR INDEX	329
APPENDIX C	FINAL PARTICIPANTS LIST	331

1 INTRODUCTION

The Digital Systems Reliability and Nuclear Safety Workshop was held on September 13-14, 1993, in Rockville, Maryland. This workshop, sponsored by the United States Nuclear Regulatory Commission (NRC) in cooperation with the National Institute of Standards and Technology (NIST), was co-chaired by Mr. Leo Beltracchi of the NRC and Ms. Dolores R. Wallace of NIST. The workshop provided a forum for the exchange of information among experts within the nuclear industry, experts from other industries, regulators and academia.

The 263 participants came from the District of Columbia, 30 states within the United States, and 10 other countries. While most participants represented nuclear industry vendors, utilities, or the NRC, many came from other power industries (e.g., fossil power) national laboratories, and other U.S. government agencies.

This workshop was designed to respond to the need for a forum on the conversion of nuclear power plant (NPP) safety systems to digital technologies. Many analog hard-wired process control systems and safety systems within NPPs are wearing out and frequently are being replaced with systems using digital technology. There are unique design and safety issues in the development and assurance of safety systems within digital instrumentation and control (I&C) systems. The NRC is developing regulations and guidelines to address these safety issues.

This workshop continued an in-depth exposure of the NRC staff and the nuclear industry to digital systems design safety issues, and also provided feedback to the NRC from outside experts regarding identified safety issues, proposed regulatory positions, and intended research associated with the use of digital systems in NPPs. While this workshop principally addressed the software engineering issues for safety-critical digital systems, the opening speakers provided a system overview and perspective to promote an understanding of the broader issues of safety systems and of the relationship between digital system design issues and the overall development and assurance of safety systems. The intention is that future workshops will focus on the hardware and the human operation components of these systems. The final workshop in the series will provide a forum to discuss a total framework for the development and assurance of all components of digital safety systems for NPPs.

The workshop began with welcoming remarks from Commissioner Kenneth C. Rogers (NRC), Mr. Eric S. Beckjord (Director, Office of Nuclear Regulatory Research, NRC) and Dr. J. Ernest Wilkins, Jr. (Chairman, Advisory Committee on Reactor Safeguards, NRC). These speakers identified the benefits of using digital technology in safety systems while acknowledging the risks and the need for regulation and guidance from the NRC.

The next set of speakers provided regulatory, research, and industry perspectives on digital upgrades. Mr. William T. Russell (Associate Director for Inspection and Technical Assessment, Office of Nuclear Reactor Regulation, NRC) presented the regulatory perspective. Mr. Leo Beltracchi (Senior Project Manager, Office of Nuclear Regulatory Research, NRC) stated the

research perspective. Mr. Richard J. Blauw (Commonwealth Edison Company) and Mr. Paul K. Joannou (Ontario Hydro) provided the industry perspective.

Three technical sessions followed the opening sessions. The first technical session gave an overview of the state of practice and research in the three major components of digital safety systems: hardware, software and human operations. The second technical session focused on the software engineering issues inherent in developing high integrity systems. The third technical session focused on the methods for reducing risks in the software development process.

The final workshop session was a panel presentation followed by an open question and discussion period. The panel consisted of academic and industry experts on the risks of safety-critical digital technology. The NRC asked the experts to address these four questions:

1. Are the proper issues being addressed?
2. What other issues need to be addressed?
3. Are proposed NRC regulatory positions complete and correct?
4. What are the considerations for further research?

Following the panel, the audience asked questions, challenged the panelists' positions, and provided their own opinions. The result was a valuable dialogue on the future use of digital technology within the nuclear industry, and the open regulatory and technical issues.

At the close of the meeting, Mr. Robert Mullens presented a short position paper written by Mr. Wayne Glidden (Duquesne Light Company) on behalf of the Nuclear Utilities Software Management Group (NUSMG). Mr. Franklin Coffman (Chief, Human Factors Branch, Office of Nuclear Regulatory Research, NRC) and Dr. Cecil Thomas (Deputy Director, Division of Reactor Controls & Human Factors, Office of Nuclear Reactor Regulation, NRC) summarized the workshop issues.

2 THE NUCLEAR REGULATORY COMMISSION AND THE ADVISORY COMMITTEE ON REACTOR SAFEGUARDS

In the opening session, Commissioner Kenneth C. Rogers (NRC), Mr. Eric S. Beckjord (Director, Office of Nuclear Regulatory Research, NRC), and Dr. J. Ernest Wilkins, Jr. (Chairman, Advisory Committee on Reactor Safeguards, NRC) welcomed participants to the conference.

Commissioner Rogers observed that while the use of digital technology in I&C systems has proliferated, the existing NRC criteria for analog systems are inappropriate for digital technology. Regulators and industry must apply knowledge from other industries' experiences and from academia about digital I&C systems to the nuclear industry. The nuclear industry has been slow to adopt digital technology, compared with other industries (e.g., avionics, chemical, transportation, and defense). Commissioner Rogers encouraged the nuclear industry to propose and implement, subject to NRC approval, new digital I&C systems. He ended by suggesting that the NRC may have delayed use of digital technology by not having standards and acceptance criteria in place.

Mr. Beckjord called on the NRC to address the public safety issues, form complete and proper regulatory positions, and undertake appropriate research as these digital upgrades occur. While digital systems may be more reliable and perform more functions than analog systems, the purpose of using digital systems is to improve operator performance. Specifically, the use of digital technology should help in I&C systems by reducing the false trip rate, improving process control, and improving the man-machine interface. For example, digital systems have the ability to present information in a more clear and coherent manner, which may help the operator make a quicker and more reliable decision. Some of the techniques used to ensure analog safety systems are inappropriate or need adaptation for digital systems. For example, the principle of diversity as used in analog systems is based on the belief that the elimination of common-mode failure is not possible by quality alone. For software, the effectiveness of diversity is hard to measure and may affect more than the design. Software testing, configuration control, security control, and formal methods need more study and use in the design of digital safety systems.

Dr. Ernest J. Wilkins, Jr. (Chairman, Advisory Committee on Reactor Safeguards, NRC) has been interested in utilization of digital technology for some time and has commented to the NRC on the obvious advantages and disadvantages of digital technology for I&C systems. Dr. Wilkins' position is that the regulations developed before the electronic revolution are inappropriate to the regulation of computerized functions in NPPs. Currently, there is no standard review plan or guide that could help both the Commission and the industry know what is expected of them. Dr. Wilkins affirmed a need for a method to assure correctness of the specification and advocated the use of formal verification and validation (V&V) procedures. Formal V&V procedures assure the correctness of the implementation of the "correct" specification. Dr. Wilkins proposed continuing recruitment by the NRC of additional staff with appropriate digital I&C background to augment current staff capabilities.

2.1 Welcome: Commissioner Kenneth C. Rogers

WELCOMING ADDRESS COMMISSIONER KENNETH C. ROGERS U.S. NUCLEAR REGULATORY COMMISSION

DIGITAL SYSTEMS RELIABILITY AND NUCLEAR SAFETY WORKSHOP

**ROCKVILLE CROWNE PLAZA HOTEL
ROCKVILLE, MARYLAND
SEPTEMBER 13, 1993**

Good morning, ladies and gentlemen. On behalf of the NRC Commissioners and our staff, I am pleased and honored to welcome you this morning to this two day workshop on Digital Systems Reliability and Nuclear Safety. We are pleased to sponsor this workshop in cooperation with the National Institute of Standards and Technology (NIST). This joint cooperation is quite appropriate in view of the long history of NIST initiatives with industry and government related to the development of digital computer systems.

As you know, the primary concern of the NRC is the protection of the public health and safety. A major component of the NRC's defense-in-depth philosophy is the control and safety systems associated with nuclear reactors. Until quite recently, these instrumentation and control (I&C) systems have all been analog systems, and I must say that for several reasons the nuclear industry in the U.S. has been slow in adopting the new digital technology in their plants. However, in part due to lack of available replacement parts for old systems, and in part due to the overwhelming advantages of digital over analog systems, that can no longer be ignored, the regulated community is starting to move toward digitization of nuclear power plant I&C systems at an escalating pace.

With the immediate prospect of increased proliferation of the use of digital technology for these I&C systems, it is essential that suitable performance criteria and standards applicable to these systems be developed and promulgated as soon as possible. The existing criteria, while appropriate for analog systems, are not entirely adequate for digital systems. The sooner new criteria for the digital systems are developed, the sooner we will all have assurance that these systems will perform their designed functions reliably. NRC will then have confidence to approve and license these systems for use and, finally, broader implementation will take place of digital technology in the nuclear industry with concomitant benefits.

There already exists a vast storehouse of information on digital computer systems that can be applied to design issues by the nuclear industry and it behooves the regulators and the industry to fully call upon that knowledge. NRC has co-sponsored this workshop to facilitate the

dissemination and exchange of relevant digital technology information between outside experts and the NRC staff and the nuclear industry. I am pleased to note that representatives of the nuclear, communications and computer industries, academe, national laboratories and the international community, as well as NRC staff members are participating in this workshop and will share their knowledge and expertise with us. I look forward to learning what you have to say.

I do not want to leave you with the impression that the NRC has not been actively involved with these digital I&C issues for some time. We have. NRC has staff with many years of experience in working with and designing digital I&C systems, serving on national and international standards committees, and writing regulatory guides and standards. They have concentrated on specific design questions and issues as they have arisen. I also know that there are many in the U.S. nuclear industry who have taken positive steps toward integrating digital technology into their I&C systems. Some nuclear plants have begun to install digital control systems, such as the Eagle 21, and we hope and expect this usage to accelerate. NUMARC has been developing guidelines to be used by licensees in evaluating I&C digital system upgrades, and NUMARC staff members have been discussing the guidelines with the NRC staff and NRC's Advisory Committee on Reactor Safeguards. However, those actions have not been sufficient. It is up to those in the nuclear industry, where the ultimate nuclear safety responsibilities lie, to propose and, subject to NRC approval, implement the new digital I&C designs that will carry their plants forward into the next century.

Of course we have witnessed the benefits of digital systems in the nuclear industry for many years through the use of full-scope control room simulators. While the first simulators may have relied on analog technology, these devices have evolved over the years to highly complex digital systems that have proven invaluable for operator training and qualification as well as for permitting plant design engineers to test the effects of proposed modifications prior to their actual implementation. However, compared to the chemical, aviation, transportation, and communication industries, and the military, the nuclear industry has been slow in adapting digital technology to its plants.

The potential benefits that can be achieved with digital technology very likely go far beyond what we have been able to see to date. People are beginning to design control systems using fuzzy logic, expert systems, neural networks, and other intelligent technologies. These intelligent control systems can complement the human intelligence of the operators, significantly easing the burden of the operators and improving the safety of nuclear plants. There may eventually be proposed a control system with so sophisticated a human and computer interface that it could result in a kind of symbiosis resulting in a new control partnership between man and machine. Regulators of today would have a great deal of difficulty accepting such a system, but in time, with demonstrated performance, it could be found acceptable.

The Department of Energy (DOE) is supporting research at the Idaho National Energy Laboratory (INEL) on various issues related to advanced digital control technology, and the NRC is doing the same at Oak Ridge National Laboratory (ORNL). The NRC is also participating with the

Organization of Economic Development (OECD) in the Halden Reactor Project in the areas of man-machine interface and advanced I&C systems. It is unfortunate that we do not presently have in the U.S. more man-machine laboratories of the same caliber as exists at Halden. The NRC staff is studying possible ways to enhance U.S. capability in this important area.

NRC itself may have delayed the implementation of digital technology in nuclear plants by not being sufficiently aggressive in developing comprehensive, well established, standards and acceptance criteria that are applicable to digital systems. We intend to improve our performance in this regard. This workshop could be another important step in developing these standards and criteria. That is why you are all here. This meeting should provide an excellent opportunity for the exchange of the latest information relevant to the application of digital technology to the I&C systems used in nuclear reactors. I am pleased to see that we are taking this important step together toward realizing the long overdue benefits of this challenging technology.

Again, welcome. May you have a very fruitful meeting.

2.2 Welcome and Opening Statement: Mr. Eric S. Beckjord

Welcome and Opening Statement
Eric S. Beckjord, Director
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555

Good morning. I am pleased to add my welcome to this Workshop on Digital Systems and Nuclear Safety. The Nuclear Regulatory Commission in cooperation with the National Institute of Standards and Technology is sponsoring the Workshop. In my introduction I will touch on 3 topics: the purpose of the meeting, the opportunities and problems associated with digital control technology for NPPs and finally a fable that I think speaks to this Workshop.

The purpose of the Workshop is to discuss the application of advanced digital systems technology to operations and control of nuclear power plants, now and in the future, in ways that will enhance safety. In particular the NRC expects to hear from experts outside of NRC on safety issues and R&D related to state-of-the-art digital systems in NPPs. In preparation for this Workshop, the NRC has provided staff draft technical position papers on regulatory review of digital systems for replacement of original control systems in operating plants, and in proposed advanced NPPs. We welcome your comment on these papers in the course of the meeting, and we intend to take account of your comments, and the important ideas that arise at the meeting, in making the draft papers final.

The people at this Workshop are internationally recognized experts. You are developers, users, and researchers of digital I&C. You are applying digital I&C in the nuclear industry and other industries including aerospace, telecommunications, defense, software engineering, and universities.

The Workshop provides the opportunity to talk about the introduction of advanced digital systems in future nuclear power plants and the retrofitting of advanced digital systems into currently operating plants. We seek to verify that:

- (1) the proper public safety issues are being addressed,
- (2) the anticipated regulatory positions are complete and proper, and
- (3) the appropriate research needs are being considered.

Whatever the NPP instrumentation and control functions, I believe that digital technology today can do better than the analog technology employed in operating plants. I do not rule out hybrid approaches, and economics will be a factor in making the choices. But there are also realms of application that analog systems cannot enter such as expert systems, fuzzy logic, and neural networks. There are obvious opportunities for these, and many not yet thought of. The question

is whether these opportunities can be realized so that safety will be maintained or enhanced along with the other potential operational advantages. I hope this Workshop will stimulate thinking and dialogue in this direction.

My own view of running nuclear reactors is that the operator must always be in charge as well as responsible for all activities including safety. Accordingly the developments yet to evolve should help the operator to do the job better, as a consequence of the better control and appropriate information being organized and made evident to the operator in real time. One concern that I have, having experienced a catastrophic hard disk deletion on my own PC, is the equivalent, someday, in a high digital technology control room. In the incident I experienced all that was at stake was a few hours of learning how to put programs and files back in place: not a bad learning experience, but certainly not the kind of training tolerable in the NPP.

I turn now to discuss some specific opportunities and problems in application of digital I&C to NPPs. Examples of opportunities include:

1. False trip rate reduction

Testing of independent reactor protection system channels is necessary to assure proper function and reliability. I&C technicians performed this task in original equipment of operating NPPs. Although a simple task in principle, the chance of error is significant in practice, and the result was a high incidence of false trips. Computer controlled testing readily solves the problem and can eliminate false trips from this cause.

2. Improved process control

Water level control in reactor vessels and steam generators can be unstable and cause trips, especially at low power. When water level falls, intuition says to increase feedwater flow. The paradox is that increased feedwater flow increases subcooling, decreases the steaming rate, and can actually cause water level to fall even further. Computer control of water level, by means of thermal-hydraulic models in combination with the usual process variables can provide much more accurate control, and maintain water content within required limits much better than the original analog signal controllers.

3. Improved person-machine interface

Digital computer systems have introduced the concept of "chunking" wherein important process variables are integrated into information and displays to make interpretation clearer and quicker. Chunking does that by collecting and formatting variables to a higher, abstract level, and presenting the information to the operator already, in effect, interpreted. The B&W Pressure-Temperature plot for monitoring operation of the primary coolant system and heat-transfer to the steam generator is an example of chunking.

Examples of problems include the following:

1. Diversity

The elimination of common-mode failure in a safety function is an important goal that system quality alone can not achieve. Diversity together with quality can do so. However, diversity in safety functions is expensive, and I know of no generally accepted measure for the effectiveness of added diversity. The value of diversity in digital systems is more of a problem than in analog systems, because the range of opportunities for employing it is broader, through language, programming, the use of models, etc. I see a need for more attention to the question of how to measure the effectiveness of added diversity.

2. Software Program Testing

Software program testing is necessary to establish that control and advisory functions are performed according to specifications. There are several classes of failure to consider. One is simply the failure to perform a safety function. This is easy to test, by establishing the conditions that require the safety function, and observing the response. A second, and much more difficult class to test for is the unintended function that could cause an unprotected safety problem. The failure of the AT&T telephone system several years ago is an example of an unintended function. A modification of the original program, thought to be minor at the time, introduced the error, and when the specific combination of conditions occurred that activated the fault, the system collapsed.

3. Configuration Control

It is reasonable to expect that there will be modifications from time to time, as a result of experience with early software versions, or a new requirement, or the discovery of an improvement. Configuration control should be exercised to assure that errors will not be introduced as a result of changes. There is also the possibility of discovery of a software logic problem, but that the corrective action is lost or forgotten. The Bruce Unit 4 refueling machine accident was such a case: the end result, some of you may know, was that the program instructed the refueling machine to move down 3 feet while the brakes were engaged. It did move, broke a pipe and caused a small break loss-of-coolant accident which was brought under control. However, such an experience is not welcome.

The security of control programs is an important aspect of configuration control. Obviously it is necessary to preclude the possibility that any person or organization could modify control and protection system programs with malicious intent.

4. Formal Methods

Another concern is that deviations from the design specifications might occur during the development of the software. Procedures are needed to verify that each step of the development is compatible with the intent of the specification. One approach is to employ the mathematical methods to verify and validate software. These methods are formal and have the potential to prove theoretically that the final product is coincident with the design specification.

It is my hope that the opportunities and problems of applying digital I&C technology to NPPs now and in the future will enter the discussions in very productive ways at this Workshop.

I will close with a reading of Aesop's fable about the crow and the pitcher which I think is a story about technological innovation, written about 2500 years ago:

A crow, on the verge of dying with thirst, spied a pitcher in the distance and flew to it with joy. But when he arrived, he discovered to his grief it contained so little water that he could not possibly get at it, despite all his efforts. At one point he decided to turn the pitcher over and break it. However, he was not strong enough to succeed. At last, seeing some small pebbles nearby, he gathered them and dropped them into the pitcher one by one. By this means the water gradually rose to the brim, and he could quench his thirst with ease.

At the risk of using an analog at a digital I&C workshop, I suggest that the living water in the pitcher is the opportunity for introducing digital instrumentation and control in NPPs. Imagination, expressed by Aesop in terms of the crow using the pebbles, is what we need to realize the opportunity.

2.3 Welcome and ACRS Perspective: Dr. J. Ernest Wilkins, Jr.

WELCOME AND ACRS PERSPECTIVE

DR. J. ERNEST WILKINS, JR.³

Chairman of the Nuclear Regulatory Commission's
Advisory Committee on Reactor Safeguards

[as edited from transcript]

Almost exactly one year ago ACRS sent a report to the Chairman of the NRC in which it commented on some of the obvious advantages and disadvantages of digital technology for instrumentation and/or control systems, with particular reference to their implications for nuclear industry regulators. The Committee observed that the NRC staff had, at that time "concentrated its attention on the vulnerability of digital systems to certain kinds of common-mode failures, principally through programming errors introduced into the software." The Committee recommended that the NRC staff revisit some of the less desirable proposed regulatory requirements, augment staff capabilities in the digital technology area, and look at other industries which have dealt with similar problems.

In November of 1992, the Committee responded to the staff's research program defining the environmental qualification requirements needed for digital instrumentation and control systems. Although this research program would ultimately study several environmental features, such as temperature, moisture, and smoke, the staff initially concentrated on electromagnetic radio frequency interference. The Committee recommended that, "The direction of the program be reassessed to account for some kind of risk ordering of a suite of likely stressors," and repeated its earlier recommendation that the staff look for relevant industry experience.

The Committee wrote its most significant letter in this area to the NRC Chairman on March 18, 1993. This letter was the outcome of a series of meetings conducted by the ACRS Subcommittee on Computers and Nuclear Power Plant Operations to explore the regulatory and safety implications of the trends toward digital technology in both existing and proposed nuclear reactors. In these meetings the subcommittee sampled views of industry, the staff, nuclear vendors, and others, both within and outside the nuclear community. Some of the observations contained in that letter follow:

"It is important not to develop a tabloid mentality about new technology."

³This talk gave Dr. Wilkins' personal opinion of the position of the Nuclear Regulatory Commission's Advisory Committee on Reactor Safeguards (ACRS). ACRS has been interested in the utilization of digital technology in nuclear power plants including the specific topics discussed in this workshop.

"One should not regard the various horror stories about catastrophic failures of major computer systems as the norm."

"Computerization provides an opportunity, not a threat."

"It is misleading to bandy failure probabilities about as if they have the same meaning as they do for familiar mechanical and electrical components."

"Formal verification and validation procedures can assure that the code correctly expresses the specifications."

The NRC must be "sure of the requirement, in order to generate verifiable software, for precise no-nonsense attention to the specification of the functions to be implemented."

"The gist of our concerns is that the regulatory procedures developed during the decades preceding the electronic revolution are inappropriate to the regulation of computerized functions in nuclear power plants."

"Neither the staff, nor the Commission, has established what could be described as a standard review plan, or even a regulatory guide, that could help both the staff and the industry know what is expected of them."

"Our recommendation is that a workshop and study with the charter to produce such a plan be commissioned to be done by the National Academies of Science and Engineering."

These are several of the remarks, observations and recommendations of the ACRS over the past year. It is frequently the fate of advice from advisory groups to be received with great applause and then quietly deposited on a shelf to gather dust. ACRS is more fortunate in this regard. The staff and the Commission have made sincere efforts to respond to some of the ACRS suggestions. This is not a sycophantic acceptance; the staff has changed its views on appropriate regulations in significant ways. Ultimately, the staff may propose rules and regulations that the Committee can enthusiastically endorse to the Commission.

The staff has tried to recruit new employees who have a suitable background in digital technology instrumentation and control for the purpose of augmenting the staff's capabilities in these areas. This effort has not been as successful as ACRS wished, partly because individuals with a background in this area are in great demand everywhere and partly because the financial exigencies facing Government agencies have made recruiting difficult.

Finally, the staff has endeavored to solicit assistance and information from external sources. It has had meetings with its regulatory counterparts in other countries and has benefitted from the exchanged of information. It also holds frequent discussions with the nuclear industry,

particularly with Nuclear Utilities Management and Research Council (NUMARC) and to some extent the staff has sought information from individuals outside the nuclear industry.

Although the staff did not follow the Committee's advice to commission the National Academies to hold a workshop, it did organize the present workshop in cooperation with the National Institute of Standards and Technology and has assembled in this room a significant group of outside experts. Because the purposes of the present workshop are less ambitious in comparison with the interests and concerns of the Committee, the attendees need to keep the larger picture in mind also.

3 ISSUE PERSPECTIVES FOR NUCLEAR POWER PLANTS

In this session, Mr. William T. Russell (Associate Director for Inspection and Technical Assessment, Office of Nuclear Reactor Regulation, NRC), Mr. Leo Beltracchi (Senior Project Manager, Office of Nuclear Regulatory Research, NRC), Mr. Richard J. Blauw (Commonwealth Edison Company), and Mr. Paul K. Joannou (Ontario Hydro) provided regulatory, research, and industry perspectives on digital upgrades.

Mr. Russell covered highlights of NRC positions intended to facilitate ongoing reviews. The reviews include those for the advanced boiling water reactors, the Combustion Engineering System 80+, and retrofits on operating reactors. He provided a basis for evolving positions on quality and diversity, he and described a block approach for performing diversity assessments. The NRC staff recognizes the potential for enhanced safety and reliability that digital systems bring to the nuclear industry. The staff also recognizes the challenges to safety that are unique to digital systems implementation. An item used in the process for assessing quality of advanced reactors is the Design Acceptance Criteria (DAC). The NRC specifies top-level system requirements and a detailed design process from system performance requirements through V&V. The process is broadly based on IEC 880 [IEC880]. This covers the approval of the design process, not the approval of the actual design of the system.

The four major elements, according to Mr. Russell, of the NRC's diversity position are the following:

1. The applicant shall assess the defense-in-depth and diversity of the proposed I&C system to demonstrate that vulnerabilities to common-mode failures have been adequately addressed.
2. In performing the assessment, the vendor or applicant shall analyze each postulated event in the analysis section of the safety analysis report (SAR) using best estimate methods. The vendor or applicant shall demonstrate adequate diversity within the design for each of these events.
3. If a postulated common-mode failure could disable a safety function, then a diverse means, with documented bases that the diverse means is unlikely to be subject to the same common-mode failure, shall be required to perform either the same function or a different function. The diverse or different function may be performed by a non-safety system if the system is of sufficient quality to perform the necessary function under the associated event conditions.
4. A set of controls located in the main control room shall be provided for system level actuation and control of critical safety functions. The displays and controls shall be independent and diverse from the safety computer system identified in Items 1 and 3.

Mr. Beltracchi presented an overview of the NRC research activities and stressed the need to define a technical basis for digital system requirements. A technical basis comprises the following:

1. A requirement has been clearly coupled to safe operations.
2. The scope of the requirement is clearly defined.
3. A substantive body of knowledge exists and the preponderance of evidence supports a technical conclusion.
4. A repeatable method exists for correlating relevant characteristics with performance.
5. A threshold for acceptance can be established.

Mr. Beltracchi led the audience through the history of NRC's regulations, which included an analysis of standards used within the nuclear industry. Only two standards cite digital systems. Mr. Beltracchi also identified two major regulatory areas of digital upgrades for the NRC to address: the diversity requirements for safety algorithms and computer unique requirements. He presented an outline for a framework for an NPP safety system. When completed, this framework would provide guidance in organizing the digital system requirements for the hardware, software and the human operator components.

Mr. Blauw voiced a concern in his presentation that while the top-level view or design of digital systems may appear simplistic, the implementation of safety regulations is complex. The nuclear industry must look at other industries for process control and monitoring. He will be working on the revision of American Society of Mechanical Engineers (ASME) standard on Nuclear Quality Assurance (NQA), Part 2.7 [ASMENQA2] to explain differences between design verification and V&V, differences between configuration control and configuration management (CM), and documentation issues. Mr. Blauw described his experiences with implementing digital systems in which the concern was the cost-effectiveness of providing the necessary assurance of the safety of these systems. Mr. Blauw then described the IEEE Standard 7-4.3.2 "Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations" [IEEE7432]; utilities were involved in developing this standard⁴. This standard is intended to be used with IEEE Standard 603 "Standard Criteria for Safety Systems for Nuclear Power Generating Stations" [IEEE603].

He also indicated that the Nuclear Utilities Management and Resources Council (NUMARC) will publish a digital upgrade guideline which recommends a licensing approach, and that the Electric Power Research Institute (EPRI) will publish other guidelines. The standards and guidelines

⁴This standard was officially approved by the IEEE Standards Board at their meeting on September 15, 1993.

address some principal issues for system design models including project management, configuration control, failure and error analysis management, and independent review. Currently, a major problem is that plant drawings and other forms of design documentation are not under configuration control and are frequently incorrect with respect to the current plant configuration.

Ontario Hydro of Canada has vast experience in applying digital system technology in a NPP. Mr. Joannou described some of the issues encountered while licensing the Darlington Reactor Station, which used a fully computerized shutdown system. One problem was lack of a widely-accepted definition of "good enough" software; the deficiency led to joint development between Ontario Hydro and Atomic Energy Canada Limited (AECL) of a family of software engineering standards, guidelines and procedures for NPP protective, control, and monitoring software. Major issues addressed by AECL and Ontario Hydro include the reviewability of the software, safety functions, ambiguities in requirements specifications, software reliability and software maintainability. One of the problems is that the use of software analysis techniques, such as reengineering and hazard analyses, is costly. The industry needs to develop cost-effective analysis methods. The new standards and guidelines developed during this experience provide rules for documentation, test types (statistically valid, trajectory-based random tests, systematic tests), software CM (SCM), audits, qualifications of personnel and independent V&V.

REGULATORY PERSPECTIVE ON DIGITAL INSTRUMENTATION AND CONTROL SYSTEMS FOR NUCLEAR POWER PLANTS



**William T. Russell
Associate Director for
Inspection & Technical Assessment**

The NRC staff recognizes the potential for enhanced safety and reliability that digital systems bring to the nuclear industry.

The staff also recognizes the challenges to safety that are unique to digital systems implementation.

ADVANTAGES

- **Safety (reduced operator errors)**
- **Stability, no drift**
- **Accuracy**
- **Reduced susceptibility to S/N effects**
- **Multiplexing, fewer cables**
- **Self diagnostics**
- **Low power requirements**
- **Cost**
- **Parts availability**
- **Graphic displays**
- **Fault tolerance/avoidance**

DISADVANTAGES

- **Complexity**
- **Requires extensive V&V**
- **Subtle failure modes**
- **Complex testing, trouble shooting**
- **Susceptibility to common mode failure**

COMMON MODE FAILURE

- **Identical hardware/software in redundant channels**
- **Software reliability cannot be quantified**
- **Software errors may result in a common mode failure**
- **Compensated by quality and diversity**

REGULATORY REVIEW BASES FOR ADVANCED REACTORS

- **Existing regulatory requirements**
- **Hardware/software quality and diversity**
- **EPRI Utility Requirements Document**

DIVERSITY

Requirement for diversity based on:

- **Discussions with experts**
 - N. Leveson (University of Washington)
 - B. Littlewood (City University, London)
 - D. Parnas (Queen's University, Canada)
 - J. Rushby (SRI International)
- **Discussions with outside organizations**
 - NASA, NIST, IBM, CSC, TRW, SEI, Siemens
- **Discussions with foreign regulatory agencies**
 - NII (United Kingdom), AECB (Canada), DSIN/IPSN (France)
- **NRC Digital System Reliability Workshop**

QUALITY

- **Hardware/software integrated development process - industry standards and EPRI requirements**
- **Addresses hardware/software change process over life of plant**
- **Top level design and process requirements subject to NRC inspection**

DIVERSITY POSITION

- 1. The applicant shall assess the defense-in-depth and diversity of the proposed instrumentation and control system to demonstrate that vulnerabilities to common mode failures have been adequately addressed.**

DIVERSITY POSITION (cont'd)

- 2. In performing the assessment, the vendor or applicant shall analyze each postulated event that is in the analysis section of the safety analysis report (SAR) using best-estimate methods. The vendor or applicant shall demonstrate adequate diversity within the design for each of these events.**

DIVERSITY POSITION (cont'd)

- 3. If a postulated common-mode failure could disable a safety function, then a diverse means, with documented bases that the diverse means is unlikely to be subject to the same common-mode failure, shall be required to perform either the same function or a different function.**

The diverse or different function may be performed by a non-safety system if the system is of sufficient quality to perform the necessary function under the associated event conditions.

DIVERSITY POSITION (cont'd)

- 4. A set of ~~safety-grade~~ controls located in the main control room shall be provided for system-level actuation and control of critical safety functions. The displays and controls shall be independent and diverse from the safety computer system identified in Items 1 and 3.**

DIGITAL RETROFITS

- **Reactor trip systems**
 - Eagle 21 (Westinghouse)
 - Spec 200 micro (Foxboro)
- **Radiation monitoring systems**
 - NUMAC (GE)
 - Others under 10 CFR 50.59
- **Diesel generator load sequencers**
 - PLC-based (Allen Bradley)
- **Auxiliary feedwater control system**
 - Woodward
- **Plant safety monitoring system**
 - Westinghouse

OPERATING REACTOR DIVERSITY

- **Assessed based on Items 1, 2, and 3
of advanced reactor position**
- **Applied to RPS and ESF retrofits**
- **Example - Reliance on ATWS
mitigation system to handle loss of
RPS**

HOW IS DIVERSITY ASSESSMENT PERFORMED?

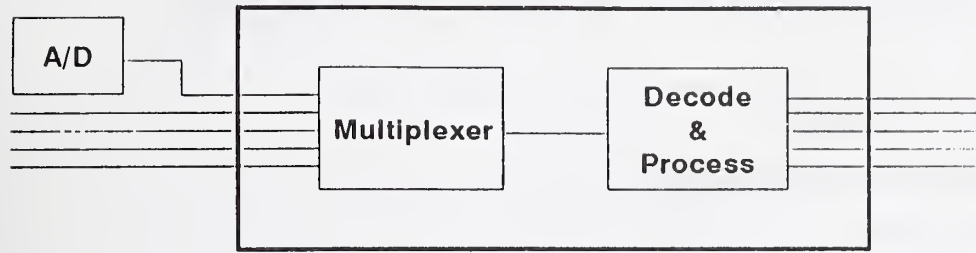
- Analyze I&C system function in design basis events
- Identify common functional elements (blocks)
- Postulate failure of similar blocks
- Evaluate system response using best-estimate methods

WHAT ARE BLOCKS FOR PURPOSES OF DIVERSITY ANALYSIS?



- A to D Converter - Not considered a block in diversity analysis when:
 - Single input/output
 - Simple, can be fully tested
 - Extensive operational history

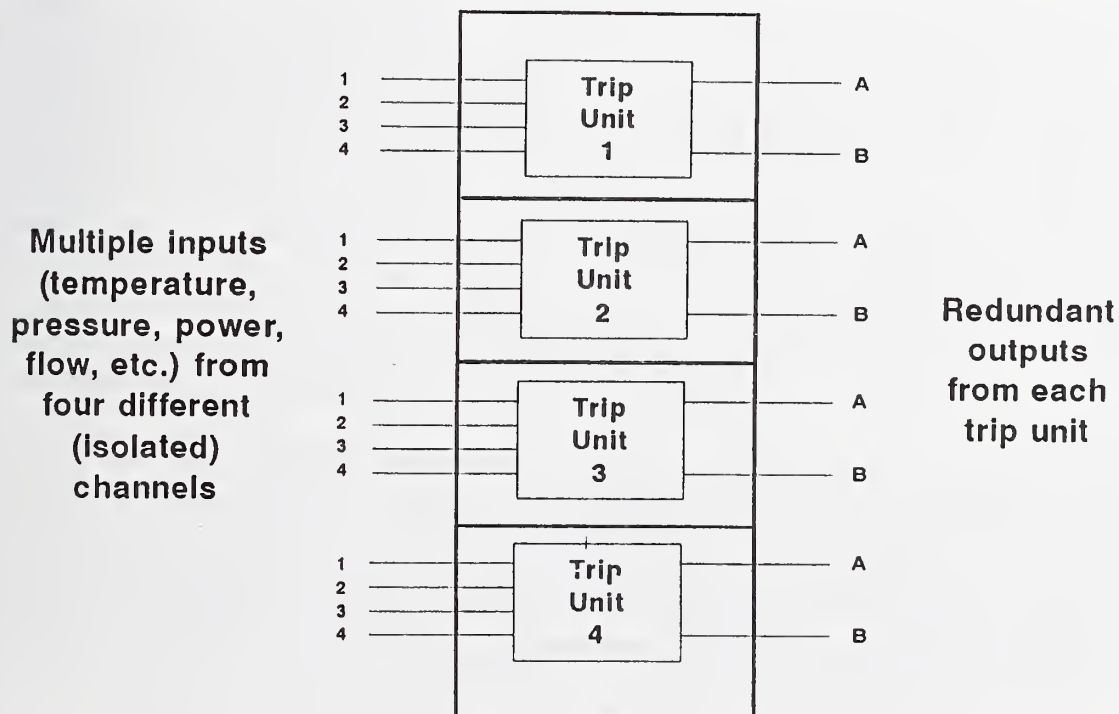
MULTIPLEXER SYSTEM



Block

- Considered a block for diversity analysis when:
 - Multiple inputs / multiple outputs
 - Complex, cannot be fully tested
 - Limited operational history

TRIP UNIT SYSTEM



DIVERSITY EVALUATION

- Review scope
 - Software language
 - Hardware
 - Function
 - Signal
 - Design (including design team)
- Diverse if:
 - All above are different
 - Different function with same software language and same vendor
 - Different vendor with same function
- Case-by-case review for other implementations

3.2 NRC Research Activities: Mr. Leo Beltracchi

NRC RESEARCH ACTIVITIES

Leo Beltracchi
U.S. Nuclear Regulatory Commission
Washington, DC 20555

1.0 ABSTRACT

This paper identifies and describes safety issues related to the design, development, and qualification of reliable digital computer systems for nuclear power plants. It also describes the U.S. Nuclear Regulatory Commission's research program on these issues. The paper discusses an evaluation of the initial standards for hard-wired based safety systems. The lessons learned in developing these standards provides guidance in the design and use of digital technology in nuclear power plants. Also, this evaluation discusses how the content of the standards should lead to a framework of design criteria and the related acceptance criteria that can be used for computer-based safety systems. The opinions and viewpoints expressed herein are the author's personal ones and they do not necessarily reflect the criteria, requirements, and guidelines of the U.S. Nuclear Regulatory Commission (NRC).

2.0 INTRODUCTION

Analog, hard-wired technology is the dominant technology for nuclear power plant instrumentation and control systems and safety systems within the United States. The design, development, and qualification of safety systems is governed by General Design Criteria (10 CFR 50, Appendix A), published by the NRC, and by industry developed standards (See Section 6.1, Standards). The General Design Criteria present top level requirements and are generally independent of implementation technology. However, the standards amplify these top level requirements and contain requirements and guidelines reflective of safety issues associated with the use of hard-wired technology. Digital technology will replace ageing hard-wired technology in nuclear power plant safety systems; standards should be revised to reflect safety issues associated with the use of digital technology. This paper describes several of the safety issues associated with the design and development of computer-based, safety systems.

There are many unique design and safety issues for digital systems. For example, digital systems operate in a discrete fashion whereas analog systems operate in a time-continuous fashion when executing a function. Thus, software execution time in computing a safety function is an important design performance issue for digital systems. Also, computer programs are more difficult to test than analog hard-wired systems. For example, tracing a sensor signal through a computer program is usually more difficult than signal tracing through hard-wired systems.

Furthermore, the loss of a safety function by common cause failure due to environmental factors must also be considered.

Because of the difficulty in establishing reliable stored logic, digital systems are usually more complex than analog systems. Much of this difficulty stems from the lack of a systems engineering approach in the design of digital systems. Leveson and Turner's (1993) analysis of the Therac-25 Accidents concluded:

"Accidents are seldom simple - they usually involve a complex web of interacting events with multiple contributing technical, human, and organizational factors."

"The problem of accidents in complex systems must be approached from a system engineering point of view and all possible contributing factors considered and handled."

The safety concern is that an inadequate design process can lead to errors in the final product, which may lead to the loss of a safety function.

A computer-based, safety system consists of hardware and software to implement the desired safety functions. There are also human interfaces from which humans monitor operation and perform maintenance on the system. Several of these elements are discussed next in the context of a need for a framework of design criteria for computer-based, safety systems.

3.0 DISCUSSION

3.1 Standards

A standard encodes a body of knowledge and accepted practices. The development of industry standards for the design of nuclear power plant safety systems first began about the time the NRC published General Design Criteria (10 CFR 50, Appendix A). Section III, "Protection and Reactivity Control Systems," Appendix A, 10 CFR 50 identifies ten criteria for the design of protection safety systems. These criteria identify the basic performance requirements and design principles for a protection system and are generally independent of the implementation technology. In order to expand on the General Design Criteria and identify specific design requirements, the nuclear industry developed standards, such as IEEE Standard 279-1971 and IEEE Standard 308-1971 (R1980). See Section 6.1, Standards, for the title of each of these standards and subsequent standards defined in this paper. IEEE 279 defines specific requirements for the design of protection systems, while IEEE 308 defines specific design requirements for Class 1E electrical systems. A reliable source of electrical power is necessary to operate the protection system.

In reviewing these standards, it is important to understand the definitions used for terms in these documents. IEEE Standard 308-1980 identifies a safety system as follows:

"Those systems (the reactor trip system and an engineered safety feature, or both, including all their auxiliary supporting features) which provide a safety function."

While this appears to be a reasonable definition, it is incomplete. A human system is also necessary as part of the safety system. Operators and maintainers are key elements of the human system necessary to manually initiate a safety action, monitor, adjust, and maintain a safety system. The data in Table A-2.5, Reactor Scram Signals, (NUREG1272, 1992) indicates operators are significant initiators of manual reactor scrams. No safety system is able to operate for the life of the plant without the support of a human system. A better definition of a safety system is then:

Those systems (the reactor trip system and an engineered safety feature, or both including all their auxiliary supporting systems and a human system) which provide a safety function.

The author reviewed IEEE Standard 279-1971, IEEE Standard 308-1971, and ANSI N18.8-1973. ANSI N18.8 identifies requirements for the preparation of a design basis for systems that perform protective functions; however, it was never published as a standard. Furthermore, it appears to be an overview type of standard for the design of a plant. In fact, ANSI N18.8 was developed after (in 1973) the aforementioned standards. Concerning design basis, ANSI N18.8 states that:

"...safety systems and their auxiliary systems shall be adequate to assure that events caused by a station transient, a failure, an act of nature, or accidental act do not produce effects that will prevent the degree of control over the containment or movement of radioactive material that is deemed acceptable for the event."

Clearly, the safety system must be tolerant of hazards to ensure the successful operation of the safety function.

While ANSI N18.8 was never published as a standard, its contents were integrated into earlier versions of ANSI/ANS 51.1-1988 and ANSI/ANS 52.1-1988. Consideration of design basis accidents is necessary prior to the design of a reactor trip system. Analysis of the design basis accidents is necessary to establish the performance requirements of the reactor trip system to maintain safety functions. Thus, ANSI N18.8-1973 logically should have preceded IEEE Standard 279-1971. This is a lesson learned from the review of previous standards; that is standards should include a top-down prescription. Another lesson learned is the need to specify the role of the human system as part of the safety system in the design process.

ANSI/ANS 50.1-Draft 6 is a draft standard that will eventually replace both ANSI/ANS 51.1-1988 and ANSI/ANS 52.1-1988. The new elements in the draft standard reflect an increment

in the knowledge base when compared to ANSI N18.8. ANSI/ANS 50.1-Draft 6 sets design requirements for safety grade and non-safety grade equipment. In addressing the overall safety design criteria, the document discusses six elements:

- 1) a general approach,
- 2) deterministic analysis,
- 3) probabilistic risk analysis,
- 4) industry codes and standards,
- 5) safety analyses, and
- 6) design criteria for specific plant systems.

These elements form the start of a framework for design criteria. However, the framework does not address the use of digital computers and it barely addresses the human system.

The six elements in ANSI/ANS 50.1-Draft 6 define a two step approach to design. The first five elements define potential safety issues and a design basis for the plant. The sixth element, design criteria for specific plant systems, defines unique requirements for plant systems such as the reactor protection system and these requirements define the second step in the design process. The requirements are stated in the form of a standard. IEEE Standard 603-1980, which replaces IEEE Standard 279-1971, identifies requirements necessary to design a reactor protection system.

A short discussion of some of the safety issues associated with the design of computer based, safety systems in the context of ANSI/ANS 50.1-Draft 6 type of standard elements follows next. These issues include: a possible need to identify diverse means of achieving a safety function, time response requirements of the safety system, and fault tolerance requirements.

A highly likely source of a common software error is a poor design process (Neumann, 1992). The most challenging part of the design process is to specify a complete set of requirements for a system. The problem becomes, what is the acceptance criterion for a complete set? A possible acceptance criteria is the operating history of the developed and installed system. However, this represents trial and error, which is unacceptable for nuclear safety. The most likely acceptance criterion for the completeness of a set of requirements is engineering judgment.

Leveson (1993) advocates the use of hazard analysis as part of the system design process in the use of digital computers for high integrity applications. System hazard analyses should be conducted at the start of the design process. The goal of hazard analysis is to identify the weak points in the system design and then to specify fault tolerant response(s) for implementation in the computer system. The hazards analysis serves an important role to identify threats to and potential failures of the safety system. Also, a hazard analysis may be a preliminary step to a probabilistic risk analysis (PRA). Furthermore, hazard analyses and PRA have a common goal in detecting and responding to potential operational faults.

One form of risk that is difficult to assess is the adequacy of the system design process. How does a designer measure the completeness of the set of design requirements to meet the stated

goals of the safety system? The omission of a key design requirement for a safety system may result in the loss of a safety function. In an on-going NRC sponsored study (Personal communication with Mr. Carl Johnson, NRC) on common cause failure event cause, it was found that 54 percent of these failures of plant hardware components were due to design or installation faults. Another 30 percent of the causes were due to test and maintenance faults.

In an unrelated study, Mr. Fujii (1993) reports that for small systems, 53 percent of the software errors were caused by an improper understanding of the interaction between the system and the software design. Furthermore, for large systems, the number jumps to 63 percent. These figures resulted from the analysis and examination of software errors in complex command and control, avionics, and critical medical control systems. Although these errors were identified from diverse, non-nuclear applications, the results of this study are very similar to the result from the on-going NRC sponsored study discussed earlier. An important lesson from these studies is the need for a systematic, rigorous effort in establishing design requirements to minimize errors in the final product.

In summary, fault tolerance requirements for a computer-based, safety system should be developed from hazard analyses and PRA studies. Also, to minimize the risk of design error, a systematic, rigorous effort is necessary in establishing and verifying design requirements.

The use of digital technology in safety systems provides an opportunity to compute safety functions directly from monitored plant parameters. For example, one critical safety function is to maintain a cool reactor core. A cooled core maintains the geometry of the core and of the passageways for the insertion of control rods. Insertion of the control rods are necessary to shut down the reactor upon threat to a critical safety function. A measure of core cooling is the Departure from Nucleate Boiling Ratio (DNBR). An algorithm for DNBR would include coolant temperature, pressure, coolant flow, and reactor power as input data. A trip set point is also necessary.

A functionally diverse measure of core cooling would be hot leg subcooling. The subcooling of the hot leg coolant is determined from coolant temperature, pressure, and saturation temperature, which is a function of pressure. A subcooling trip set point and response time must also be specified through scenario analysis to establish the limiting design basis event. The use of a functionally diverse means of achieving a safety function should reduce the risk to a common cause design error; however quantifying the risk reduction may be difficult.

A framework of design criteria for a computer-based, safety system should contain a time response performance requirement. A time response requirement is necessary because a computer requires a finite amount of time to process stored safety logic but the system must react in sufficient time to maintain the safety function. The analysis of limiting design basis events develops important performance data for each safety function in the design of computer-based, safety systems. The limiting design basis events help to determine the time response requirements of a safety system. The time response of the safety system must then be divided and allocated to sensor response time, computer response time to calculate the safety algorithm,

and the response time of the actuator. The specification of computer response time is an important parameter for the performance of the processor. The processor must complete the execution of the program within the allotted time.

In summary, deterministic analyses are necessary to identify and document each safety function, diverse means of achieving a safety function, time response of the safety system, and the sensor data necessary to implement each function.

ANS-50.1-Draft 6 identifies basic design requirements for a nuclear power plant; it does not identify design requirements for plant systems. However, it does identify other standards that contain requirements for the design of plant systems. For plant safety systems, it identifies IEEE Standard 603-1980 and IEEE Standard 308-1980 as standards containing system specific requirements. These standards do not contain requirements for computer-based safety systems.

One standard that addresses application criteria for computer-based safety systems is IEEE-7-4.3.2-1993. This standard was developed to augment IEEE Standard 603-1980 because of the uniqueness of software in computer systems. The standard contains requirements for software development, hardware-software integration, computer system validation, and verification. IEEE-7-4.3.2 also identifies criteria in specifying requirements for computers used as part of a safety system. Moreover, it specifies computer specific requirements to meet the criteria of IEEE Standard 603-1991. IEEE Standard 603-1991, IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations, provides system level hardware criteria for safety systems.

IEEE 7-4.3.2 endorses the use of IEEE software standards. One purpose of these software standards is to control the development process and thereby minimize the potential for error in the application software. In developing software for a safety application, a designer should select and use standards to cover all elements of the development process such as requirements analysis, design, and test. Furthermore, project type standards for quality assurance, verification and validation, and configuration control are also identified and should be used to minimize the potential for error in the final product.

ANSI/ANS 58.8(Draft, Revision of 1984 Standard) establishes time response design criteria for safety-related operator actions. The criteria are used to determine the minimum response time intervals for safety-related operator actions, such as manually initiated reactor trip. The draft standard also contains general guidance for instrumentation and controls necessary to support safety-related operator actions. However, this draft standard is not directly linked to safety system design standards, e.g. IEEE Standard 603-1980 and ANSI/ANS 50.1, Draft 6.

The human system is not included in the plant system in ANS 50.1-Draft 6. One reason for not identifying this system may be because the human system is one of the most flexible systems in the plant. However, the human system is an important support system necessary for the successful operation of the plant's safety system. Much of the information necessary to generate emergency operation procedures comes from the design analyses for safety systems. Safety-related operator actions for interacting with safety systems must be specified during the design

process as part of the effort in establishing requirements for safety systems. ANSI/ANS 58.8 (Draft, Revision of 1984 Standard) is a first step in this direction, but it must be integrated with the appropriate design standards for safety systems.

In summary, Figure 1 shows the relationship among the major standards for the design of safety systems. ANSI N18.8(1973, no longer valid), ANSI/ANS 51.1-1988, ANSI/ANS 52.1-1988, and ANSI/ANS 50.1, Draft 6 define requirements for the design basis of safety systems. IEEE Standard 308-1972(R1980) defines requirements for Class 1E electrical systems necessary to operate safety systems. ANSI/ANS 58.8(Draft, November, 1992) establishes time response design criteria for safety related operator actions. However, this standard is not integrated with and cross referenced to the other standards identified in Figure 1. Finally, IEEE Standard 603-1980 identifies requirements necessary to design a reactor protection system. ANSI/IEEE-ANS 7-4.3.2-1982 and its successor IEEE Standard 7-4.3.2-1993 is the only standards that identify design requirements for the use of digital computers in safety systems. However, these standards do not identify all of the requirements necessary for the design of a computer-based, safety system. Based on NUREG/CR-5930 and the review of ANSI/ANS 50.1- Draft 6, it appears that standards for the design of safety systems need to include computer unique design requirements. Because many standards and disciplines are involved it is important to establish a framework of design criteria to ensure a reasonable degree of completeness in the specifications.

3.2 Outline Of A Framework Of Design Criteria

The outline of a framework of design criteria for a computer-based, safety system in Figure 2 is based on the review of standards and lessons learned. This outline is a functional version of the standards presented in Figure 1, with some additional detail. The goal of the example system identified in Figure 2, the reactor trip system, is to operate the NPP safely. The design basis must identify all functions performed by the safety system. These functions must then be allocated to the individual systems within the safety system. Furthermore, deterministic analyses would be performed to identify the limiting design basis events.

Plant design basis events would also be identified, analyzed, and documented. The plant events analyzed could include system and component failures as well as challenges from environmental hazards. Also, emphasis could be placed on identifying vulnerabilities and environmental limitations to develop acceptance criteria for qualifying the hardware of digital systems. The susceptibility of digital systems to electromagnetic interference (EMI) and radio-frequency interference (RFI) is a major concern. Furthermore, the operational history of safety systems and the challenges to safety systems could also be considered (NUREG 1272, 1992). In summary, a plant hazard analysis and then a probabilistic risk analysis identify threats to safe operation. Fault tolerance and response requirements to hazards could be determined and specified as part of the design basis.

Figure 3 presents a conceptual model of a system design and development life cycle based on the discussions in this paper. Once system requirements and functions have been established, they must then be allocated to hardware, software, and humans. After completion of this step,

the next efforts for software development could be detailed design, coding, unit tests, and unit integration followed by software test and validation. Similar efforts could also be done for hardware development and for the human system. Environmental qualification of the hardware could be an important step in the development process. The design requirements for the human interface will impact the software design and the hardware design, and this relationship is not illustrated in Figure 3. The integration of the hardware, software, and human interfaces could be necessary to construct the system. Once assembled, the system could be then subjected to test and validation in response to pre-established system requirements.

Not shown in Figure 3 are the verification activities for the various steps in the design and development life cycle. Effective verification activities early in the design and development life cycle minimizes the resources that would be needed to detect and correct problems later in the life cycle. Furthermore, the lessons learned from the verification and validation activities provide important information for updating the standards and guidelines used in the design and development process.

In summary, the design and development process for digital systems consists of many activities and the use of many standards. To minimize the potential for errors in these activities, consideration could be given to the development of an overview standard.

4.0 RESEARCH PROGRAMS

4.1 Hardware Programs

Most operating plants in the U.S. contain instrumentation and control (I&C) systems that were designed over 25 years ago. As these systems age, maintenance and support costs increase due to obsolescence, lack of original equipment support, and increased testing requirements. On the other hand, major advances in the electronic industries have produced products that were never envisioned during the original design process of nuclear power plants. In order to benefit from these evolving technologies, the NRC initiated two research programs to perform confirmatory research, and develop a technical basis for acceptance criteria for hardware qualification of digital I&C systems which will be used in existing nuclear power plants and in the proposed new plants.

First, under the auspices of the NRC, the Oak Ridge National Laboratory (ORNL) is conducting a study with a view to identify functional and environmental issues arising from the application of new technologies to the instrumentation comprising the next generation of nuclear power plants. The purpose of this program is to develop an understanding of the technical issues involved in evaluating long-term properties of advanced digital instrumentation and control systems. Emphasis has been placed on identifying vulnerabilities and environmental limitations that could be imposed on microprocessor-based systems in nuclear environments.

Second, ORNL is developing a technical basis for evaluating the susceptibility of digital systems to electromagnetic interference (EMI) and radio frequency interference (RFI). IEEE Standard 1050-1989 was found for the most part to do an adequate job of specifying electromagnetic

compatibility design and installation practices that are applicable to nuclear power plant environments. Relevant military standards are MIL-STD-461C and MIL-STD-462. These standards were found to be reasonable starting points from which to begin an evaluation of relevant test criteria and methods for nuclear power plant applications. The results from this study are currently under internal review.

In summary, the objectives of these programs are to:

- * To develop regulatory guidance on susceptibility to EMI and RFI,
- * To develop regulatory guidance on the qualification of digital I&C hardware.

4.2 Software Programs

A clear need exists for standards and a technical basis for acceptance criteria for the use of digital computers in safety systems. There is not yet however, a clear consensus on the safety issues and the technical basis for their resolution in the area of digital technology. Generally, a technical basis exists when:

1. The topic has been clearly coupled to safe operations.
2. The scope of the topic is clearly defined.
3. A substantive body of knowledge exists and the preponderance of the evidence supports a technical conclusion.
4. A repeatable method to correlate relevant characteristics with performance exists.
5. A threshold for acceptance can be established.

Establishing a technical basis for the use of computer-based systems could be a significant, time consuming and expensive effort.

One means by which part of the technical bases are being developed is by the NRC's participation in the Organization for Economic Cooperation and Development (OECD) Halden Reactor Project. The OECD Halden Reactor Project is one of Europe's largest experimental laboratories conducting research on fuels, materials, man-machine interfaces, and advanced instrumentation and control systems. One area of interest to the staff is the Halden Project's research on the use of formal methods and theorem provers for the design of computer-based safety systems. Another area of interest is the research on the effectiveness of various software test techniques (Dahll, Barnes, and Bishop, 1990).

The NRC is also conducting other research to establish the technical basis for guidelines and acceptance criteria on the use of digital computers in nuclear power plant safety systems. The objectives of some of these programs are as follows:

- * To identify and document the positive and negative attributes resulting from the use of standards and computer aided software engineering (CASE) tools when used in the

design, development, evaluation, and certification of high integrity software for nuclear power plant safety systems,

- * To evaluate the feasibility of (Phase A) and develop and test (Phase B) a prototype CASE tool for assessing the degree of functional diversity within software safety systems,
- * To independently evaluate, test, and improve guidelines for use in the audit of computer-based, safety systems,
- * To develop system classification guidelines and qualitative reliability measures,
- * To develop and document guidelines for verifying and validating expert systems,
- * To review and assess software languages for use in nuclear power plant safety systems,
- * To assess how digital technology changes human actions and error rates, systems unavailability, and core damage frequency; and to improve methods for analyzing this human performance in PRAs.

These programs are starting to produce useful products. A survey and assessment of conventional software verification and validation methods has been published (NUREG/CR-6018). Also, an assessment of standards and guidelines for high integrity software has been published (NUREG/CR-5930). The results from these studies are helping to identify a need to clarify the "safety system" versus "software" issues and to establish a framework of design criteria for computer-based, safety systems. The NRC is now in the process of formulating additional research with the objective of identifying and documenting a framework of design criteria. The first step in this direction is the integration of research products to develop the technical basis for regulatory positions on software. This effort will survey the existing research programs within the NRC and other industries and integrate the relevant products into a matrix of requirements versus technical basis. The end product will be to develop and document the technical basis for regulatory use. A second step is necessary to refine the technical basis for computer-based, safety systems.

5.0 CONCLUSIONS

The design and use of digital computers to perform safety functions within nuclear power plants must be guided by guidelines, standards, and acceptance criteria. While the existing set of standards provide useful information, they do not provide all of the requirements necessary for the use of digital computers. While some effort exists to revise standards, such as the effort that resulted in IEEE 7-4.3.2 Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations being updated in 1993, additional effort could be needed to revise existing standards to incorporate the use of digital computers. To ensure a comprehensive approach to the development of standards and regulatory guidelines, a need exists for a

framework of design criteria. The framework could include a requirement for a systematic approach to define, classify, and allocate functions to hardware, software, and humans. Finally, a technical basis could also be necessary for the computer unique requirements and guidelines to support the framework and to provide acceptance criteria.

6.0 REFERENCES

6.1 Standards

American National Standards Institute, ANSI N18.8, October, 1973, Criteria for Preparation of Design Basis for Systems that Perform Protective Functions in Nuclear Power Generating Stations (Trail use and comment).

American National Standards Institute/American Nuclear Society, ANSI/ANS 51.1-1988, Nuclear Safety Criteria for the Design of Stationary Pressurized Water Reactor Plants.

American National Standards Institute/American Nuclear Society, ANSI/ANS 52.1-1988, Nuclear Safety Criteria for the Design of Boiling Water Reactor Plants.

American National Standards Institute/American Nuclear Society, ANSI/ANS 50.1, Draft #6, Nuclear Safety Design Criteria for Light Water Reactors, January 1993.

American National Standards Institute/American Nuclear Society, ANSI/ANS 58.8-1984, Time Response Design Criteria for Safety-Related Operator Actions.

American National Standards Institute/American Nuclear Society, ANSI/ANS 58.8, Draft, Time Response Design Criteria for Safety-Related Operator Actions, November, 1992.

American National Standards Institute/Institute of Electrical and Electronics Engineers-American Nuclear Society, ANSI/IEEE-ANS-7-4.3.2-1982, Application Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear Power Generating Stations.

Institute of Electrical and Electronics Engineers, IEEE Standard 279-1971, IEEE Standard: Criteria for Protection Systems for Nuclear Power Generating Stations.

Institute of Electrical and Electronics Engineers, IEEE Standard 308-1972, IEEE Standard: Criteria for Class 1E Electrical Systems for Nuclear Power Generating Stations (IEEE Standard 308-1980, a later version).

Institute of Electrical and Electronics Engineers, IEEE Standard 603-1980, Criteria for Safety Systems for Nuclear Power Generating Stations.

Institute of Electrical and Electronics Engineers, IEEE Standard 1050-1989, Guide for Instrumentation and Control Equipment Grounding in Generating Stations

Institute of Electrical and Electronics Engineers, 7-4.3.2-1993, Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations.

MIL-STD-416C, Electromagnetic Emission and Susceptibility Requirements for the Control of Electromagnetic Interference.

MIL-STD-462, Measurement of Electromagnetic Interference Characteristics.

6.2 Articles and Books

Dahll, G., Barnes, M., and Bishop, P., "Software Diversity - A Way To Enhance Safety?" Second European Conference on Software Quality Assurance, Oslo, Norway, May 30 - June 1, 1990.

Fujii, R., "Software Engineering for Instrumentation and Control Systems," Nuclear Plant Instrumentation, Control, and Man Machine Technologies, Oak Ridge, TN, April 19-21, 1993.

Leveson, N.G. and Turner, C.S. "An Investigation of the Therac-25 Accidents," COMPUTER, pas 18-41, July 1993.

Personal Communication with Mr. Carl Johnson, NRC.

Neumann, P.G., "Illustrative Risks to the Public in the Use of Computer Systems and Related Technology," ACM SIGSOFT, Software Engineering Notes, Vol. 17, No. 1, pas 23-32, January, 1992.

U.S. Nuclear Regulatory Commission, "Survey and Assessment of Conventional Software Verification and Validation Methods," NUREG/CR-6018, April 1993.

U.S. Nuclear Regulatory Commission, Licensing of Production and Utilization Facilities, Title 10, Code of Federal Regulations, Part 50, Appendix A: General Design Criteria for Nuclear Power Plants and Appendix B: Quality Assurance for Nuclear Power Plants. (Published Yearly)

U.S. Nuclear Regulatory Commission, "Analysis and Evaluation of Operational Data, 1991 Annual Report," NUREG-1272, Vol. 6, No. 1 and No. 2, July 1992.

U.S. Nuclear Regulatory Commission, "High Integrity Software Standards and Guidelines," NUREG/CR-5930, September, 1992.

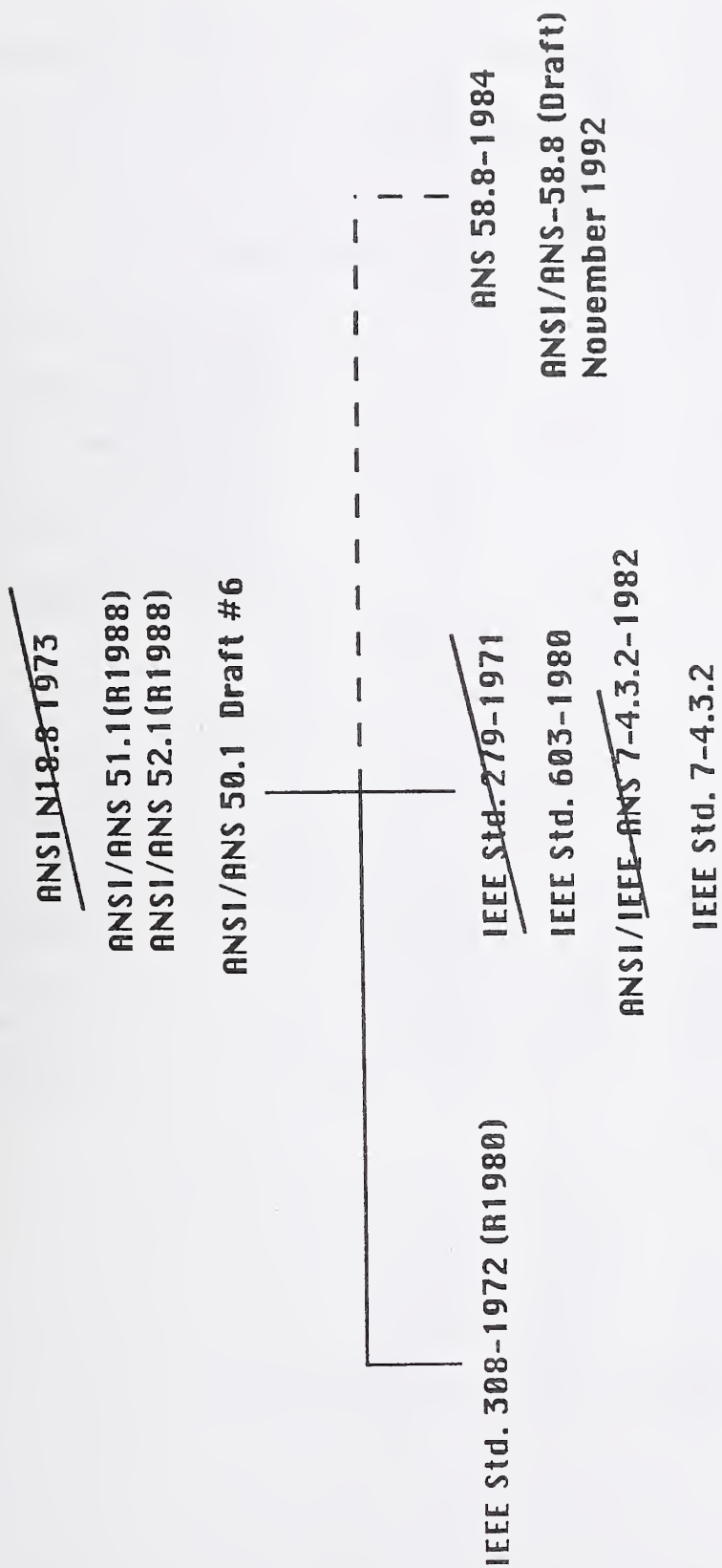


Figure 1: US Standards for the Design of Safety Critical Systems

Outline of a Framework of Design Criteria for a Computer-Based Saftey System

Goal: Operate NPP Safely

Development Process Guidelines and Standards

- Software Design Standards and Guidelines
- Quality Assurance Standard
- Verification and Validation Standard
- Configuration Management Standard

Design Basis Data Base

- Identify and Document All Functions
- Deterministic Analyses Saftey Functions
 - Limiting Design Basis Events
 - Response Times
 - Single Failure Criteria
- Hazard Analyses
- Probablistic Analyses

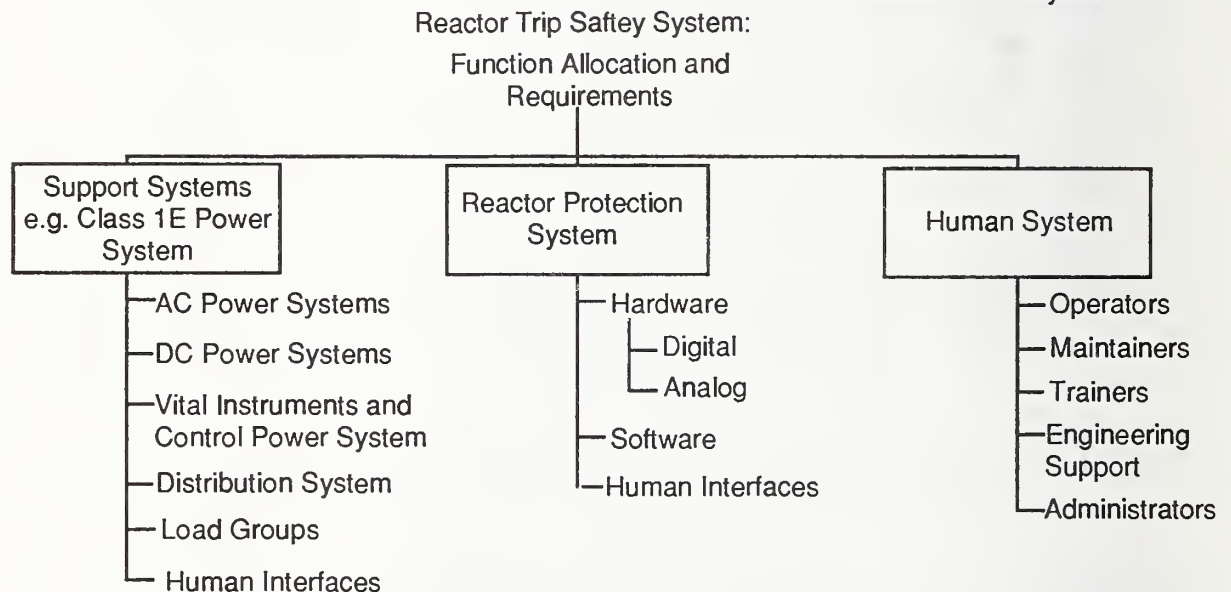


Figure 2: Outline of a Framework of Design Criteria

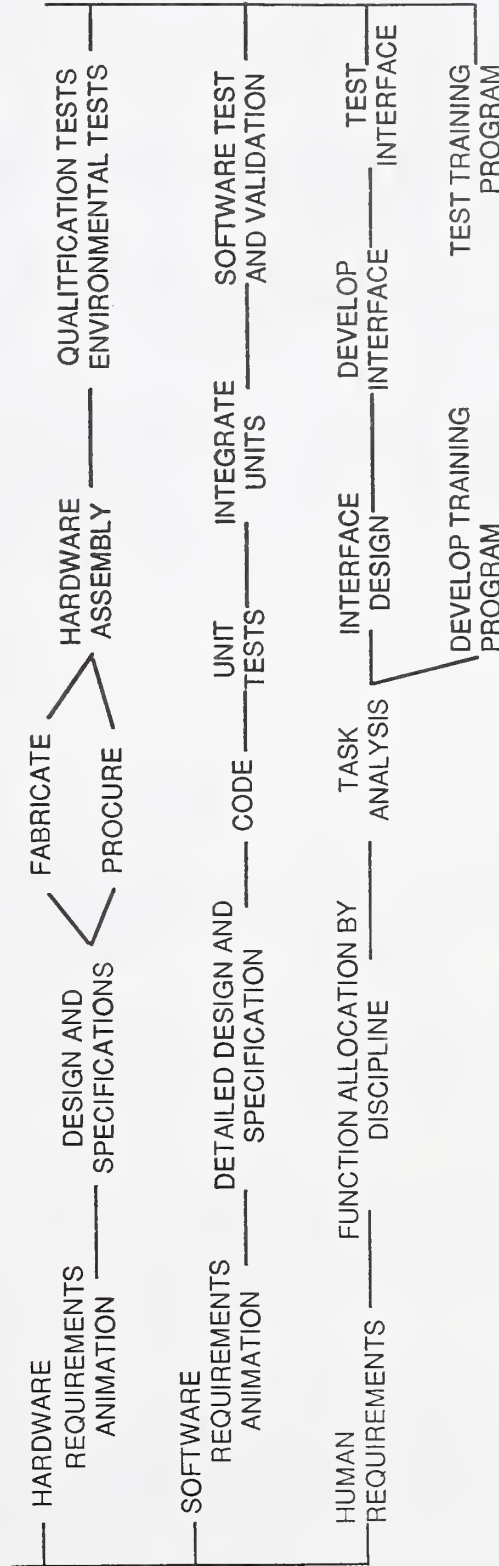
SYSTEM ANALYSES

GOALS

FUNCTION ANALYSIS
 DETERMINISTIC SAFETY ANALYSES
 SYSTEM HAZARD ANALYSES
 PROBABLISTIC RISK ASSESSMENT

SYSTEM REQUIREMENTS

FUNCTION ALLOCATION



SYSTEM INTEGRATION

SYSTEM TEST AND VALIDATION

Figure 3: Conceptual Model of the System Design and Development Life Cycle
 Note: Model as Displayed Omits any Assurance Activities

3.3 Industry Perspective on Digital Upgrades: Mr. Richard J. Blauw

A UTILITY PERSPECTIVE ON DIGITAL UPGRADES

Richard J. Blauw
Commonwealth Edison Company

presented to:
Digital Systems Reliability and Nuclear Safety Workshop
September 13-14, 1993
Rockville Crown Plaza Hotel
Rockville, MD

Abstract

Nuclear utilities face the need to upgrade aging and obsolete safety related and other critical equipment. This is the result of operation and maintenance concerns for reliability and maintainability. Digital technology is an option for these upgrades.

A number of utilities have attempted exercising the digital option. The regulatory licensing results have been inconsistent and have raised a variety of issues. These issues and the subsequent licensing uncertainties have caused some utilities to temporarily drop digital technology as an upgrade option.

Resolution of these issues and the need for regulatory stability is driving the development of industry standards and guidelines. These will provide guidance to support consistent design and implementation of digital upgrades. Successful completion of these documents is necessary for renewed consideration of the use of digital technology.

This paper will present a utility perspective on how project management, configuration control, and a rigorous design process can serve to address the present regulatory issues. These issues include commercial grade dedication, reliability, electromagnetic interference, and failure and error management. This perspective is consistent with the standards and guidelines development effort.

Introduction

Nuclear utilities face the need to upgrade aging and obsolete safety related, and other critical equipment. These upgrades are needed to address concerns associated with reliability and increased maintenance activity. These activities can have a profound impact upon component availability and maintaining personnel radiation exposure As Low As Reasonable Achievable. Minimizing the number of mechanical components through the use of digital technology will improve maintenance efficiency, and component and system reliability.

The need for digital Instrumentation and Control (I&C) in the Advanced Light Water Reactor (ALWR) has been recognized. The basis for this need, though, is different than that for existing reactors. The lessons learned from upgrading aging equipment should be factored into ALWR design and licensing efforts. However, care should be exercised in attempting to apply the ALWR concepts to digital upgrades on existing nuclear power plants.

This paper will present a utility perspective on how project management, configuration control and a rigorous design process can serve to address present issues. These issues include commercial grade dedication, reliability, electromagnetic interference, and failure and error management. This perspective is consistent with standards and guidelines development efforts.

Most of this information is based upon experiences gained through the development of proposed revisions to Institute of Electrical and Electronics Engineers (IEEE) P-7-4.3.2, *American National Standard, Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations*.

Additionally, information is based upon:

- support of emergency diesel generator performance monitoring and governor control systems for Zion Nuclear Power Station, and
- work supporting the development of the Nuclear Management & Resources Council (NUMARC) Digital Upgrade Guideline,
- work associated with the Work Group Computer Software under the American Society of Mechanical Engineers' Nuclear Quality Assurance Committee,
- work with the Nuclear Utilities Software Management Group,
- support for the Station Blackout Diesels at Dresden and Quad Cities Nuclear Power Stations.

History of IEEE P-7-4.3.2

In July of 1990, efforts were initiated to revise ANSI/IEEE-ANS-7-4.3.2-1982. This was based upon the need to reflect current computer software and hardware design practices in this standard. The need for a clearer definition of the relationship of this standard to its mother document, IEEE Std 603-1991, *Standard Criteria for Safety Systems for Nuclear Power Generating Stations*, was also deemed necessary. As such, the working group chose to replicate the IEEE Std 603-1991 format in P-7-4.3.2. This provided a framework to maintain focus on existing safety system design criteria. The working group chose to provide amplification for those safety system design issues which are unique or perceived unique to computer hardware, software, or firmware. It should be clearly understood that when a computer is to be used as a component of a safety system, both IEEE Std 603 and P-7-4.3.2 should be used to define all of the design criteria.

Much of the revision effort was focused on the following safety system criteria: quality, equipment qualification, system integrity, independence, and reliability. In addition, electromagnetic environment issues were addressed under the safety system design basis.

It should also be understood that IEEE Std 603-1991 and IEEE P-7-4.3.2 usage of the phrase "safety system" is consistent with the 10CFR50.49 usage of the phrase "safety related electric equipment". There is, therefore, a one-for-one relationship between the standards and regulation.

NUMARC Digital Upgrade Guideline

No consensus document exists which presents guidance to a design approach for microprocessor based digital systems. In work efforts associated with the development of IEEE P-7-4.3.2, this was informally identified as a future effort item. There are many existing standards for specific requirements and design techniques. None provide overall digital system development approach recommendations.

In recognition of the industry wide impact of current regulatory digital upgrade concerns, NUMARC has initiated development of an industry position on this subject. This effort is directed toward delineation of a digital upgrade design process. This will include defining failure modes, and recommendations to minimize their affect upon nuclear plant safety. This guideline will also provide plant data based recommendations to electromagnetic related issues. These definitions and recommendations will be embodied within a structured approach to system design. This guideline will help to fill the void identified during the IEEE P-7-4.3.2 development.

With Nuclear Regulatory Commission staff involvement, this effort should result in clarification of an appropriate digital upgrade design and licensing methodology. Completion of this guideline is critical to utility digital upgrade efforts.

System Design Model

The issues of concern (e.g., commercial grade dedication, electromagnetic interference, reliability, verification and validation) may be addressed through project management, configuration controls, and a systematic approach to the design process.

Project Management and Configuration Control

At the onset of system design and implementation, project scope and roles should be clearly defined. This project management task is a cornerstone for successful completion of a safety related system digital upgrade. This task should include definition of the methods for configuration control of the configuration items produced during design and implementation activities, and subsequent operation and maintenance activities should be identified. Procedures and standards should be identified to support the design process. These procedures and standards should address design verification activities to be performed and by whom. Finally, it should also provide delineation of any required commercial grade dedication efforts.

The level of design verification and commercial grade item dedication activities might be based upon a sub-categorization of safety related systems. This graded approach would allow emphasis to be placed upon those safety related systems for which a failure would have a significant impact upon safety (e.g., the Reactor Protection System and Engineering Safety Feature Actuation Systems), and a lower level of emphasis upon systems for which a failure would have a reduced impact on safety. A joint Electric Power Research Institute (EPRI)/Utility working group dealing with issues of Verification and Validation (V&V) is

working to define software design activities levels based upon a safety related system sub-categorization.

In addition to scope and roles definition, digital upgrade configuration baseline establishment should be performed. This should be accomplished through verification of existing plant information including, wiring diagrams, schematic drawings, equipment and system alarm and trip setpoints, and any other applicable engineering calculations and assumptions. This verification eliminates nuclear plant digital upgrade base information uncertainty. This information is the foundation of the entire design, installation, and testing effort.

Design Process

A rigorous design process should exist for the development of safety related digital upgrades. As stated in the introduction, no currently used standards define a system level development process. Numerous standards delineate criteria and requirements to be met. None offer guidance in the steps which should be performed. The developing NUMARC Digital Upgrade Guideline should help address this void. This guideline will delineate how current issues of concern can be dealt with during safety related system design.

An inherent element of the design process should be the identification and resolution of Abnormal Conditions and Events (ACEs) which have the potential to defeat the safety related function. ACEs include external events as well as conditions internal to the computer hardware or software. P-7.4.3.2 provides details of ACEs which could be introduced during specific activities of the design process. ACEs consideration is necessary to provide assurance that the computer

and the remaining components of the safety related system will respond when exposed to realistic normal and abnormal situations, including potential common mode failures.

Analysis techniques, such as Failure Modes and Effects Analysis (FMEA) or Fault Tree Analysis (FTA), are among the methods recommended to identify ACEs requiring attention. These analyses should be of sufficient detail to include consideration of utilized software. During the design and implementation, periodic reviews should be performed to confirm the analysis results. The analysis should be updated as appropriate. This periodic review provides assurance that identified failure modes are properly addressed, thus minimizing design errors.

Specific requirements should exist for design verification throughout the design process, from requirements definition to installation and check-out of the final product. P-7-4.3.2 provides considerations supporting an integrated verification approach.

Figure A depicts a view of an example design process. At the onset of safety related system design, requirements are allocated to non-computer hardware and to the computer. A secondary allocation is made of computer requirements to computer hardware and software, and their integration. Design verification should be performed at the conclusion of requirements allocation. This verification provides confidence that appropriate allocation of requirements has been made. During requirements allocation, the FMEA/FTA could be initiated to address system level failure concerns.

The next activity of this design model is the design and implementation of non-computer hardware, computer hardware, and computer software. The system level FMEA/FTA should be amplified to

address computer hardware and software, and non-computer hardware failure possibilities. This information becomes input to the design. Design verification should be performed following completion of design and implementation.

Computer hardware and software design verification testing is intended to provide confidence of the correct applicable requirements implementation. It can be accomplished through one of three approaches. The first approach would include design verification of the test plans, test cases, and test results to provide confidence that the test developer has considered both normal and abnormal situations in assuring that the non-computer hardware, computer hardware, and computer software components perform correctly. In this approach, the designer would be able to be the preparer of the test plans and cases, the executor of the tests, and the reporter of the test results. A second approach would have an independent party develop the test plans, and test cases, execute the tests, and report on the results. A third approach would include design verification of the test plans and test cases, design verifier execution and reporting of the tests and test results, respectively. The emphasis here is design verification should not be a duplication of documented work performed, so long as the requirements for verifier independence are met.

Computer hardware and software integration activities should be performed following completion of applicable design, implementation, and test activities. This integration may include testing activities to satisfy the designer that all software requirements which were dependent upon computer hardware and the computer hardware requirements which were dependent upon software have been met. In addition, testing should confirm the computer hardware and software failure detection and response mechanisms

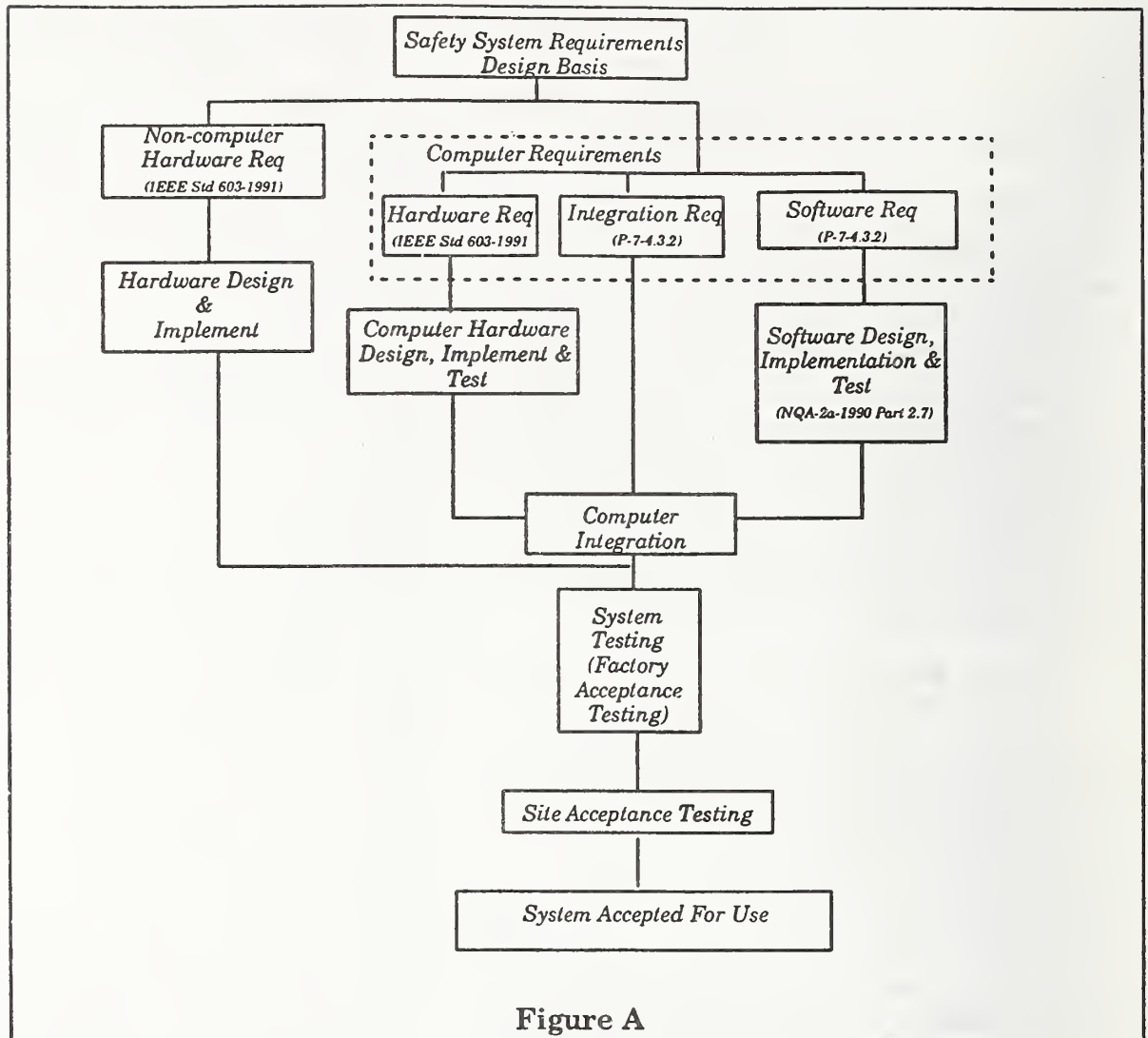


Figure A

function properly. Design verification of these activities should be performed.

As is practical and feasible, testing of the integration of the computer with a portion of the hardware design should be performed during factory acceptance testing. This testing should begin to confirm that the original requirements have been implemented. However, this testing should not be based only upon requirements. Realistic scenarios such as loss of power, failure of components, loss of communications, and other failures identified in the FMEA/FTA should be

played out to assess the ability of the computer and the non-computer hardware to respond in an appropriate manner. Design verification for factory acceptance testing can be accomplished in one of the three approaches previously described.

Upon completion of the factory acceptance testing, site acceptance testing should be performed to confirm the correct implementation of requirements which were not tested during factory acceptance testing. Site acceptance testing is intended to confirm that the system is compatible with the plant and to provide assurance that no

shipment or installation damage occurred. This should include testing of field instrumentation output thru to the safety related system output. The three design verification approaches described previously would be applicable for site acceptance testing.

This example of a design process approach inherently has the level of independent reviews and tests (i.e., V&V) necessary to produce a system which has a high quality level and assurance of proper and correct operation. No additional independent activities should be necessary. This is to say that no additional activities are necessary to address what is known as V&V.

Design Process Issues

Through a systematic approach to design, current issues can be addressed in a planned and orderly manner. These issues include commercial grade dedication, reliability, electromagnetic interference, and common cause failures. These issues should be addressed during the course of safety related system design from a system engineering perspective. As stated earlier, this is the current direction of the efforts associated with the development of the NUMARC Digital Upgrade Guideline.

Commercial Grade Dedication

One key commercial grade dedication area which requires attention is commercial vendors development and control process. Believing that one can completely test the commercial grade computer in a manner similar to what has been done for other commercial grade items is inaccurate. Utilities are beginning to develop and implement processes for understanding the internal structure of such items as pipes and valves (e.g., non-destructive examination). In the same manner, a dedicator needs to understand the internal structure of the computer, including the software, upon which a particular application is built. This means that an evaluation based upon nuclear industry standards must be performed on the vendor's computer hardware and software development and configuration control processes.

Many vendors have procedural control of their design process. In some cases this results in documentation consistent with non-nuclear computer standards. This documentation should be reviewed and evaluated by technically knowledgeable people representing the utility. This review could serve as an offset for lack of independent reviews by the vendor. Compensating factors,

such as product maturity, product stability and operating experience in a similar application (Programmable Logic Controller opening valves or starting pumps in a refinery), may serve as an offset in those situations where documentation is lacking.

Another compensating factor may be evidence of product stability. An established configuration control process and error notification will help provide the necessary evidence of (or lack of) product stability. No one type of compensating factor should be used in lieu of a documented design process. Combinations of operating experience, configuration control, and error notification may give an appropriate confidence level in the overall quality of the commercial product. The utility needs proper evidence to support commercial dedication. The fact exists that as a result of this detailed dedication effort, a vendor's product might be eliminated from consideration for safety related purposes.

Acceptance levels for compensating factors should be defined. These levels might be based upon safety related system sub-categorization. This might also include definition of commercial grade dedication activities which are and are not acceptable.

The dedicator should test the product for acceptability with safety related system requirements. This testing is in addition to the previously described vendor evaluation. A document, detailing the functional and performance requirements which the computer must be able to satisfy to perform its safety related function, should be prepared and approved. This document should serve as the basis for commercial computer test development, acceptance, and execution. The

results should be summarized and approved. The level of testing should be commensurate with the sub-categorization.

The commercial dedication process should be viewed as supplementing commercial documentation and not as a design control avoidance mechanism. In point of fact, the utility must shoulder all responsibility for the installed product, including any reporting requirements.

Reliability

The need for a quantitative reliability estimate is dependent upon the design approach employed - probabilistic or deterministic. The former is based upon a Probabilistic Risk Assessment (PRA) resulting in system reliability goals used as an input to the design process. It should be noted that the PRA process focus for existing US reactors is identification of potential vulnerabilities. It is not necessarily for the determination of system reliability goals for use as design input.

The draft Supplement to International Electrotechnical Committee (IEC) 880, *Software Important to Safety for Nuclear Power Plants*, indicates that under certain conditions, a 10^{-4} probability of revealed failure can be predicted. A reliability ranging between 10^{-3} and 10^{-5} may be achievable based upon a rigorous design process and dynamic testing. These estimates are predicated upon good design practice and product testing.

System design is typically performed in a procedurally oriented, or deterministic manner. Therefore, a qualitative view of reliability is more appropriate. Emphasis should be placed upon a rigorous design process to achieve reliability. Identification of Abnormal Conditions and Events (ACEs) throughout system design or through a Failure Modes and Effects Analysis (FMEA) or Fault Tree Analysis (FTA) should be considered. The use of an FMEA or FTA is

suggested by IEEE Std 603-1991 for addressing reliability. Our focus should be on the design process, identification of ACEs, and testing, not on so strongly upon generating a PRA number.

While the probabilistic approach may have merit for the design of the next generation of reactors, it is not an approach consistently used for upgrades to existing nuclear plant systems.

Electromagnetic Interference

Electromagnetic Interference/Radio Frequency Interference (EMI/RFI) and other electromagnetic concerns have received a significant amount of attention. P-7-4.3.2 states that the computer must be able to function in the environment to which it will be exposed. This environment includes electromagnetic fields. A joint EPRI/Utility group effort is presently working to define a process for meeting this requirement. Final recommendations will be consensus standards and plant data based.

Initial attempts to address this issue have included development of an EMI/RFI map, representing both the atypical and a "worst case" transient environment. This approach attempts to demonstrate that the mapped emissions levels are sufficiently less than the susceptibility levels for which the equipment was tested. However, IEEE Std C62.41-1991, *ANSI/IEEE Recommended Practice on Surge Voltages in Low-Voltage AC Power Circuits*, states that "While short-term monitoring of an individual site often gives some useful information, the environment is so dynamic that the analysis of a brief period may not give a good prediction of the future environment."

It would seem that while mapping may have some limited value, focus should be upon proper design techniques, installation techniques, and testing. Proper equipment shielding and grounding

should serve to protect both the computer and surrounding equipment. Appropriate tests should be executed to confirm the adequacy of these actions.

Component and system test evidence strongly indicates that some of the analog components in a digital upgrade (e.g., analog power supplies) may be more susceptible to EMI than the digital components. Licensee Event Reports provide evidence that EMI (i.e., RFI from walkie talkies) has been a contributing factor in plant transients for many years. It should be understood that the EMI/RFI issue pertains to both analog and digital systems; it is not unique to digital systems. Subcommittee 6 (SC-6) of IEEE's Nuclear Power Engineering Committee has agreed, in principle, to deal with this as a system requirements issue. This effort will be initiated at the SC-6 Fall 1993 meeting.

Managing Failures and Errors

Safety related system failure and error management is a design issue which requires attention. Determination of Abnormal Conditions and Events (ACEs) through the use of a Failure Modes and Effects Analysis (FMEA) or Fault Tree Analysis (FTA) is a practical method for this effort. Design decisions can be made to address the failures identified. Use of an analysis technique allows the emphasis to be on the cause of failures and errors instead of the effect (e.g., software common mode failures).

Software common cause failures are the result of design deficiencies. These may in turn lead to common mode failures. Common cause failures may be the result of common hardware, software or components; common personnel, languages or tools; common documentation practices; similar internal interface points, such as voting system inputs, or input interfaces; common processes such as detection of states, algorithms, signal

conversion, or voting. It should be recognized that these are design related issues and should be dealt with at that level. Two methods which may address common cause failure concerns are the separation of design teams and analysis of the signal trajectory path through the software. This analysis, as detailed in the Draft Supplement to IEC 880, is used to confirm independence of common modules or software developed with common tools. Draft Supplement to IEC 880 provides additional detail regarding software trajectories.

IEEE Std 379-1988, section 5.5, indicates that common cause failures not subject to single failure analysis include those that can result from design deficiencies or manufacturing errors. Design qualification and quality assurance programs are intended to afford protection from design deficiencies and manufacturing errors. This guidance should be equally applicable to digital and analog system design. Therefore, for analysis purposes, the assumption that a software common mode failure could occur should not be required.

However, if one must assume the potential of a common mode failure due to software, the focus should be on the successful completion of a safety function, not on the necessary completion of a safety related task. Recognition of the existence of backup systems should be given (e.g., BWR core injection at low pressure using either LPCI or core spray, manual actions in lieu of automatic ones). These echelons of defense have been a foundation of plant safety. Common cause failure concern should be limited to those situations when the safety function may not be accomplished, not when one level of a defense echelon may be lost. Overemphasis on common cause/common mode failures at the software component level has overshadowed the design processes and

techniques which can serve to minimize failure and error potential.

Through performance of an FMEA or FTA, system level management of failure and errors can be emphasized. Use of various design techniques such as watchdog and deadman timers can be employed to identify a failure and provide an acceptable response (e.g., warn, alarm, or placement in the appropriate preferred failed mode).

In addition, a design process similar to the model described earlier should be followed. This is consistent with the guidance provided in IEEE Std 379-1988. Emphasis should be placed upon the design process which addresses failure and error identification, and testing to provide assurance that the implemented design can identify and respond correctly.

Recommendations

As stated earlier, use of a rigorous design process with appropriate levels of design verification should be acceptable to address concerns for common cause failures, software reliability, and EMI. Successful completion of the NUMARC Digital Upgrade Guideline is a key element to provide consistency in the design of digital upgrades. With any process, improvements can always be made. One such potential area of improvement is delineation of requirements.

Studies have shown that the majority of design problems associated with software occur as specification faults. In particular this occurs in the English language translation of requirements to a designed computer program. Industry standards (e.g., IEEE P-7-4.3.2, IEEE Std 1012-1986, NQA 2a-1990 Part 2.7) place emphasis upon stating compete and unambiguous requirements. This is a basis for assuring that the final validated system complies with the requirements. Attention should be given to techniques which would facilitate specifying of requirements and design verification processes. Combinations of text and graphical techniques (e.g., SAMA logic diagrams, state transition diagrams, truth tables) may be a vehicle to minimize the potential for specification faults. Some of these techniques are used today in the computer and control system markets.

Another recommendation is use of analysis techniques such as FMEA or FTA to identify Abnormal Conditions and Events, and to address both safety related system reliability and the single failure criterion. It should be understood that for these to accurately reflect the safety related system and all its components, one must consider the affect of the computer.

Finally, EPRI and EPRI/Utility groups efforts addressing commercial grade Programmable Logic Controllers (PLCs) usage, Verification and Validation, and EMI should be encouraged and expeditiously completed.

Closing

The current uncertainty in the licensing arena has had a significant impact upon Commonwealth Edison including the cancellation and deferment of safety related digital system implementations. These actions resulted directly from the licensing experiences for Zion Nuclear Power Station's digital protection system upgrade, and other utility licensing attempts.

Eagle-21™ Digital Reactor Protection System

The design approach used in the Zion Station Reactor Protection System (RPS) replacement is compatible with the systematic design approach presented. This replacement employed Westinghouse Electric Company's Eagle-21™ product.

During Commonwealth Edison's design and implementation efforts, EMI mapping of the area where Eagle-21™ was to be installed was conducted. Based upon analysis of Westinghouse testing documentation, Commonwealth Edison believed that Eagle-21™ would be suitable for the environment. In addition to environmental suitability, the Westinghouse Design and V&V process, and test results were evaluated and found acceptable. Additionally, a review of documented NRC concerns from other utility digital upgrades was performed to identify unaddressed issues. None were found. The conclusion was reached that Eagle-21™ was a qualified system for Commonwealth Edison use.

This replacement was subjected to an extremely detailed NRC audit. As a result of the audit, Commonwealth Edison was required to perform additional EMI mapping and testing. No new EMI related concerns were identified. Additional analysis was also required to address the concern of common cause failure. An extensive defense in

depth analysis was conducted. In support of this analysis, extensive simulator scenarios were executed, and a detailed review of normal and abnormal procedures was performed. The requirement for this effort did not seem to be based upon any documented technical basis or NRC staff position. Complying with these requirements was not without financial, scheduling and manpower impact.

One key point should be clearly understood. NO DESIGN CHANGES WERE REQUIRED as a result of these additional efforts. This serves as a successful validation of the design approach presented.

The defense in depth analysis took credit for the existence of the Anticipated Transient Without Scram (ATWS) system. As a result, the NRC encouraged Commonwealth Edison to place an Administrative Technical Specification on this Balance of Plant system.

Emergency Diesel Generator Control and Monitoring

Significant efforts have been undertaken to improve the reliability of Zion Station's emergency diesel generators. This included the implementation of a safety related Programmable Logic Controller (PLC) based emergency diesel generator control system and a non-safety related Distributed Control System for data acquisition and performance monitoring.

Prior to Eagle-21™ licensing efforts, a wide range of available options, from relay rack to PLC to Versa Module Eurocard (VME) bus architecture, were considered. Emphasis was directed toward the two latter options, both digital based. No evidence of previous NRC review of a VME based

system existed. However, evidence of NRC review and approval of PLC based digital upgrades did exist. Believing that this familiarity would facilitate licensing efforts, the conservative non state-of-the-art PLC option was selected. However, as a result of the Eagle-21™ licensing effort, the decision was made to utilize 1960s hardwired relay technology. This method has allowed Commonwealth Edison to proceed with the implementation under 10CFR50.59 and avoid the uncertainties associated with NRC approval of a digital system.

Even though the digital control system was dropped, Commonwealth Edison is moving forward with the implementation of the non-safety related DCS. This action was taken due to the distinct need for data to assist in improving the reliability of the diesels.

It is believed that elimination of moving parts is a key element of overall improved system reliability. A PLC implements relay ladder logic with minimal use of moving parts. While the reliability of the Zion Station emergency diesel generators will be improved, it may not be to the level which would have been possible with a PLC.

Emergency Diesel Governor

As part of additional efforts to improve the reliability of Zion Station's emergency diesel generators, a digital based governor installation was proposed. A final course of action has yet to be determined. It is hoped that stability in the licensing arena would allow the decision to be based upon technological considerations rather than regulatory ones.

Author's Observations

There are significant safety and cost benefits associated with digital systems. However, the uncertainty in the licensing arena has led to many

utilities adopting a "wait and see" attitude. Only with the adoption of a consistent practical digital upgrade approval process, will the utilities realize these benefits.

Has all the attention which has been given to digital upgrades resulted in improvements to safety? Based upon the lack of documented, quantifiable evidence, it is doubtful.

Contributions

The significant technical and editorial contributions of the following individuals is acknowledged and greatly appreciated

J. Katrenak
J. Lasky
R. Mason
J. Matras
T. Randolph
R. Reeves

References

- IEEE P-7-4.3.2, *American National Standard, Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations*, Draft 7
- IEEE Std 603-1991, *Standard Criteria for Safety Systems for Nuclear Power Generating Stations*,
- IEC 880, *Software Important to Safety for Nuclear Power Plants*
- IEEE Std C62.41-1991, *ANSI/IEEE Recommended Practice on Surge Voltages in Low-Voltage AC Power Circuits*

3.4 Experiences from Application of Digital Systems in Nuclear Power Plants: Mr. Paul K. Joannou

Experiences from Application of Digital Systems in Nuclear Power Plants

Paul K. Joannou
Ontario Hydro
700 University Avenue
Toronto, Ontario
M5G 1X6

ABSTRACT

Digital systems have been integral to CANDU nuclear power plants from their inception 25 years ago. The four most recently commissioned CANDU generating units at our Darlington Nuclear Generating Station have the highest degree of incorporation of digital systems of the 20 units operated by Ontario Hydro. Darlington was the first CANDU plant to use digital systems for the complete safety shutdown function. The use of digital systems in the shutdown systems at Darlington led to some licensing issues that required a substantial effort by Ontario Hydro and Atomic Energy Canada Limited (AECL), the designers of the shutdown systems, to address. Our years of experience with digital systems and our experience in addressing the Darlington licensing issues were used to develop an approach to engineering digital systems for nuclear power plants. The approach is documented in a family of standards, procedures and guidelines. This paper defines the licensing issues that needed to be addressed for Darlington shutdown systems, defines the expectations of the regulator and the utility with respect to the approach, describes the approach itself, describes the issues that still require further work, and describes the status of the work program that is in place at Ontario Hydro and AECL to address outstanding issues and continually improve the approach.

BACKGROUND

Ontario Hydro and AECL have a long history of successful application of digital systems within CANDU nuclear power plants. Our first plant was the single unit Douglas Point Nuclear Generating Station that was commissioned in 1968. This station used a digital system for process monitoring. The station was owned and designed by Atomic Energy Canada Limited (AECL) and was operated by Ontario Hydro. In each subsequent station the use of digital systems has increased. The Pickering A Nuclear Generating Station is a four unit 540 MW per unit station commissioned from 1971 to 1973. It made use of dual redundant digital control computers to control reactor neutron flux, as well as other major control tasks. It provided process monitoring of both analog and contact inputs. Each of the subsequent four unit stations Bruce A, Pickering B and Bruce B made evolutionary improvements on the use of digital systems. Increased number of process inputs and outputs were added as more control and monitoring functions were added. Evolutionary improvements to the operator interfaces were also made with improved usage of colour CRTs. Digital systems are also used for control of fuel handling machines, operator information systems and in various other control and protective applications.

The design expertise for the major digital systems has been provided by AECL, General Electric Canada and Ontario Hydro. The performance of our nuclear stations routinely scores in the top ten of the world's reactors and the digital systems are credited as one of the contributors to this success.

Our most recent station, Darlington, makes more use of digital systems than any of its predecessors. Some of the key areas of increased usage were the use of an Ontario Hydro designed programmable logic controller (PLC) for nuclear system discrete logic that had been implemented using relays in previous stations, and the use of digital systems for both reactor shutdown systems.

The use of digital systems to implement the reactor shutdown function in both shutdown systems led to licensing difficulties with our regulator, the Atomic Energy Control Board (AECB). AECL, the designer of the shutdown systems, had developed prototype digital shutdown systems around 1980 to determine their feasibility. Feedback from the AECB at that time indicated that there was no inherent problem with the use of digital technology in the shutdown systems. A rigorous software quality assurance program was put into place, and development was started in the early 1980s. An AECB audit of the project in the late 1980s resulted in some audit findings that ultimately left the AECB with a lack of confidence in the software. The AECB hired an external consultant, Dr. David Parnas, to provide an expert opinion on the adequacy of the software. Dr. Parnas' report identified several areas for improvement in the software. Over a two year period Ontario Hydro and AECL expended substantial effort making changes to the software to try to address identified areas of concern expressed by the AECB and Dr. Parnas. Ultimately, the confidence of the AECB was gained and they were convinced that the software was safe and that Darlington could be licensed, but by this time the delay in getting a license contributed to several months of delay in commissioning the station. Several months delay translated to many tens of millions of dollars when interest charges and replacement energy costs were taken into account.

One of the key problems during the two year period was that there was no widely accepted definition of what constituted good enough for software used in safety critical applications such as shutdown systems. Available standards did not provide the measurable acceptance criteria that was needed. As a result of our two years of effort to convince the regulator of the acceptability of the software a license was granted to Darlington. The license was only granted for the short term though. In the medium term, the AECB wanted the software redesigned to make it more easily reviewable and modifiable, so that the effort required to re-gain confidence each time a change is made to the software, will not be as great as for the initial version. The AECB also required that agreement be reached on the standard that will be used as the basis of the re-design.

This led directly to the strategy by AECL and Ontario Hydro to jointly develop a family of standards, procedures and guidelines that document the acceptable practices for software engineering of software used in nuclear power plant protective, control and monitoring applications. This joint program has been coordinated by a committee called the Ontario Hydro / AECL Software Engineering and Standards (OASES) committee. The OASES work program has resulted in a scheme for categorization of software applications with respect to

nuclear safety, a high level standard defining the methodology independent requirements for each software category and detailed procedures documenting specific methodologies that we have developed for some of the processes where adequate methods were not readily available.

The standard for safety critical software and its associated procedures are being applied by Ontario Hydro for the development of a digital trip meter to replace an obsolete analog trip meter at our Pickering Nuclear Generating Station. They are also being applied by AECL for the development of two shutdown systems for the South Korean Wolsong 2 Nuclear Generating Station. The experience gained in applying them will be used to update them before their application to the Darlington shutdown system software re-design project.

DARLINGTON SHUTDOWN SYSTEM LICENSING ISSUES

During the two year period that we tried to convince the AECB of the acceptability of the software various issues were raised that needed to be addressed. Below the issues are described along with the actions that were taken to address each issue. The issues have been grouped into the areas of reviewability issues, safety issues, functionality issues, reliability issues and maintainability issues.

Reviewability Issues

One of the key issues was the reviewability of the software. Review of the software is one of the methods by which the regulator gains confidence in the software. In order to be reviewable, a design documentation organization must be used that describes the design in increasing levels of detail, or decreasing levels of abstractions, so that a reviewer may fully understand the intended functionality of the software, then understand the partitioning of the software into design modules along with their interactions, and finally to understand the code that implements each module. At each step, the documentation must be organized to facilitate third party review.

The reviewability of the software is also affected by the structure of the software design. Most designs are sufficiently complex that a reviewer cannot review the whole system at once, but rather must review the design on a per module or group of modules basis. If the design has not resulted in loosely coupled modules with high cohesion, then to gain an understanding of the functionality of an individual module is difficult since the reviewer must concurrently understand the relevant interactions with other modules.

For the initial version of the Darlington shutdown system trip computer software, the AECB and its consultant felt that the software design and documentation were not adequate to allow sufficient review of the software. In response to these issues, the software was restructured to increase its reviewability, the software design documentation was improved and the code commentary was improved.

Safety Issues

Gaining confidence that the behaviour of the software would be predictable over time and safe in the presence of failures were other issues of concern to the AECB. The issue of predictable behaviour had been addressed in the original design by having a very simple control flow structure to the software design. The software in each trip computer consists of initialization code that ran upon startup, followed by an infinite loop that repetitively called the same sequence of routines on each pass of the loop. One change that was introduced to address the issue of predictable behaviour was to ensure that each variable in the software was initialized to a known state at the beginning of the main program loop, to the maximum degree possible (some variables needed to retain state information between program passes to implement the trip computer functionality and hence could not be initialized at the beginning of each pass). The initial value of each variable was chosen to be a value that would result in safe behaviour of the software in the presence of expected failures, eg flags used to track the trip state of each trip parameter were initialized to the "tripped" state and therefore would have to be written to by each trip module to set them to the "untripped" state.

Self checking software was included in the initial version of the trip computer software to detect hardware and software failures, and to then put the system into a safe state. One issue centered about the rationale for the degree of self checking software. The self checking software adds complexity to the software design and hence there is a tradeoff between adding self check software to increase the ability of the system to behave safely in the presence of failures verses decreasing the reviewability and reliability of the software through increased complexity. In response to this issue a review was done of the extent of self checking software which resulted in a smaller set of self checks that were balanced the degree of self check verses complexity equation.

Functionality Issues

Another concern raised was that of ambiguities within the requirements specification documents. The AECB found instances where they felt that the requirements specification was ambiguous and therefore they could not be confident that the software implementers actually interpreted the requirements in the same manner that the system designer, that produced the specification, intended.

To address this issue a more formal requirement specification was added to the document hierarchy between the textual requirements specification and the software design description documents. The more formal requirements specification was written using a notation that Dr. Parnas had used to specify the requirements for the A-7 aircraft while he was working at the Naval Research Laboratories. The formal notation was useful in producing a requirements specification that was uniquely interpretable.

Reliability Issues

Demonstrating to the AECB that the software was adequately reliable required more rigorous unit, system and validation testing than was originally established. An independent consultant was hired to review the adequacy of the testing performed and to recommend additional test cases.

The degree of diversity between the software within the trip computers of each shutdown system was also an issue. Diversity between the two shutdown systems is a regulatory requirement which had been addressed by using different hardware platforms (General Automation versus DEC), different implementation languages (Fortran and GA-assembler vs Pascal and DEC-assembler), and independent design teams for each of the shutdown systems. To address the concern an analysis was done of all Software Change Requests generated for each shutdown system trip computer during development and verification in order to demonstrate that no SCRs from the different trip computer designs would have resulted in a common mode error.

The reliability concern was also addressed by performing statistically valid, trajectory-based random tests to demonstrate that the probability of failure of the software was less than the 10^{-4} failures per demand that was required of it. The reliability demonstration was done by performing 7000 test cases, where each test case was randomly chosen to model one of six accident scenarios that the shutdown systems are designed to protect against.

Maintainability Issues

The AECB also had concerns with the ability to maintain the level of confidence in the software after changes are made to the software during its in-service period. The issues were associated with the adequacy of the design trails that documented the traceability of high level safety requirements through the design documentation into the software implementation, and the degree of understanding of the design rationale for the design decisions made during the design process. The concern was that without adequate documentation of these aspects of the design, future maintainers may make design change decisions that are inconsistent with the design decisions that were the basis of the original design. To address this issue the design principles that were the basis of the design decisions were documented, and supplementary design notes produced to document specific analyses that were done as a basis of some key design decisions.

Another related issue was with respect to the degree to which the development procedures would allow future maintainers to reliably make changes to the software with sufficient configuration control and re-verification. All project procedures were reviewed, and improved where deemed necessary.

EXPECTATIONS ON THE USE OF DIGITAL SYSTEMS

As the designers and operators of nuclear power plants, AECL and Ontario Hydro have expectations that the following benefits will be realised from the use of digital systems:

- higher system reliability
- higher availability because of self-annunciation of failures
- increased safety due to higher availability of safety systems
- increased production due to higher reliability, availability and better algorithms to avoid spurious trips
- improved human-machine interfaces
- more reliable, quicker design changes
- lower cost solutions
- integration of data with other plant control and monitoring systems

As the regulators, the AECB's expectations on digital systems used in safety related applications within nuclear power plants seem to be:

- adherence to an appropriate standard for the engineering of the digital system
- explicit criteria to decide if software is acceptable
- use of well defined and appropriate production processes
- design and construction consistency
- design and production rigour
- confirmation that requirements are correct and complete
- independent verification that requirements are fully satisfied
- adequate testing
- reliability estimation
- third party reviewable documentation
- maintenance at low risk
- coherent, rationale structure consistent with design principles
- use of competent, well qualified staff

OASES FAMILY OF STANDARDS, PROCEDURES AND GUIDELINES

The experience gained in developing and licensing the software for the Darlington shutdown systems, along with our common needs, led directly to formulating a joint Ontario Hydro and AECL strategy for developing a set of standards, procedures and guidelines for software engineering.

The joint approach made sense from many perspectives. The highly specialized skills required for safety critical software development were scarce and hence required a pooling of the small number of software engineers at AECL or Ontario Hydro who had the background to carry out such work. The effort to address the problems was significant and the joint approach allowed pooling of limited funds. Both companies had to interface to the same

regulatory body, the AECB. The activities to address the problems are being coordinated by a joint committee called OASES (Ontario Hydro/AECL Software Engineering and Standards).

The first task we worked on together was a high level standard for software engineering of safety critical software. As discussed in the previous sections, we felt that there was no agreed upon measurable definition of acceptability for the engineering of safety critical software. Until such a standard existed we felt that the licensing of future designs with the AECB would not be practical. A working group of AECL CANDU and Ontario Hydro staff prepared the Standard For Software Engineering of Safety Critical Software that was issued for trial use in December of 1990.

This standard defines the software engineering process, the outputs from the process and the requirements to be met by each output. The requirements are expressed in methodology independent terms so that various techniques for software engineering may be used to meet the requirements of the standard. This allows the standard to be used to assess the acceptability of various proposed techniques and allows for techniques to evolve without requiring changes to the standard.

This high level standard is intended as part of a framework of standards which consists of the following components:

- i) a guideline defining the procedure for categorizing software with respect to the effect of its failure on nuclear safety (with safety critical as the most stringent category).
- ii) a high level standard addressing the overall software engineering process for each category.
- iii) for each category, sets of standards, procedures, and guidelines to be used to perform specific activities within the software engineering process.
- iv) a guideline defining how to qualify pre-developed¹ software for use in each category.

The high level standards for software engineering define the requirements on the software engineering process, the outputs from the process, and the requirements that must be met by each output. The requirements are specified to be as measurable as possible, but do not unnecessarily constrain the methodology used to produce the output. For example, in the standard for safety critical software the requirements on the software requirements specification output specify that the specification must define the required behaviour of the software using mathematical functions but does not specify which notation or format should be used.

A set of specific standards, procedures, and guidelines are then developed for each of the categories. These specify the detailed methodology to be used in producing the outputs

¹pre-developed software refers to software that has been procured or developed for a previous project; i.e. already existing software

specified by the corresponding high level standard for the category.

Any of the specific standards, procedures, and guidelines may consist of or reference industrial and international standards provided that they conform with the appropriate high level standard.

The majority of our work to-date has been with safety critical software engineering. Over the last 2 years, we have used the high level standard as the basis for developing the detailed methodologies to be used in all phases of the software development cycle. Detailed procedures and tools to support the methodologies have been produced. Both AECL and OH are in the middle of projects that implement these methodologies for the first time. Ontario Hydro is designing a digital trip meter for retrofit to the Pickering B station while AECL is using these new methodologies in the new Wolsong units 2,3,&4 (in Korea) for the safety shutdown systems.

We have applied this framework concept for software in the lower categories of safety criticality as well. Our categorisation guideline assumes 4 categories of software criticality from category 1 (safety critical) to category 4 (no safety impact). To date we have prepared high level standards for categories 2 and 3. These standards are aimed at the more complex systems that typically comprise category 2 and 3 systems in the station, such as reactor control (category 2) and plant display systems (category 3).

One area that is quickly emerging is that of qualification of pre-developed software. As the software industry evolves, it is becoming clear that for the future, we will be using commercial off-the-shelf software more and more. Since we are putting such emphasis on the software engineering processes for our developed software we must also determine what constitutes adequate quality for pre-developed software. It seems that although this area is one of interest to many groups, very little consensus on how to accomplish the task has been attained within the community. In the interim, to support projects such as the Bruce Rehabilitation Project, we are issuing a guideline for trial use that suggests techniques for selection of pre-developed software as well as describing qualification criteria such as strong usage history, fitness for use, maintainability, etc. Currently we are gaining experience in the application of the guideline with the intent of updating the guideline to reflect our experience gained. It is hoped that over time a standard for qualification of pre-developed software can be produced that provides objective acceptance criteria, but this has proven difficult to attain to date.

In order to achieve a national approach for software engineering in the nuclear industry, we have initiated a Canadian Standards Association (CSA) committee to produce a set of CSA standards under the N290.14 grouping. Participants include all the nuclear utilities, AECL CANDU, the AECB, GE Canada, and other interested parties such as nuclear industry software consulting companies. Several meetings have already been held and it has been decided to wait until the second revision of each OASES standard is produced before issuing them as CSA standards.

Participation in OASES has evolved from our beginnings with just AECL CANDU and

Ontario Hydro to now include AECL Research at the Chalk River Labs which plays a major role in the development and maintenance of our software development tools as well as taking a leading role in areas such as reliability testing. Other companies such as GE Canada have also been involved in OASES due to their involvement with the fuel handling control computers and retubing projects.

ENGINEERING OF SAFETY CRITICAL SOFTWARE

The high level standard for safety critical software, category 1, documents our approach to engineering of safety critical software. One of the main purposes of producing the standard was to document a set of acceptance criteria for safety critical software, so that we could reach consensus between ourselves and the AECB on what constituted "good enough". AECB comments on drafts of the standard were addressed and the AECB has accepted revision 0 of the standard as the basis for development of the safety critical software for the Pickering Digital Trip Meter project within Ontario Hydro.

The fundamental principles which were the basis of the requirements in the high level standard for safety critical software are described below.

A planned and systematic Software Engineering Process must be followed over the entire lifecycle of the software. Both the original development and any revisions must be treated as an integral, continuous and interactive process. Any changes must be verified to the same degree of rigour as the original development.

In order for software to be "engineered" it must be developed according to a planned and systematic process that has been designed to produce software of the required quality. In order to maintain the quality, all revisions made to the software until its retirement should also be performed according to a planned and systematic process.

The plan must adopt a specific model for the software engineering process breaking it down into well defined activities, the inputs they require, and outputs they produce. The scope of work for each activity must be unambiguously described. The plan must also specify the approach and methodologies to be used for the activities. Detailed standards and procedures must be in place before work can commence.

The plan must specify suitable facilities, tools, and aids to be used for the software engineering process and identify the necessary support personnel required to maintain and manage the facilities.

The plan must be revised when there are major changes to either the software scope of work or to the organizational structure.

Periodic audits must be performed to verify that the software engineering process is being conformed to and that the product is of acceptable quality.

Documentation must be prepared to clearly describe the required behaviour of the software using mathematical functions written in a notation which has a well defined syntax and semantics.

Mathematical functions provide a mechanism for completely, precisely and unambiguously specifying the required behaviour of the software.

By having the behaviour of the outputs specified by mathematical functions the following advantages are achieved:

- i) the requirements will be more complete since input domain coverage can be checked to determine if the required behaviour of the outputs has been specified for the complete, valid range of each input and for all combinations of inputs that affect each output.
- ii) the requirements can be uniquely interpreted since the notation has a well defined syntax and semantics. This will avoid misinterpretation of the requirements by the various users of the requirements specification; i.e. the specifier, the designer, the verifier, the validator, and the auditor.
- iii) the mathematical representation facilitates the use of mathematical verification techniques that allow the design to be transformed into a mathematical function form for direct comparison with the requirements. This provides an effective mechanism for demonstrating that the software conforms to its requirements.

The outputs from each development process must be reviewed to verify that they comply with the requirements specified in the inputs to that process. In particular, those outputs written using mathematical functions must be systematically verified against the inputs using mathematical verification techniques or rigorous arguments of correctness.

Stepwise refinement is an important concept not only because it allows the developer to tackle several more manageable problems instead of one large one, but also because it allows the verifier to more effectively perform review. It is very difficult to review software listings to determine if they represent a solution to the right problem. It is much more manageable to first verify that the requirements are correct, then verify that the design description correctly satisfies the software requirements, and then, finally, verify that the code satisfies the design description.

As mentioned earlier, mathematical verification provides an effective mechanism for demonstrating conformance to specifications. Mathematical verification is most effective when it is integrated into the design process.

This means that the requirements specification uses mathematical functions to specify the required behaviour of the software system in terms of system inputs and outputs, the design description uses mathematical functions to specify the required behaviour of each program in

terms of its program inputs and outputs and that the code provides a representation of a mathematical function of the program outputs in terms of program inputs.

The design description can therefore be mathematically verified against the requirements specification and the code can be mathematically verified against the design description.

The structure of the software must be based on "Information Hiding" concepts.

Information hiding is a software design technique in which the interface to each software module is designed to reveal as little as possible about the module's inner workings. It was developed by Dr. D.L. Parnas in 1972 and is described in reference [3]. In this way, if it is necessary to change the functions internal to one module, the resulting propagation of changes to other modules is minimized. This results in modules that are loosely coupled or independent of each other as much as possible.

Loosely coupled modules are easier to review since the reviewer can focus on the module under review instead of the dependencies between several modules. Also, loosely coupled modules mean less interaction between various software developers which results in higher productivity.

Those functions of the system which are most likely to change and the form those changes are likely to take should be identified by the engineer responsible for preparing the system requirements. Based on this information the software engineer designs software to localize and isolate these functions to facilitate the potential changes.

Both systematic and random testing must be performed to ensure adequate test coverage.

Testing is the process of executing a program with the intent of finding errors. Errors may consist of non-conformance with the requirements specification or the design description, or incorrect object code produced by the compiler or assembler. It is impractical to exhaustively test the software because the number of test cases to provide every input combination (exhaustive input testing) or to cause every path through the software to be executed (exhaustive path testing) is too large. The question is therefore "What set of tests, less than exhaustive tests, constitutes an adequate set of tests?".

Adequate test coverage must be accomplished through a combination of systematic white-box and black-box test cases along with randomly generated test cases. The design of the black-box and white-box test cases should be such that a predefined coverage is accomplished that should uncover many of the most common errors. The purpose of the random test cases is to improve the effectiveness of the test cases by compensating for false assumptions and biases of the tester.

Testing must be composed of different overlapping activities which use the code, the design description, the software requirements and the system requirements as a basis for establishing

test cases and reviewing test results. Test reports are required to demonstrate that adequate test coverage has been achieved successfully.

Reliability of the safety critical software must be demonstrated using statistically valid, trajectory-based, random testing.

As discussed above, it is not practical to exhaustively test software. As a result the software will be placed in service with the knowledge that it may encounter a combination of input conditions never tested for and for which the software may fail to meet its requirements. It is essential that this degree of uncertainty be quantified so that it can be shown to be consistent with the reliability requirements of the overall system.

It is possible to use random testing as a means of determining the probability that a software product will encounter an input sequence that will lead to errors. From this it is possible to determine the number of random test cases required to demonstrate a specific reliability value. [2]

To use random testing as a means of measuring software reliability it must be trajectory based and statistically valid. To this end the following requirements must be met:

- i) For each test case, the initial values of the test inputs must be randomly selected from the set of input values for which the system does not take action (i.e. the test starts in a normal system state).
- ii) For each test case, the final values of the test inputs must correspond to values for which the system must take some action (i.e. the test ends with the system in an accident scenario state).
- iii) The time period of the each test case must be sufficiently long to ensure that the effects of the retained memory of the software do not invalidate the statistical validity of the tests.
- iv) The behaviour of the test inputs over the test period must be defined by a time-related function which corresponds to the function that the input would assume during an accident scenario. The function must include the effects of instrument response times, signal noise, and any other characteristics that are known about the inputs.
- v) The initial values, final values, and time related function must be consistent with what the software would experience during an actual accident scenario.

Configuration management must be maintained throughout the entire life of the software to ensure up-to-date and consistent software and documentation.

The objective of Configuration management is to identify the configuration of a software system at discrete points in time for the purposes of systematically controlling changes to the configuration and maintaining the integrity and traceability of the configuration over the entire lifecycle of the software. Configuration management of the outputs of the processes must be maintained to ensure that the correct version of each output is being used at any point in time.

The objective is also to control all changes made to the software. Software engineering is iterative in nature since changes to requirements, design, code, and verification procedures occur at many points during the process. Change requests must be formally documented and reviewed to assess the impact of the change requested, to approve or reject the request, and in the event of approval, to decide on the scope of the change to be made, and to issue an approved change request to the appropriate personnel for action.

Configuration management also provides an ongoing analysis of encountered errors which is used as input to the continuous improvement of the standards and procedures.

Audits must be performed periodically to ensure that the software and all development-related processes conform to standards and procedures.

To provide assurance that the planned, systematic software engineering process is being followed, periodic audits must be performed. This is important both with respect to improving the development process and with respect to demonstrating to a licensing authority that the software was developed using a disciplined software development process. Deficiencies and non-conformances must be followed up in a timely manner.

Ongoing training must be undertaken to ensure that personnel have the skills required to perform their jobs.

Since software engineering is a relatively new field, there is not yet a definition of the minimum set of skills that a software engineer must possess. This problem is also complicated by the fact that software is being introduced in areas where personnel are not familiar with the specialist techniques required to develop safety critical software.

Therefore it is necessary that the skills required to perform the various software engineering processes be identified and compared with the skills of the individuals performing the processes. Training must be undertaken to make up for any deficiencies and be tailored to the various software engineering roles (e.g. designer, verifier, validator, auditor).

Verification of the software must be carried out throughout its entire life. All changes to an output must be verified in the same way as the original output.

Between the time software is first released for use and its final retirement it undergoes changes to correct detected errors and to respond to modifications and enhancements in requirements. To ensure that the software does not degrade over time, the level of verification must be maintained. The verification of changes must therefore be performed in the same manner and to the same degree of rigour as changes would be verified during initial development.

Independence of design and verification personnel must be maintained to help ensure an unbiased verification process.

The effectiveness of the verification process is greatly enhanced when personnel other than the designers perform the verification. Independence of the verifiers provides a perspective to the verification that is not biased by the design of the software but is strictly based on the available documentation.

Because verifiers will become intimately involved in the internals of the design and code during the course of performing their work, it is necessary that they not be involved in the final validation of the software against the original system requirements. For this work to be performed objectively, it is necessary that it be carried out by an independent validator whose perspective is strictly from the requirements.

Besides the designer, verifier, and validator, it is necessary to identify a fourth independent role to act as an overseer of the entire software engineering process. This role, known as auditor, will ensure that all standards, procedures, and guidelines established for the project are being followed correctly.

Analyses must be performed to identify and evaluate safety hazards associated with the computer system with the aim of either eliminating them or assisting in the reduction of any associated risks to an acceptable level.

To provide adequate confidence that the safety critical software will operate in a safe manner at all times an analysis must be performed whose purpose is to identify any failure modes that may lead to an unsafe action, and then either eliminate them or, where possible, ensure that the failure mode can be detected and the system put into a safe state.

EXPERIENCE TO DATE WITH THE FRAMEWORK

To date the framework has been successful in meeting many of its objectives.

One of the biggest concerns after the Darlington shutdown system software problems was that the regulatory risk associated with using digital technology in safety systems would be too great to allow their use in future projects. This would have been unfortunate since digital technology offers some real benefits. Having reached initial consensus on the standards and procedures, by virtue of ongoing review by the AECB, has alleviated concerns to the point that the Pickering Trip Meter project and the Darlington Re-design project have received management approval within Ontario Hydro. AECL CANDU is proceeding with the development of two shutdown systems for the Wolsong 2 stations. Without the OASES framework, it is questionable if the risk associated with any of these projects would have been acceptable.

We have found that the use of mathematical functions in a tabular format to be of great value in our specifications. One of the major concerns with their use was their degree of understandability for safety, system and human factors designers who would have to review them. Experience has shown that the reviewers have found them to be reviewable and an effective way to capture requirements. Several areas have been noted for improvement in this area and are being investigated for incorporation in the next revision of the procedures.

The specifications have been useful in finding areas of incompleteness in the higher level specifications and for assisting in identifying areas of complexity. The tools have proved useful in automating a number of consistency and completeness checks that can be performed on the specifications.

Since most of the issues for Darlington had been centred on the software, our concentration has been on software engineering. We have found that more of our emphasis needs to be placed in the area of systems engineering. AECB expectations in this area are focused on ensuring that the safety and system requirements are accurately captured, and that the system engineering process adequately demonstrates that they have been met. Another area of concern is the adequate incorporation of human factors engineering in the system level engineering process.

AECL CANDU has also found that the framework can also be applied to systems that use a block function language. One of the two Wolsong 2 shutdown systems is based upon an ABB platform that is programmed in a block function language.

The framework has also been applied to form the basis of the Software Quality Management and Assurance program for our Bruce A station, which is currently undergoing a major rehabilitation. The framework has provided a common, rationale basis for the degree of rigour being applied to software in the various applications covered by the rehabilitation.

CHALLENGES AHEAD

We have made considerable progress since OASES started; however, there are a number of challenges that need continued attention. The fact that software engineering is a relatively new frontier with advances being made constantly requires we keep abreast of these activities and integrate them into our methodologies on a selective and practical time frame.

Secondly, we need to be able to address evolving regulatory requirements. The AECL have tended to make strong suggestions about software engineering techniques that appear promising and we need to be aware of these approaches and make informed defensible judgments about their practicality, and suitability for our applications.

Thirdly, we must ensure that our methodologies are cost effective. A lot of the newer requirements are aimed at achieving a higher degree of demonstrability of having produced high reliability software. We must ensure that any techniques applied significantly add to the achieved safety/reliability of the system, or significantly add to the degree of assurance of having achieved safety/reliability. If the processes that we must use for safety critical software are not cost effective, the utilities will end up substituting non-appropriate technology as a misguided means of risk avoidance, and possibly missing opportunities of increasing the safety of systems.

We believe that through the OASES or team approach across members of the nuclear engineering community, we are able to meet the above challenges. Our concept of a framework of standards has been accepted as the basis for developing a set of N290.14 standards for the CSA. The OASES Standard For Software Engineering of Safety Critical Software has been received positively by international organisations such as the National Institute Of Science and Technology in the United States. Our methodologies, procedures, guidelines, and tools are being applied in projects within Ontario Hydro and AECL. We have been keeping abreast of international developments in software engineering and maintain contacts with key gurus in Canada, the United States, and Europe. By trying promising approaches we are able to select appropriate techniques and discard unsuitable ones. The biggest benefit of the team approach is that the costs are shared. Software engineering for safety critical systems has too big a price tag for one group alone. By working together, we have been able to keep our individual costs reasonable, and have been able to reach a consensus on an industry approach to software engineering.

CONCLUSIONS

Software has unique problems associated with its use in systems requiring high reliability. The problems are not unique to the nuclear industry. Ontario Hydro and AECL have applied their knowledge and experience in the design of computer systems for control to address these new issues by forming a joint group called OASES. The OASES group has created a framework of standards covering all categories of safety criticality. As we move forward towards our goals, we face a number of formidable challenges which we feel we can overcome through the team approach we have adopted across the nuclear industry.

REFERENCES

- [1] IEC 880 "Software for Computers in the Safety Systems of Nuclear Power Stations".
- [2] Parnas, D.L., Schouwen, A.J., and Kwan, S.P., "Evaluation of Safety Critical Software", Communication of the ACM, Vol. 33, Number 6, June 1990, pp 636-648.
- [3] Parnas, D.L., "On the Criteria to be Used in Decomposing Systems into Modules", Communications of the ACM, Vol. 15, No. 12, December 1972, pp. 1053-1058.

3.4.1 Questions: Mr. Paul K. Joannou

QUESTION: GORDON HUGHES (Nuclear Electric): What reliability values are associated with your software categories? What is the combined non-reliable claim for the two diverse systems on Darlington for the most critical fault?

MR. JOANNOU: We have a guideline for categorizing the reliability value associated with the software. For a system in the safety-critical software category, the associated system reliability requirement is 10^{-3} , plus having the most significant category of consequence of error. We try to demonstrate that the software has a reliability of 10^{-4} and hence is not a major contributor to system unreliability of 10^{-3} . I guess the answer is 10^{-4} .

MR. HUGHES: Does that have units? 10^{-4} what?

MR. JOANNOU: Well, that's a probability of error upon demand.

This is combined reliability shown for the two diverse systems. I've tried to get this answer and I keep getting different answers from different safety analysts. One of the questions is whether or not you claim that the systems are so diverse that you can just get a 10^{-6} system out of two 10^{-3} systems?

The requirement for the two shutdown systems together is 10^{-6} reliability. We've tried to demonstrate that the software is 10^{-4} in each one. I guess nobody sat down and showed me a very clear analytical equation that maps that, but I guess the overall reliability for the two shutdown systems is 10^{-6} .

QUESTION: DR. LANCE A. MILLER (SAIC): I understand that you set your threshold at 10^{-4} . How do you assess the reliability of some component of new software? If you have a module, what do you do? Do you use a MUSA model or some other kind of software reliability model to estimate the reliability of the new software module, that you don't have data on?

MR. JOANNOU: We don't try to assess the reliability of a module of software. All we try to establish is for something like a trip system what is the system reliability requirement and then given the role of the software what's the equivalent software reliability requirement on the software within that system. In terms of the degree of rigor we apply, we use the reliability plus the consequence of error to determine what degree of rigor to apply, a Category I standard of rigor, or Category II, et cetera. Within Category I we have a particular activity which tries to demonstrate the achievement of a particular software reliability target in terms of this trajectory-based random testing. But it's not done on a per-module basis; it's done on a software system basis.

4 DIGITAL SAFETY SYSTEMS FOR NUCLEAR POWER PLANTS

Mr. A.L. Sudduth (Duke Power), Dr. John C. Cherniavsky (National Science Foundation (NSF)), and Dr. Lewis F. Hanes (nuclear industry independent consultant) discussed the problem of replacing an analog with a digital control system in a NPP from the different perspectives of hardware, software, and human factors. These different views of the problem resulted in different definitions of the problem and led to different approaches to solving the defined problem.

Mr. Sudduth provided a hardware view of replacing an analog with a digital system from the perspective of a senior engineer involved in digital upgrades for control systems in fossil fuel plants. His concerns are to maintain or improve the level of safety available from the new system, but also to minimize the downtime for the plant. Mr. Sudduth proposed several alternate hardware designs using proven components to improve safety. To minimize plant downtime, Duke Power used a complete control-room simulator to speed up the process of testing the new system in a setting which closely approximates the actual control room and for training personnel to work with that new system. Digital upgrades for Duke Power now require only three months of downtime.

Dr. Cherniavsky provided a software view of the problem of installing a digital upgrade of an analog system from a research perspective. He discussed the research supported at NSF in the High Performance Computing Communications Initiatives and placed that within the larger context of NSF software research support. The NSF has a continuing interest in topics related to safety-critical software. In the past, NSF has funded formal methods research and, in the future, may develop a Center for Software Safety studies.

Dr. Hanes provided a human factors view of the problem of converting a control system from analog to digital displays. Dr. Hanes raised issues across the entire system lifecycle including new requirements that might be imposed on these systems, a better understanding of anthropometrics and biomechanics, designing these systems to provide the information the operator needs when he or she needs it, and intelligent aids to support the operator's decision making after the system is in operation. Dr. Hanes drew on experience in other industries to identify ways that a digital system can improve crew performance, enhance plant safety, and avoid problems encountered in those industries.

Each speaker's discussion of methods for addressing the problems reflect their experience and expertise in a specific technology. Mr. Sudduth viewed the conversion of control systems as essentially a solved problem from a hardware perspective. The hardware is available and in use in fossil plants which have an established conversion process. The use of digital control systems has supplied substantial data on the human factors issues in converting plants, but these issues are not entirely resolved and still require more research according to Dr. Hanes. In contrast, software engineers are just beginning to grapple with some of the issues in developing safety-critical systems according to Dr. Cherniavsky.

4.1 Hardware Aspects of Safety-Critical Digital Computer Based Instrumentation and Control Systems: Mr. A.L. Sudduth

Hardware Aspects of Safety Critical Digital Computer Based Instrumentation and Control Systems

A. L. Sudduth, Engineering Consultant
Duke Power Company
Charlotte, NC

INTRODUCTION

During the last 20 years, instrumentation and control systems based on digital computer technology have come to be used widely throughout industry. Included are applications that are of a safety critical nature -- that is, where the consequences of failure of a device or system could have substantial adverse impact on the public. The successful application of digital computer technology in the aerospace and process industries for monitoring and control, and the benefits that this technology offers when compared to the analog technology of 25 years ago, require that consideration be given to the more widespread application of digital computer technology in US nuclear stations as obsolete monitoring and control systems are replaced.

The purpose of this paper is to examine hardware aspects of safety critical digital control and instrumentation systems. It will discuss how to achieve adequate measures of system reliability and how to ensure that the required level of reliability is achieved. Special techniques for design of the hardware architecture of a digital computer based monitoring and control system that are able to make the system single or even double failure proof are discussed. Examples of hardware architectures which achieve high reliability are given, including those currently in use in US nuclear stations or expected to be applied in the future.

COMPARISON BETWEEN ANALOG AND DIGITAL MONITORING AND CONTROL SYSTEMS

Nuclear stations in the US were constructed with process control systems based on analog hardware. In particular, the safety critical

portions of process control, the reactor protection systems, were generally of electronic analog design. Examples include the Westinghouse 7300 series and the Bailey 860 series. Both of these architectures were used in industrial process control and in fossil fueled power stations as well as nuclear stations. The safety related portions of sequential control (such as diesel generator load sequencing) was implemented almost universally with relay logic, using discrete relay devices hardwired together. In addition, the non critical portion of the control in most stations was originally analog, and pneumatic for the most part, particularly for stations constructed in the 1960's and early 1970's. Digital computers in nuclear stations were used solely for monitoring purposes (the Operator Aid Computer, or OAC) or for control of specialized systems such as the main turbine. Though there were limited exceptions, such as the Combustion Engineering Reactor Protection Computer system, digital systems were not used for any important control functions in nuclear power stations in the US.

For the non critical portions of nuclear station control, particularly for closed loop control of secondary systems, that situation is changing rapidly, as pneumatic instrumentation and control are being replaced by distributed digital systems such as the Bailey Net-90 and Westinghouse WDPF. Digital systems are also beginning to be applied in systems that are "important to safety," such as the control of feedwater in both PWR's and BWR's. The two areas of nuclear station control that have been least affected by modernization are the safety critical systems, such as Reactor Protection and relay based sequential control, and the operator interface in the control room, which remains a benchboard and vertical board system with hard controls and indicators.

This section will examine some of the major hardware differences between analog and digital implementations of process control and monitoring systems, and some of the differences between a digital computer based process control system and more common computing devices, such as desktop computers. Of course, the underlying difference between analog and digital technologies is in the manner in which physical variables (process states) are

represented in the system. In an analog system, these variables appear as voltage levels or current levels, with values in electrical units directly proportional to the quantities they represent. Thus a process temperature might have a range of 0-500 °F, and appear in the system as a value between 0 and 5 volts. In contrast, digital quantities are encoded numbers. The analog value is converted to a digital value by a sampling system, and the digital value is represented as a binary number, as are all values in a digital computing device. Rather than varying smoothly over a range, digital representations of process variables are discontinuous.

In the earlier days of computing, the representation of analog quantities by digital numbers using sampling was of concern because of errors due to quantization and roundoff. Because of industry experience in the representation of numbers in digital form, such as the standards for floating point numbers developed by IEEE, and the rapidly advancing use of 32 and 64 bit architectures in inexpensive digital systems, problems with issues such as quantization error or roundoff are practically non-existence in current digital systems. Another area of concern, that of adequate sampling rates to avoid aliasing errors, while it may apply in certain aerospace applications, is not a problem in power stations because underlying rates of change of process states are not rapid enough to require unusually fast sampling rates.

A more significant distinction in architecture involves modularization issues, including intermodular communication. If we look carefully at a typical analog control system, we would find that even though the system is basically modular, almost every module has been customized in some way. This is because the application programming and tuning of such a system are explicit in the design of the analog circuit. Thus by looking at the schematic of an analog control module, we can tell its function exactly. In contrast, a typical digital control system consists of many modules of only a few types. If we examine the schematic of one of these modules, we would have no clue as to its specific function in the control strategy of the process.

Control strategy appears in firmware and software programming of a typical digital processor. Rather than performing a single function, as do most analog hardware modules, a digital processor performs many hundreds of arithmetic operations involving tens to hundreds of separate variables or process states. To determine the function of a digital processor module, it is necessary to examine the software programming (applications code) for that module. Digital systems also have much more interprocessor communications than did typical analog systems. One consequence of the interprocess communications feature is that much more complex strategies may be implemented in a digital system, such as multivariate control of a process. Multivariate control is seldom used in analog systems. The tradeoff in the additional capability available in digital systems is a significant increase in system complexity. It apparently this complexity exceeds the comfort level of many regulators, as the difficulties in acceptance of digital technology in nuclear stations indicate.

A typical hardware architecture for a digital control system processing channel is shown in Figure 1. The input and output connections to the process sensors and actuators are handled by the I/O processing module. In this module, such tasks as engineering unit conversion, signal validation, and simple filtering and compensation are carried out. The data from the I/O processor appears on the I/O bus, which is generally implemented as a standard hardware bus. The processor module receives input from two sources — the I/O bus and the data highway, processes those inputs as dictated by its application software programming using its local memory storage as a conventional computer does, and places its output on the data highway and on the local I/O bus, where applicable. The data highway controller connects the processor to the highway, providing the required protocol conversions, since the data highway is generally a Local Area Network. Not shown in this picture are support systems such as power supplies. A typical system consists of 10 or 20 of these subsystems, all identical in hardware. When configured into a system, additional hardware functions are added, such as special interfaces to hardware devices (such as serial data links

or connections to Logic Controllers), additional computing support for off-line (not real time) functions, and the human-computer interface connections for operators and management personnel. A typical layout of such an integrated system is shown in Figure 2.

Intermodule communication in analog systems is carried over wires, with one wire per variable. Using an electrical meter, one may measure the electrical analog value on any of these wires and thus the value of the variable being carried by the wire. In contrast, intermodular communication in digital systems is handled by hardware bus architectures, such as MultiBus, and data highways carrying computer network traffic, such as Ethernet or Token Bus. The communication busses and highways carry multiple variables; indeed a typical data highway (single wire) carries all of the input-output values that any of the modules in the control system needs to do its work, perhaps as many as 10000 variables.

There is a difference in the manner in which analog and digital systems interface with the real world. Instruments and actuating devices are for the most part still analog devices, thus the interface to an analog system is more natural. To talk to these devices a digital system must use analog to digital conversion (A/D or sampling) for its inputs and digital to analog (D/A) systems for its outputs. Thus digital systems add additional levels of complexity in the signal path. The opposite is true of the human interface. It is much easier and cheaper to create a well-engineered operator interface for a digital system than to develop a well-engineered analog interface using discrete hardware devices. The discrete devices that make up a typical nuclear station benchboard require too much physical area to be as effective in conveying information as a well-designed interface based on CRT screens. Despite considerable effort in human factors engineering over the years since the TMI incident, deficiencies in control room design noted by the Kemeny Commission have been addressed but not eliminated due to the lack of flexibility created by the dependence on discrete devices. The CRT screen, and a pointing device such as a touchscreen or mouse, has proven to be very effective in providing an innovative human interface that contributes to improved

operator performance, while retaining a degree of adaptability far greater than that of current nuclear station control rooms.

We may compare a digital control system to a network of desktop computers. Within each system, each processor receives a set of inputs, processes these into a set of outputs, and uses data storage, such as RAM memory. In both systems, the processor executes two software programs — an operating system program and an application program. Generally we think of the application program as being under the control of the operating system program. One of the most obvious hardware differences between networked desktop computers and a distributed digital control system is the need for a real time information interface. The collection of inputs and outputs is gathered together into a communication or data highway. The structure of this highway may be completely hardwired, such as a hardware bus, or it may be implemented using a computer network through the addition of a suitable network interface and an extension of the operating system. Data must pass among processors connected to this highway in a predictable manner, so that real time operation can be assured.

A second needed hardware change is an upgraded clock which operates at the system level. Because monitoring and control systems are real time computing systems, and the actions of distributed processors must be synchronized among the processors, the clock is much more important in these systems than in desktop computers. If we were to place a priority on the tasks that must be accomplished in order for our system to maintain operability, ensuring the integrity of the real time clock would be at the top of the list.

The third needed change is the addition of more sophisticated hardware interrupt capability. Hardware interrupts are used in many desktop computer systems, but generally only to service peripheral equipment, such as disk drives, or to indicate irreparable errors in processing, such as division by zero. In our monitoring and control system, we need some hardware interrupts which can be used to change the task or mission of the system when required by external influences. For example,

if a system failure caused a previously standby controller to take over for the primary controller, it would be accomplished in many architectures through the use of hardware interrupts. At the system level, if an emergency condition were detected, a hardware interrupt could be used to guarantee that processors switch to the tasks required to respond to the emergency.

There are other changes that we might include in order to speed up the processing in our system, some of which are beginning to appear in high end desktop computers. The inputs and outputs could have caches and buffers to adjust for their slow speed. We might add additional reliability features such as parity checked memory (not used on all desktop computers) or multiple processors. But in the end, the basic architecture of a distributed digital system for monitoring and control is not significantly different from the architecture used in today's networked desktop computer systems. The microprocessors are generally the same (Intel 80X86 or Motorola 680X0), communications channels are the same (RS-232, IEEE 802.4), and peripherals are similar (floppy and hard disk drives). Most of these systems provide interfaces to commonly used minicomputers, such as the VAX family, or to the newer RISC based processors, such as the Sun SPARCStation. In summary, there is nothing exotic or unusual in the basic digital control system hardware architecture, nor does it take extraordinary skill or effort to configure or maintain these systems. A detailed discussion of digital hardware architecture is contained in Williams (1984).

ASPECTS OF HIGHLY RELIABLE DIGITAL HARDWARE

High reliability implies that equipment operates free of faults and failures. We use the term fault to refer to a point type of failure occurring in a localized area of a larger system. An example would be the failure of a specific component on a circuit board that renders the board incapable of performing its function. A failure is the consequence of a fault -- some aspect of the digital system is unable to complete its mission, resulting in a possible adverse impact on the controlled process. This

might be the initiation of an inappropriate control action or the failure to initiate a required action. Our overall objective in achieving high reliability is the avoidance of failures. It is obvious that it is impossible to design electronic circuitry that will be free from faults.

Our ability to achieve the uninterrupted monitoring and control of a process involves a combination of fault avoidance techniques and the achievement of the required degree of fault tolerance. The next section will discuss these twin aspects of reliable hardware systems.

FAULT AVOIDANCE

Fault avoidance is important for digital systems as it was for analog. There is a need to achieve the highest level of reliability of individual components and assemblies to minimize the challenges to fault tolerance features.

The techniques for achieving satisfactory levels of hardware reliability in digital systems are similar to those used for high reliability analog systems. These techniques include:

1. Selection of high reliability components - - digital components are available with certified levels of reliability based on sample testing. However, the specification of high reliability components for digital systems is made more complex by the complexity of individual components themselves. If VLSI is used in the fabrication of certain components, testing for quality assurance is more difficult than it was for commonly used analog components. There are many standard VLSI components on the market, such as the established microprocessor families and their associated support microcircuits, for which high reliability versions may be obtained. Thus at the component level, the use of custom fabricated components should be minimized, and a quality assurance program established for all components. Techniques commonly applied to analog components, such as required burn-in periods, are applicable.

The requirement of high reliability components for digital systems must be carefully evaluated as part of the overall reliability program in

order to avoid unnecessary costs. If fault tolerant architecture is to be used, then part of the incentive, and part of the benefit, for using expensive high reliability components is eliminated. Analytical techniques that establish quantitative measures of system reliability should be applied to ensure that the additional cost of high reliability electronic components can be justified by a measurably significant improvement in system reliability.

2. Application of components suitable for the operating environment — here again the techniques of highly reliable design are similar to those used for analog devices. Standards have been established for the operating environments to which electronic components and assemblies are subjected, and established standards require that appropriate qualification testing be performed. Qualification of digital systems for environment must include all relevant adverse environmental conditions, including interference from electromagnetic fields.

3. Testing of components, subassemblies, and integrated systems — in addition to qualification testing of components described previously, integrated system level testing is required for digital systems because of the integrated nature of the digital architecture. The modular nature of digital systems, the distribution of operations and functionality, the complex nature of interprocess communication, and the interaction between hardware and software all dictate that an integrated test of the system is required.

4. Establishment of installation procedures — the routing of system cables and establishment of proper system grounding are crucial to the integrity of the system and avoidance of common mode faults. There are differences between the typical vendor test floor and the environment of a power station. This need is best met by a carefully designed installation that is described in detail for the organization that will do the installation. The provision of highly reliable auxiliary systems, such as uninterruptable power and cooling, has the same importance for digital control systems as they did for analog systems.

There are some unique features of digital systems which should be considered when

hardware reliability is discussed. Failure modes of digital systems vary, depending on the architecture (e.g., typical failure modes of TTL logic are not the same as those of CMOS). Though digital components tend to follow the first part of the classic failure curve (Figure 3), it is not clear that there is a wear-out period as in the classical sense. Obsolescence appears in most cases prior to true wear-out response of digital hardware.

FAULT TOLERANCE

In spite of all of the best design and testing techniques applied for high reliability, the long term operation of the system fault-free cannot be achieved. The system must therefore be designed to be tolerant of hardware faults. The design of fault tolerant digital hardware has been an active research issue for many years, and this paper will provide only an overview of some of the latest work. Papers by Rennels (1984) and Siewiorek (1991), and the book by Lala (1985).

Fault tolerance is the ability of a system to experience some type of fault, to avoid having that fault propagate into a system failure, and to be able to achieve its mission in spite of the fault. Thus the fault tolerance required for a system is very much a function of the mission and the nature of the faults expected, as well as the accessibility of the system for repair. For applications in power stations, we can restrict our discussion to real time process monitoring and control systems, whose mission is to protect the health and safety of the public and the investment of the owners, and which is reasonably accessible for repair. In that case, we can describe the following required features of a fault tolerant system to be used in a safety critical application:

1. The system must provide complete coverage for hardware faults — Coverage is defined as the fraction of all possible faults to which the system may be subjected for which the system is able to continue to carry out its mission following occurrence of the fault. Complete coverage implies that the mission continues for any hardware fault occurring independently and randomly of any other fault.

2. The system must provide effective fault masking and recovery -- A fault will eventually have an effect on the system, necessitating some combination of masking and recovery. Fault recovery is the process of ensuring the continuation of the system mission; fault masking involves preventing the effects of the fault from propagating in a manner that affects the controlled process. Complete masking implies that no recovery is necessary, but if the fault cannot be completely masked, fault recovery, which requires some amount of time, must be performed by the system. The time available to recover from a fault varies from application to application. If the system is controlling a loop with very fast dynamics, the loss of control for a few hundred milliseconds may be enough to produce catastrophic results. Luckily, power station systems do not have such fast dynamics, and a delay of a few seconds is not generally critical. However, avoidance of unnecessary alarms and transients would dictate that fault masking is preferable to fault recovery, the fault masking scheme should be as effective as possible consistent with costs, and the time for recovery should be kept short.

3. The system must provide unambiguous diagnostics -- since the effects of faults will be masked from the process or contravened by recovery actions initiated by the fault tolerant features, there must be some way to inform the operator of the occurrence of a fault so that repairs may be initiated.

4. The system must provide graceful degradation -- since the time to repair a fault will not generally be zero, the system will be operated for periods of time in a faulted condition. The degree of reliability of the system in a faulted condition must be adequate to ensure that its mission is not compromised. It is also highly desirable from the owner's standpoint that the system be able to continue normal productive operation during this time, as the economic consequences of shutdown, and the plant degradation associated with frequent shutdowns, are to be avoided. A desirable feature would be the continued performance of the mission, though perhaps with degraded reliability, after the occurrence of two independent faults.

5. The system must be repairable without compromising the mission -- the need for continued performance of the safety critical functions of the station, as well as the need for continued production, dictate that repairs be achieved while operating. The system must then provide for the graceful reintegration of the repaired unit into the process and the re-establishment of the previously unfaulted condition.

These features of the fault tolerant system have been achieved though the work of a number of researchers, and successful fault tolerant systems have been constructed for real time process control and monitoring. A subsequent section will review several of these systems -- two classical systems developed from research with hardware based fault tolerance, and two systems currently in use or under development by utilities for power stations -- after more background information on fault tolerant architectures is discussed.

HARDWARE ARCHITECTURES FOR FAULT TOLERANCE

We previously discussed fault avoidance as a technique to improve the reliability of a digital system and listed some of the desired features of a fault tolerant system -- wide coverage, minimal recovery time, hot repairs, etc.. We will now look at hardware architectures which implement fault tolerance. The important aspects of fault tolerance for which this architecture must be designed include fault detection, diagnosis, assessment, reconfiguration, repair, and return to service.

In a fault tolerant architecture, faulty information processing channels must be detected before they cause or contribute to a system failure. One method for detecting faults is to augment the application program or operating system so that the processor executed some form of self testing program periodically. Such features are implemented in a very limited way on a typical desktop computer; we would need a more sophisticated program in a control and monitoring system processor. For example, we might devote a small fraction of every cycle to the execution of a limited self checking program that could verify the

integrity of memory and the status of communication channels. Indeed, just the ability to execute the operating system and application program over and over within the time allocated for each cycle is itself a test of system integrity, and a special circuit called a watchdog timer has been used in some applications for this purpose.. A module of a digital control system that executes some form of self diagnosis periodically is called a Self Checking (SC) module.

The problem with a channel containing only a single self checking module is that if the module fails the test, indicating that it contains a potential fault, there is nothing with which to replace it. Such a system is not fault tolerant; the best we can hope is that it quits trying to influence the process once it determines that it is faulty and turns control over to the operator. We also have the problem with a single module that its failure mode may not lead to a clear decision on its integrity. It may be so faulty that it thinks it is OK and continues to try to execute. The result could be inappropriate control action or failure to take a needed action. It is obvious therefore that to achieve hardware fault tolerance, we must provide some form of hardware redundancy. There are many methods for achieving fault tolerance through hardware redundancy. This paper will examine four of the most common.

Dual Redundancy

Dual redundancy implies the addition of a second processor to a control channel. The second processor provides the backup needed to ensure fault tolerance if one of the processors fails. There are several ways of configuring such a system. One commonly employed method is to use two self checking processors, one considered to be the master and one the slave. Both processors run identical application and operating system programs. The output of the master processor is normally accepted by the downstream units. Periodically both processors run their self checking diagnostic programs; if the master fails the diagnosis, then the slave takes over. The basic idea of dual redundancy is shown in Figure 4.

There are several variations of the basic dual redundant architecture. Some of these are shown in Figures 5, 6, and 7. If the two processors are running synchronously (that is, using the same clock) then it is possible to have a third unit that compares the outputs. Upon detection of a discrepancy between the outputs, both processors are ordered to execute diagnostic software. The arbitrating unit then selects which processor to use and sends a message to the system that a processor failure has occurred. Comparison may also be done by routing the output of each processor to the input of the other processor, letting them detect any discrepancies. The processors may then freeze their outputs momentarily while executing diagnostic software, then the processor that decides that it is operable may proceed. Other variations are given in the literature cited in the references.

All of the dual redundant hardware architectures have key features in common. Such systems are limited in their ability to mask the effects of the fault. The degree to which the process is affected by a fault depends on the ability of the system to identify the faulty processor and switch in the output of the operable processor quickly, and to dispose successfully of any processing work that fails to be completed while the switchover is going on. This makes the system susceptible to some situations in which it will not be single failure proof. For example, if the fault is transient or intermittent, self diagnostic software may not detect the presence of the fault and identification of the operable processor may not be possible. When control is transferred from a faulted processor to an unfaulted one, there may be a transient created in the system while the previously bypassed processor is integrated into the system. If an external arbitrating unit is responsible for initiating diagnostics or causing a transfer between processors, it becomes a common mode failure point. Finally, during the time that the faulty processor is being identified and disabled and the second processor is taking over, certain critical actions may fail to occur. Though they may execute successfully in the next cycle, the fault may propagate into a failure.

Triple Modular Redundant

Triple redundant system (also called triple modular redundant or TMR) use three identical processor channels running in parallel. This is a very common form of hardware redundant system and is widely used in safety critical digital control and monitoring systems in aerospace applications. In analog form, it appears commonly in current nuclear station protection systems. The use of triple systems is becoming more widespread as the cost of computing hardware has decreased. The basic idea of TMR is shown in Figure 8.

In a TMR system, three processors running the same application program (though not necessarily the same software) in parallel each generate outputs which should be nearly identical under unfaulted conditions. These outputs are compared by a voting or auctioneering circuit and a single one of the outputs is selected to be passed on to subsequent units. If one of the processors begins to deliver outputs which are different from the other two due to the presence of a fault, the arbitrating circuit will select one of the two unfaulted signals to pass on. The effects of the faulted processor are thus masked from the rest of the system, and the mission continues with no discontinuity. One of the simplest of these auctioneering circuits is called a median select, because it picks the processor output in the middle as being the unfaulted one.

The advantages of TMR over dual redundant systems were studied in a research project sponsored by EPRI (Kisner and Battle, 1992). The results of this study are summarized in Table 1. Since the arbitrating circuit selects from among three signals, under a single fault assumption it will always be able to select the output from an operable processor. Thus a TMR system is truly single fault proof. Since a previously bypassed processor does not have to be integrated into the system when one processor becomes faulty, there is no potential system transient created by the occurrence of a fault. It is not necessary that any self testing software be run in any of the processors because there is no dependence on diagnostic software to detect a fault. This design is so effective at masking a processor fault that attention must be paid in the design to ensure

that faulty processors are identified by some auxiliary means.

A TMR system may also provide for graceful degradation by configuring itself into a dual redundant system once one of the processors has been determined to be faulted. This reconfiguration could be done by having the processors begin to execute some type of self checking code that can arbitrate operability if their outputs begin to deviate significantly.

Quad Redundancy

An even higher degree of reliability may be obtained through the use of four processors. Quad redundancy comes in two forms – one that is an extension of dual redundant systems (a sort of doubled dual redundant system called dynamic quad redundant) and one that is an extension of TMR (called static quad redundant). The disadvantage of such systems is the increase in cost and complexity of supplying four channels and the associated auxiliary units that handle module checking and reconfiguration.

A dynamic quad redundant system uses a pair of dual non-self checking redundant processors. All four processors are running identical programs, so that comparisons may be made between the outputs of each pair. As long as both outputs of a pair are in agreement, that pair is considered operable and its output may be passed to downstream units. When there is disagreement, the pair is considered faulty, and the redundant pair takes over. Note that upon the occurrence of the first failure, it is not necessary that the specific faulty processor be identified, only that the pair that contains the fault be isolated and identified – a more simple determination to make. In a variation of this scheme, self checking processors are used, but the self checking feature is not used unless one pair of processors is found to be faulty through a disagreement in their outputs. The remaining pair then continues to operate, but with self checking enabled and one of the processors becoming the master.

In a static quad redundant system, an additional processor is added to a TMR system. Once a faulty processor has been identified by a significant deviation in its output from that of

the other two processors, the apparently faulty processor is disabled and the fourth processor is substituted for the faulty one. This restores that TMR configuration of the system.

Note that quad redundant systems have the capability to retain some degree of fault tolerance in the face of the occurrence of a single fault, though the effects of the second fault may not be as effectively masked as the first one was. In the case of static quad redundant systems, the degree of fault tolerance is not degraded by the occurrence of the first processor fault; therefore the static quad system may be considered double fault proof. Another advantage is that the time available for repair of the first fault is greatly increased over that of the TMR system without affecting system reliability.

The next section of the report will discuss examples of fault tolerant architectures that have been developed and implemented — two pioneering research efforts sponsored by NASA, and two applications that are currently being used or proposed for nuclear station control.

EXAMPLE FAULT TOLERANT ARCHITECTURES

Fault Tolerant Multiprocessor (FTMP)

An extensive research program over many years produced a fault tolerant central computer design that could be used for control in safety critical situations (Hopkins, Smith, and Lala, 1978). FTMP was developed under the sponsorship of NASA Langley, with much of the work performed at Draper Labs. The predicted performance of this system was an overall failure rate of 10^{-9} failures per hour.

The objective of FTMP was to produce a fault tolerant architecture not dependent on software for management of the fault tolerant features of the design. It was recognized that imbedding the fault management features in the operating system or in the application software (it had to go one place or the other) created nightmares in the validation of the system. In addition, software based fault tolerant schemes are

highly dependent on the ability of software to arbitrate faulted and unfaulted behavior, another rather nightmarish problem. It was concluded that if one could accomplish the requirement to arbitrate behavior by simply comparing results of application programs in hardware, a more robust system would emerge.

The FTMP consists of multiple modules, arranged in triples for redundant calculation; a set of spare modules that may be substituted for a faulted member of a triple; and the associated management software and hardware. A triple may be assigned to any of the system tasks, and may be dynamically reassigned if conditions require. The members of a triple operate synchronously, so that voting may be done on a bit-by-bit basis. The basic architecture is shown in Figure 9.

Intermodule communication is handled by two levels of redundant highway. The lowest level consists of a memory highway, which permits interprocessor communication by shared memory. Each module contains private or cache memory that buffers the communication with the memory highway while providing for local storage. The upper level highway provides for interfacing of processor modules to higher level management functions, such as data input/output, operator interface, and configuration control. Each processor is connected to the highways by bus isolation circuits that prevent the propagation of low level module faults through the highways.

Software Implemented Fault Tolerance (SIFT)

The SIFT program was another effort by NASA, Langley, to study and develop a hardware architecture for a fault tolerant computer system for aircraft control (Wensley, et. al., 1978). The majority of the development work was done by SRI International of Menlo Park. This system depended on hardware for the isolation of faulted processors, which were detected by software programs designed specifically to monitor the health of the processors. The basic architecture of SIFT is shown in Figure 10.

An important advance in thinking about digital system reliability was precipitated by the work on SIFT. Up to that time, researchers had been concerned with failure modes -- that is, trying to isolate and characterize the manner in which digital hardware failed. It was thought that if such low level failure modes could be identified and categorized, techniques for identifying the faults could be made more effective. The SIFT project researchers recognized that the specific manner of hardware failures was less important from a reliability standpoint than the system level effects that such failures produced. Accordingly, the emphasis was on identifying processors which were not producing correct results at the system level, and the isolation of these failures from influencing the behavior of the controlled system.

The general method of fault isolation was through the use of private memory. Each processor was provided with a section of private memory into which its results were placed. Thus no processor failure could corrupt at the system level; at worst, the failure created problems only in that processors private memory. It was not necessary to determine the specific hardware origin of the incorrect results in private memory - it could be processor failure or memory failure - only that the results in a particular private memory were faulty.

The manner of determination of faulty results was through the use of triple redundant processors for each critical task. The results of three processors (the contents of their private memories) were passed to the system through redundant highways and compared by software based voters, and the voters selected a non-faulty set of results to be passed to the system as the control demands and indications necessary for accomplishment of the system level tasks. Once the voter had made a selection, it had also in effect provided diagnostic information, because the signals from the non-selected processors were now suspect. Further software based checking was used to identify a failed processor unit, and the system had the capability to reconfigure itself to replace a faulty unit and restore full redundant operation. The effects of a faulty processor were masked from the system by this triple redundant

scheme, and there was no associated recovery time.

Other important aspects of high reliability digital schemes examined in the SIFT project include synchronous vs. asynchronous operation of the processors. It was found that exact synchronization of tasks was not necessary, but periodic resynchronization of processors clocks was required. This might be characterized as "closely asynchronous " operation. It was recognized that synchronous systems introduce additional common mode failures associated with maintaining a master clock for the system, but truly asynchronous systems could, for a variety of reasons, get so far out of cinch that incorrect results could be produced.

Using a Markov reliability model for the system, its failure rate was predicted at less than 10^{-9} per hour. Assuming a double fault, with reconfiguration, the failure probability was calculated to be less than 10^{-10} per hour.

An attempt was made to commercialize the SIFT design (Wensley, 1987, and Wensley and Harclerone, 1982), and a series of research projects sponsored by EPRI looked at this architecture for possible applications in nuclear stations. It is a commentary on the demand for such products at the time that the commercialization was unsuccessful, though the system itself had many desirable features which should be considered seriously in future fault tolerant hardware designs.

Westinghouse Digital Feedwater Control

The Westinghouse Digital Feedwater Control System has been installed at several Westinghouse PWR plants. The following discussion relates specifically to the installation at the Catawba Nuclear Station. A detailed discussion of the design of this system is contained in Eastman, et. al. (1987).

Early operation at Catawba 2 indicated that the control of feedwater (steam generator level) was particular difficult for both the operator in manual control and for the analog feedwater control system. These difficulties are related to the design of the steam generator in this unit. It

became apparent that the algorithms for control which had been adequate traditionally in the industry would result in unsatisfactory levels of reliability.

In order to increase the effectiveness of control, the feedwater at Catawba 2, which consisted of Westinghouse 7300 based analog control, was replaced with a digital system using Westinghouse WDPF digital hardware. The result has been greatly improved feedwater control by the automatic system, and enhanced reliability of the station when compared to the analog control system. The Westinghouse system is discussed in detail in other documents -- this paper will deal specifically with the hardware design of the system and the reliability experience.

The Westinghouse WDPF system is a dual redundant asynchronous system. The architecture of this system is shown in detail in Figure 11. Each processor executes software which performs self-checking functions, including watchdog timing. Should the self checking features determine that the master processor is producing potentially faulty results, control is transferred to the backup processor. The operator is notified that such a transfer has occurred. Due to the time constants associated with the feedwater control, the time delay for transfer from the primary to the backup processor does not affect system dynamic response. There have been several successful instances at Catawba 2 where a master processor failure has caused a transfer to the backup processor. There has been a single hardware related system failure, caused by incorrect spare parts being used to effect a processor replacement. Overall experience has shown a significant improvement in feedwater system reliability through the use of the digital feedwater control system, and plant personnel are pleased with the system.

Foxboro/B&W Owners Group Advanced Control System

Control of the B&W PWR plants has from the beginning been more complex than that of Westinghouse or CE PWR plants. The Once through steam generator in these designs dictates that control of the reactor and the steam

plant must be much more tightly integrated than is necessary in systems using recirculating steam generators. The B&W plants came equipped with a system called the Integrated Control System (ICS) which is very similar to the coordinated control system used on supercritical fossil stations. The ICS was implemented in various vintages of Bailey analog control hardware.

Recognizing that the Bailey ICS systems were becoming obsolete, and that design problems in these systems had led to several incidents of improper control system behavior in these stations, the B&W Owners Group initiated a research and development program to deliver a digital control system which could replace the ICS and eliminate the potential for inappropriate interactions between the controls and the power station systems. This extensive program, carried out over a number of years and involving many technical personnel at the affected utilities, has resulted in the development of the Advanced Control System (ACS), a distributed digital system which replaces the ICS.

The reliability of the ACS is assured by the extensive of TMR architecture. All critical system instrumentation and the associated processors are triplicated, as are the voters which select the control demands to send to the plant. The system is implemented using hardware from the Foxboro Intelligent Automation (I/A) series, which itself provides a number of fault tolerant features, including the ability to remove from service and replace almost all system hardware components while the system is operating without an effect on system output.

One of the features of the development effort on this project was a careful analysis of the hardware related failures which had caused problems in the ICS, then incorporating specific design features in the ACS which would ensure that these problems would not recur. Another innovation was the construction of an integrated test facility in which the actual hardware and software to be installed in a station was tested using a high fidelity simulation. This integrated test facility is discussed in more detail in a later section.

HARDWARE BASED COMMON MODE FAILURES

A common mode failure disables the fault tolerant features of a system. Therefore it is necessary to analyze hardware designs for possible common modes failure points, then take appropriate action to eliminate them. The three major hardware sources of common mode failures are the sensors and final control devices, the input-output bus, the power supplies, and the data highway.

Sensors and final control devices may be supplied with the same level of redundancy as processors, but this is done only for the most critical channels. It is more common to have only a single sensor associated with a process measurement and a single final control device, such as a valve. If all redundant processors receive signals from a common sensor or supply output to a common device, then a fault in the input-output system could represent a common mode failure. This is a common mode failure in analog control architectures as well, but digital control systems provide more capability to avoid adverse effects of sensor failures. It is common for all input channels to be provided with some form of signal validation capability and a set of active limits on the value passed to the processor. For example, if a sensor is expected to provide a signal that ranges from 4 to 20 ma, the input channel is designed to recognize an input outside of this range and to reject the sensor signal as defective. The channel may then be frozen at the last value considered to be valid, and the operator notified of a sensor failure. More sophisticated signal validation features, such as rate of change checking or analytical redundancy, are also possible with digital systems.

On the output side, the digital to analog converter that supplies signals to actuators and final control devices may also be supplied as a redundant design. A self checking feature, which consists of a sampler and comparison circuit, may order the swapper from one D/A to another if the master circuit appears to be producing an inconsistent output.

The input-output bus is responsible for communication between the input and output devices (and their signal conditioning and sampling circuitry) and the processors. Because data from all of the sensors connected to a particular processor channel is required by all of the redundant processors in a subsystem, it is placed on a hardware bus that is accessible to all of the processors. A significant hardware problem on the I/O bus would therefore disable all of the redundant processors.

Power supplies are another possible common mode failure point, though the problem here is almost identical to the power supply problem involved in analog control system architectures. One additional complicating factor in digital control systems is the distribution of processors throughout the station, as opposed to the centralized nature of analog systems. It is necessary to ensure that a highly reliable power source is available at all locations of processor cabinets. It is standard practice to provide this power through rectifier-inverter combinations, with battery backup. Within individual processor cabinets, redundant power supplies are installed, and it is possible to deenergize, remove, and replace a power supply without interruption of service in that cabinet.

One of the most critical points in the system for common mode failures is the data highway. The data highway is a computer network, and it is shared among all of the processors in the system, those which perform control calculations and those which provide interface to the operator. Thus if a data highway becomes inoperable, the system loses all communication capability and all control and monitoring functionality is lost. Recognizing the critical nature of an operable data highway, manufacturers have provided features to ensure its integrity.

The data highway is almost always redundant. Processor output is isolated from the highway by highway controllers that provide protection for the processor in the event that one of the redundant data highways becomes inoperable. The system is designed to switch transparently from one highway to the other. Highways are also protected against external problems. The redundant highways may be routed so that they

are separated, except when they come together at a cabinet. If external magnetic or electric fields may be a problem, it is possible to use fiber optic cables for the highway that are unaffected by external fields. It should be possible to break or short one channel of the redundant highway without affecting the system operation.

One final possible source of common mode failures needs to be discussed. If there is an error in system hardware design, then that error will appear in all modules. Such a structural weakness has the potential to create a common mode failure, since all modules would be affected the same way under identical conditions. The best defense against a faulty design is an extensive operating history of a particular design in the field, under a variety of conditions and controlling a variety of processes. This is why the most logical course of action in adopting digital control and monitoring systems in nuclear stations is to use the widely applied offerings from commercial vendors. These systems have thousands of processor-years of application experience, active users groups that work to share information and improve the product, and a large number of well trained and experienced application engineers and technicians available. The fault tolerant features of these systems have been shown to be successful for a number of challenges. To undertake or require a custom development for nuclear applications is an invitation to an unsuccessful program that will significantly hinder the ability of utilities to take advantage of the many improvements possible through the application of digital systems in nuclear stations.

ESTABLISHING THE LEVEL OF SYSTEM RELIABILITY

Hardware Analysis

Quantitative methods are available that can be used to assess the reliability of safety critical digital systems and perform comparative analyses of various fault tolerant options. These methods are based on the development of various models of the system. In order to

understand these methods, some basic ideas and definitions are required.

The probability of system failure $F(t)$ is the probability that the system will fail sometime between time 0 and time t , given that it was operable at time t . The reliability $R(t)$ is $1 - F(t)$. The failure density function $f(t)$ is the derivative of $F(t)$ with respect to time, and the hazard function is

$$z(t) = \frac{f(t)}{1 - F(t)}$$

It is the hazard function that is most often used to describe the performance of a system, since it has units of failures/unit time. If we plot the hazard function versus time for typical electronic components, it produces the famous "bathtub curve." In early stages of operation, an electronic system suffers from so-called "infant mortality" as weak parts fail early under normal operational stresses. This early period is followed by a long period of constant failure rate, called the useful life period. The end of the useful life period is marked by an increase in failures due to components reaching the end of life.

Specific steps are taken during manufacture and factory test to eliminate the early failures, so that we may assume that equipment placed into plant service is operating in its useful life period. If we take the constant failure rate during that period to be equal to λ , then we get the following equations for our three important constants

$$f(t) = \lambda e^{-\lambda t}$$

$$F(t) = 1 - e^{-\lambda t}$$

$$R(t) = e^{-\lambda t}$$

This constant failure rate, λ , is what is documented in MIL-HDBK-217D for many

electronic parts. Such an assumption may not necessarily hold for certain types of VLSI circuits, and alternative techniques of assessing individual component reliability may be necessary.

Mean time to failure (MTTF) is the expected value of the reliability, $R(t)$. Mean time to repair (MTTR) is the expected value of the repair time for the system. Mean time between failures is the sum of these two expected values:

$$MTBF = MTTF + MTTR$$

Finally, system availability $A(t)$ is the probability that a system is operational at time t . If a system is unrepairable, then $A(t) = R(t)$. If a system can be repaired and kept in service, the limit of $A(t)$ as t gets very large is a constant, A , which has the value

$$A = \frac{MTTF}{MTTF + MTTR}$$

One of the most common methods for modeling the reliability of a system and performing a quantitative assessment is through the use of Markov models. A Markov model consists of a set of mutually exclusive states that represent the possible conditions of the system with respect to the operability of its components and subsystems, along with a set of transitional probabilities that represent the probability of state transitions. Markov models are particularly useful for evaluating the relative reliability of various redundancy schemes. When Markov modeling of reliability is combined with the other aspects of the objectives of system reliability, such as the necessity for fault masking vs. a penalty for recovery time for unmasked faults, it is possible to arrive at a reasonable assessment of whether the reliability of a particular system is adequate to meet the needs of the critical application. For example, it is possible to show whether a particular digital computer based replacement of an analog controllable leads to greater or less reliability.

Further discussion of the use of reliability modeling of fault tolerant digital systems may be found in Mathur (1971) and Ng and Avizienis (1980).

Integrated Testing of System Reliability

The compactness and modularity of digital control and monitoring systems make possible the construction of integrated system testing facilities that were not possible with analog systems. The basic architecture of such testing facilities involves control and monitoring hardware identical to that installed in the plant interconnected with a simulation model of the plant process. Such a facility has multiple uses in ensuring the operability and reliability of the plant:

1. The facility may be used to test and verify all of the software to be installed in the system. This includes the application software and the graphics software that provides the operator interface. There is a significant cost advantage in reducing the length of a startup by ensuring that software is essentially correct.
2. All of the fault tolerant features of the system may be tested by imposing various hardware faults on the system and assessing their effects. This includes imposing failures at all system inputs and outputs, faults on the processor boards themselves, and faults in the power supplies. The capability of the system to be repaired on line without adverse effects on system operability can also be tested.

The importance of this type is testing cannot be overemphasized. A Failure Modes and Effects Analysis (FEMA) can never be as good as an actual hardware test. Such testing in the past has been prohibitively expensive or only partially complete. With the declining cost of hardware and the modularization and standardization provided by digital systems, it is now a feasible measure to provide assurance that the system will perform as required by the specifications.

3. The facility may be used as a training simulator for plant operators. By duplicating the human-machine interface to be used in the control room, as well as the actual inputs and

outputs of the control system, a simulator that is much more complete and realistic than any currently in use in the industry can be provided at a much lower cost.

4. An integrated test facility is a research and development laboratory for all types of advanced technologies. These new technologies include computer based operator aids using AI, advanced control strategies and methods, new ways of integrating facility management and plant operation, and the development of predictive maintenance programs.

Two such facilities have been recently completed and will be described below.

The Duke/EPRI Mobile Simulation Facility was developed to integrate a Westinghouse WDPF control and monitoring system with a simulation model that could be used for training station operators at several plants in the Duke Power system. This facility consists of two mobile units containing a complete WDPF control system, including operator consoles and processors, and a set of networked 486 based computers that run a simulation model of the plant for which training is desired. The interface between the simulation model and the control system takes place at the I/O bus level through the use of common memory. The Westinghouse hardware consists of processor and memory units, as well as power supplies and cabinets, identical to those installed in power stations. The software that is run on the WDPF equipment is identical to the software run in the power station, and the operator interface screens are also the same as in the plant. The hardware layout of the mobile simulator facility is shown in Figure 12.

This facility has been used for the development of the control application software for a supercritical fossil station with about 3500 I/O points, and is currently being used to train the operators for this facility. Feedback from the operators and from the control designers has been very positive. Because this is a fossil station application, there is not a systematic program for the assessment of failure modes nor testing of fault tolerant features of the system, but if this were a requirement, the facility would perform these tasks very well.

In this facility, the I/O processing channels are not present, and the interface takes place at the I/O bus level, this is, at the output of the I/O circuitry. The same basic architecture may be used to construct a system that includes the complete I/O channels as well as the processors.

The B&W Owners Group test facility at the Davis Beese nuclear station training center is an additional application of an integrated test facility for digital control and monitoring systems. This facility uses a Foxboro I/A digital control system interfaced to a simulation model developed as a training model for the Rancho Seco nuclear station. Special hardware interface equipment has been constructed to convert the outputs of the simulation model to the physical signals that would be expected from the plant (e.g., 4-20 ma) and the outputs from the control system are converted to a digital demand signal for input to the simulation model. Thus the entire digital control system, including all of its I/O equipment, the processor channels, the operator interface, and the power connections and other terminals are duplicated in the test facility. The software used in the test facility is identical to that to be installed in the B&W Advanced Control System when it is implemented at a station.

For this facility, a comprehensive testing program to assess the fault tolerant features of the control system is being performed. A complete hardware Failure Modes and Effects test is also being performed. Combined with a software validation effort, a high degree of assurance will be obtained that the level of reliability of the system is adequate to meet its specification requirements.

Both of the described facilities are fully operational now. One of their great advantages is that the basic architecture of digital control systems enables facilities like these to be used for a variety of power stations. Development of a simulation model of the particular facility is necessary, but a modular simulation development system is being used to simplify this step. The control system application software is identical to that used (or to be used) in the power plant. The basic architecture may be used with control systems manufactured by Westinghouse, Bailey, Foxboro, and Forney.

REFERENCES

- Eastman, M. C., K. A. Gaydos, K. F. Graham, M. H. Lipner, N. P. Mueller, C. N. Nasrallah, A. K. Negus, R. E. Paris, W. F. Schaefer, J. B. Wachio, and D. D. Woods, *Advanced PWR Steam Generator - Feedwater Control System*, EPRI NP-4919-LD, April, 1987
- Hopkins, A. L., T. B. Smith, and J. H. Lala, "FTMP -- A highly Reliable Fault Tolerant Multiprocessor for Aircraft," *Proceedings of the IEEE*, Vol. 66, No. 10, October, 1978, pp. 1221-1239
- Kisner, R. A. and R. E. Battle, *Fault Tolerant Architecture: Evaluation Methodology*, EPRI TR-100803, August, 1992
- Lala, P. K., *Fault Tolerant and Fault Testable Hardware Design*, Prentice Hall, Englewood Cliffs, NJ, 1985
- Mathur, F. P., "On Reliability Modeling and Analysis of Ultrareliable Fault Tolerant Digital Systems," *IEEE Transactions on Computers*, Volume C-20, No. 11, November, 1971, pp. 1376-1382
- Ng, Y. W. and Avizienis, A. A., "A Unified Reliability Model for Fault Tolerant Computers," *IEEE Transactions on Computers*, Volume C-29, No. 11, November, 1980, pp. 1002-1011
- Rennels, D. A., "Fault Tolerant Computing -- Concepts and Examples," *IEEE Transactions on Computers*, Vol. C-33, No. 12, December, 1984, pp. 1116-1129
- Siewiorek, D. P., "Architecture of Fault Tolerant Computers: A Historical Perspective," *Proceedings of the IEEE*, Vol. 79, No. 12, December, 1991, pp. 1710-1734
- Wensley, J. H., L. Lamport, J. Goldberg, M. W. Green, K. N. Leavitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Weinstock, "SIFT: Design and Analysis of a Fault Tolerant Computer for Aircraft Control," *Proceedings of the IEEE*, Vol. 66, Number 10, October, 1978, pp. 1240-1255
- Wensley, J. H., "Fault Tolerant Computers Ensure Reliable Industrial Controls," *Electronic Design*, Vol. 29, No. 13, reprinted in V. P. Nelson and B. D. Carroll, *Fault-Tolerant Computing*, The IEEE Computer Society Press, 1987
- Wensley, J. H. and C. S. Harclerone, "Programmable Control of a Chemical Reactor Using a Fault Tolerant Computer," *IEEE Transactions on Industrial Electronics*, Vol. IE-29, No. 4, pp. 258-264
- Williams, T. J., *The Use of Digital Computers in Process Control*, Instrument Society of America, Research Triangle Park, NC, 1984.

Table 1. Comparison of Dual and Triple Redundant Architectures

	<i>Triple</i>	<i>Dual</i>
<i>Fundamental Operating Principle</i>	<i>Compare Duplicate Answers</i>	<i>Check Hardware</i>
<i>Decision Methodology</i>	<i>2 out of 3 majority voting</i>	<i>Self Checking of hardware</i>
<i>Intrinsic Complexities</i>	<i>Increased hardware</i>	<i>- Additional Software - Diagnostic Software</i>
<i>Fault Response</i>	<i>Error masking</i>	<i>Error detection</i>
<i>Recovery Technique</i>	<i>Static</i>	<i>Dynamic</i>
<i>- Fail over Mechanism</i>	<i>None</i>	<i>Processor Switching</i>
<i>- Fault Coverage</i>	<i>100%</i>	<i>99% (vendor claim)</i>
<i>- Basis for coverage projections</i>	<i>Inherent operating principle</i>	<i>Vendor Experience base and claims</i>
<i>Reliance on Coverage data</i>	<i>0%</i>	<i>100%</i>
<i>Unique common mode failure potential</i>	<i>- Software - Voters (if not redundant)</i>	<i>- diagnostics coverage - transfer mechanisms - EMI /RFI - operating systems</i>
<i>Industry based availability experience</i>	<i>100%</i>	<i>99%</i>
<i>V&V Required</i>	<i>Moderate</i>	<i>High due to fault tolerant software requirements</i>
<i>Hardware dependencies</i>	<i>None</i>	<i>FT pairs must be identical</i>
<i>Software dependencies</i>	<i>None</i>	<i>FT software must match</i>
<i>Overall Cost (as % Installed system cost)</i>	<i>13-18%</i>	<i>10-15%</i>

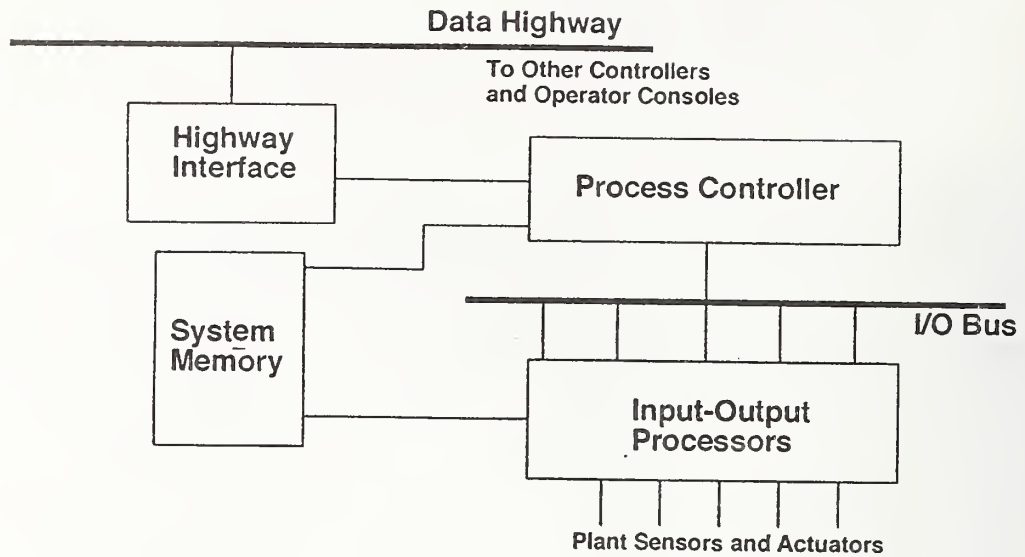


Figure 1. Typical Digital Control System Hardware Architecture (Processor Level)

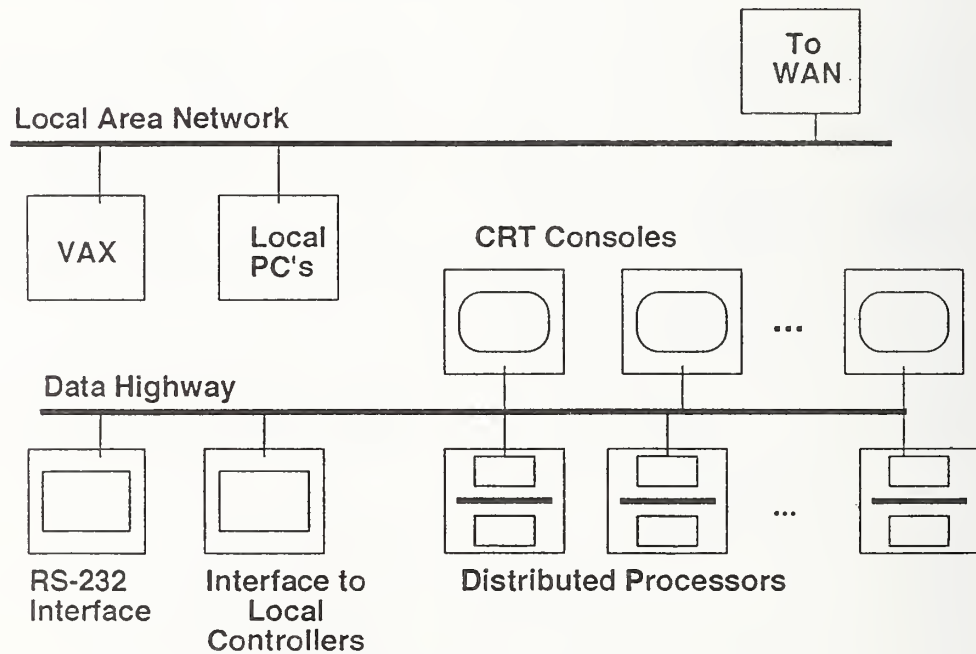


Figure 2. Typical Digital Control System Hardware Architecture (System Level)

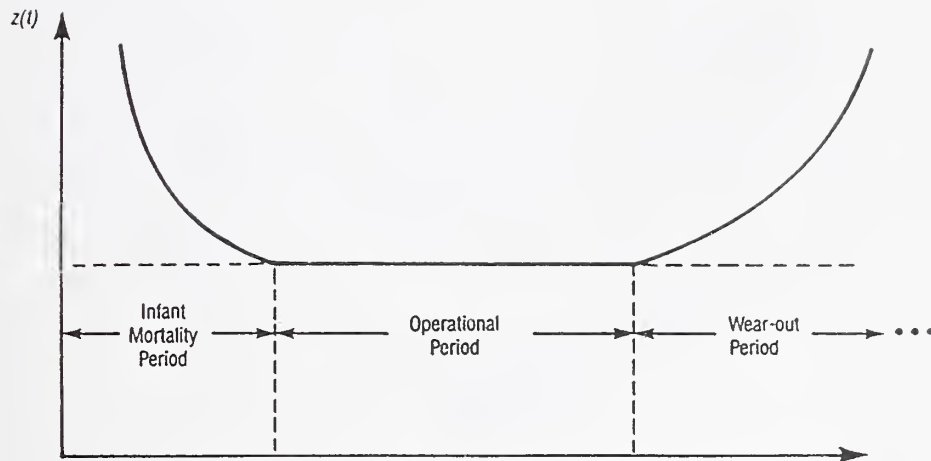


Figure 3. Traditional Hardware Failure Curve

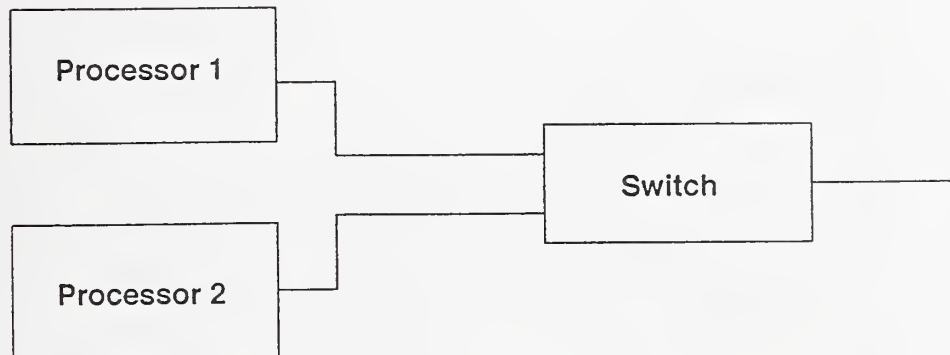


Figure 4. Basic Dual Redundant Architecture (Self-Checking Modules)

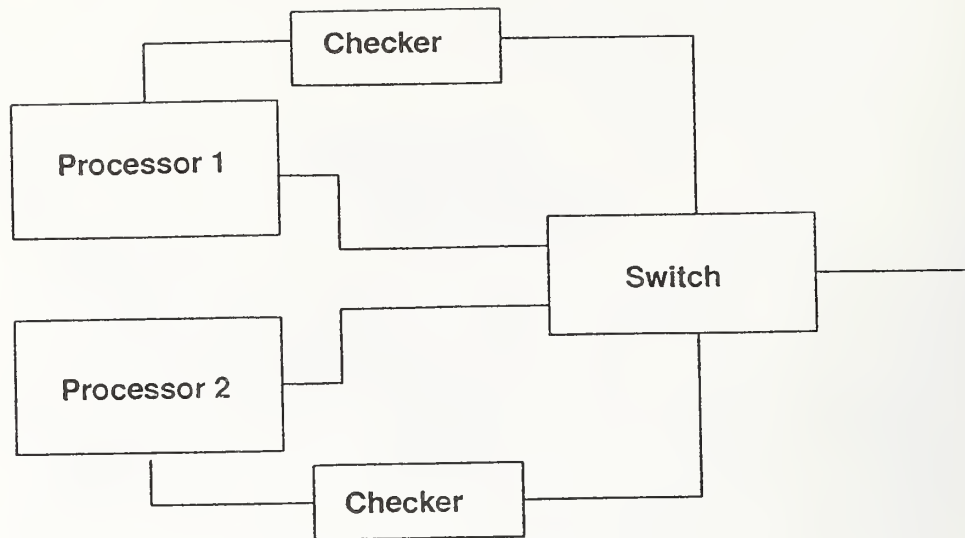


Figure 5. Dual Redundant System (External Checking)

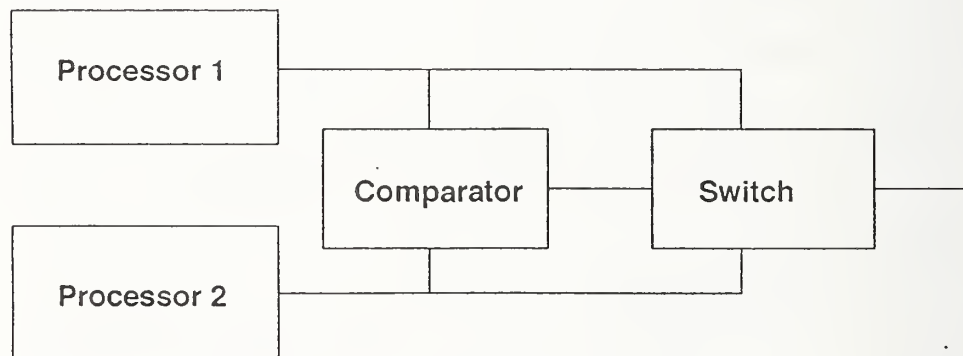


Figure 6. Dual Redundant Synchronous System

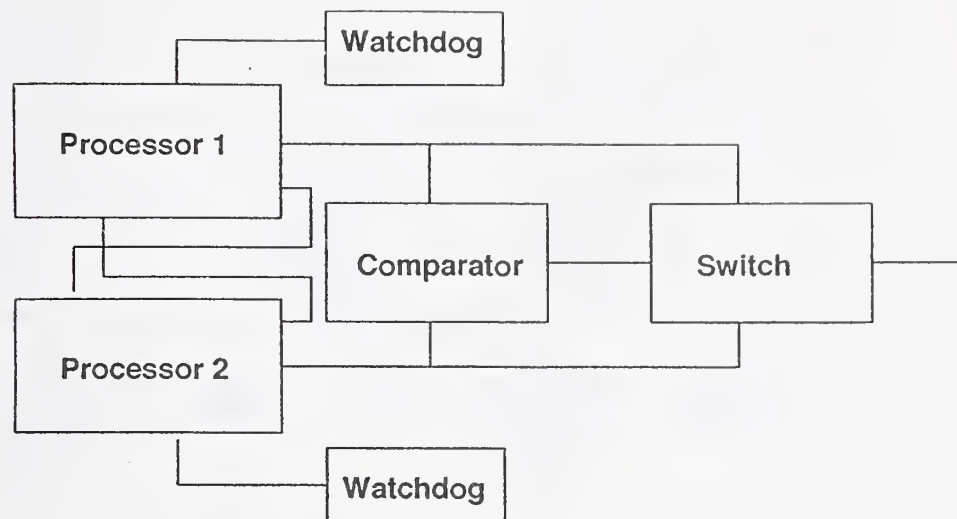


Figure 7. Dual Redundant Synchronous with Cross Checking and Watchdog Timers

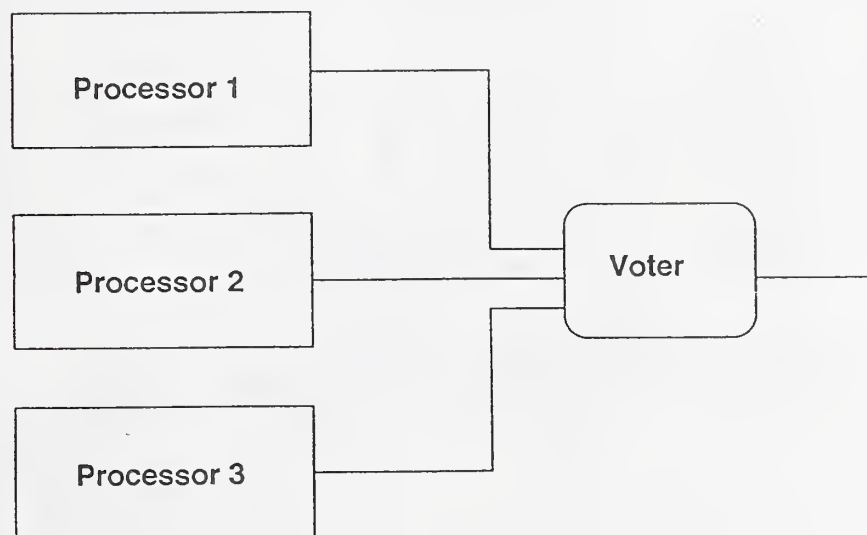


Figure 8. Triple MODular Redundant (TMR) Processors

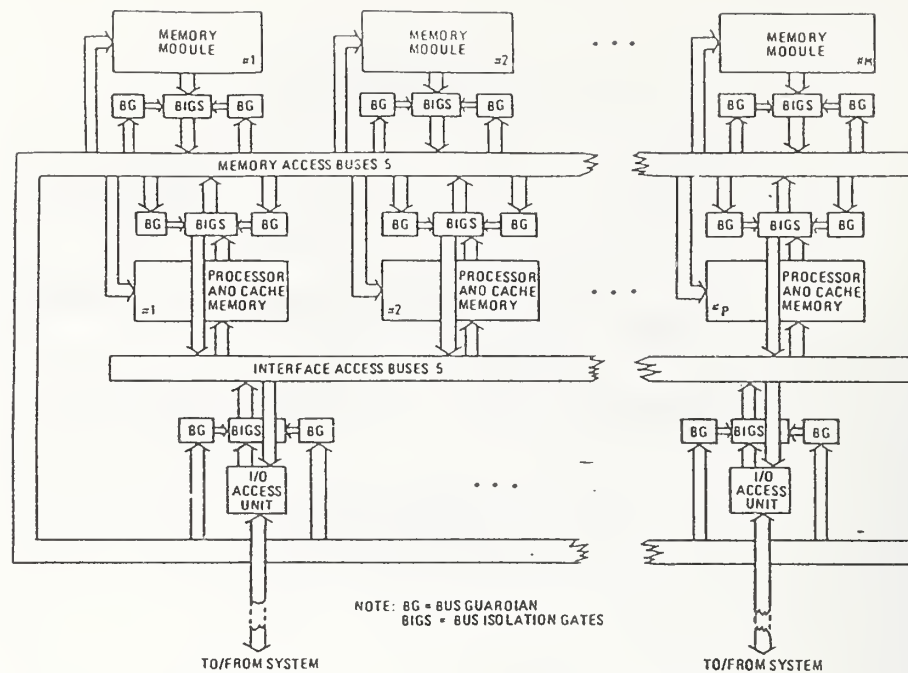


Figure 9. Architecture of Fault Tolerant Multiprocessor (FTMP)

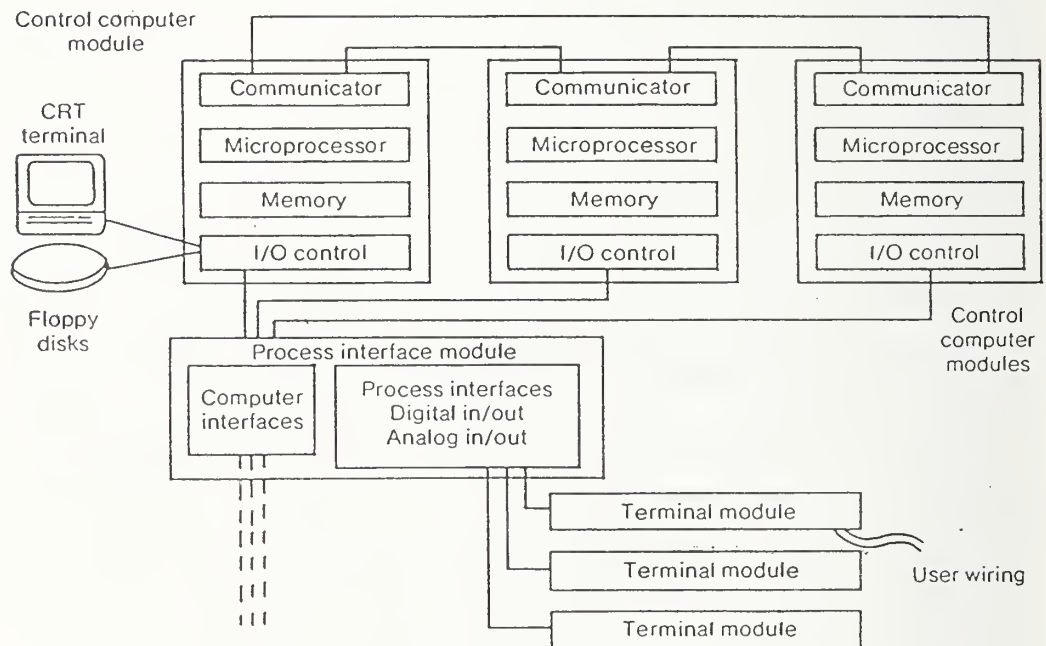


Figure 10. Architecture of Software Implemented Fault Tolerant (SIFT) System

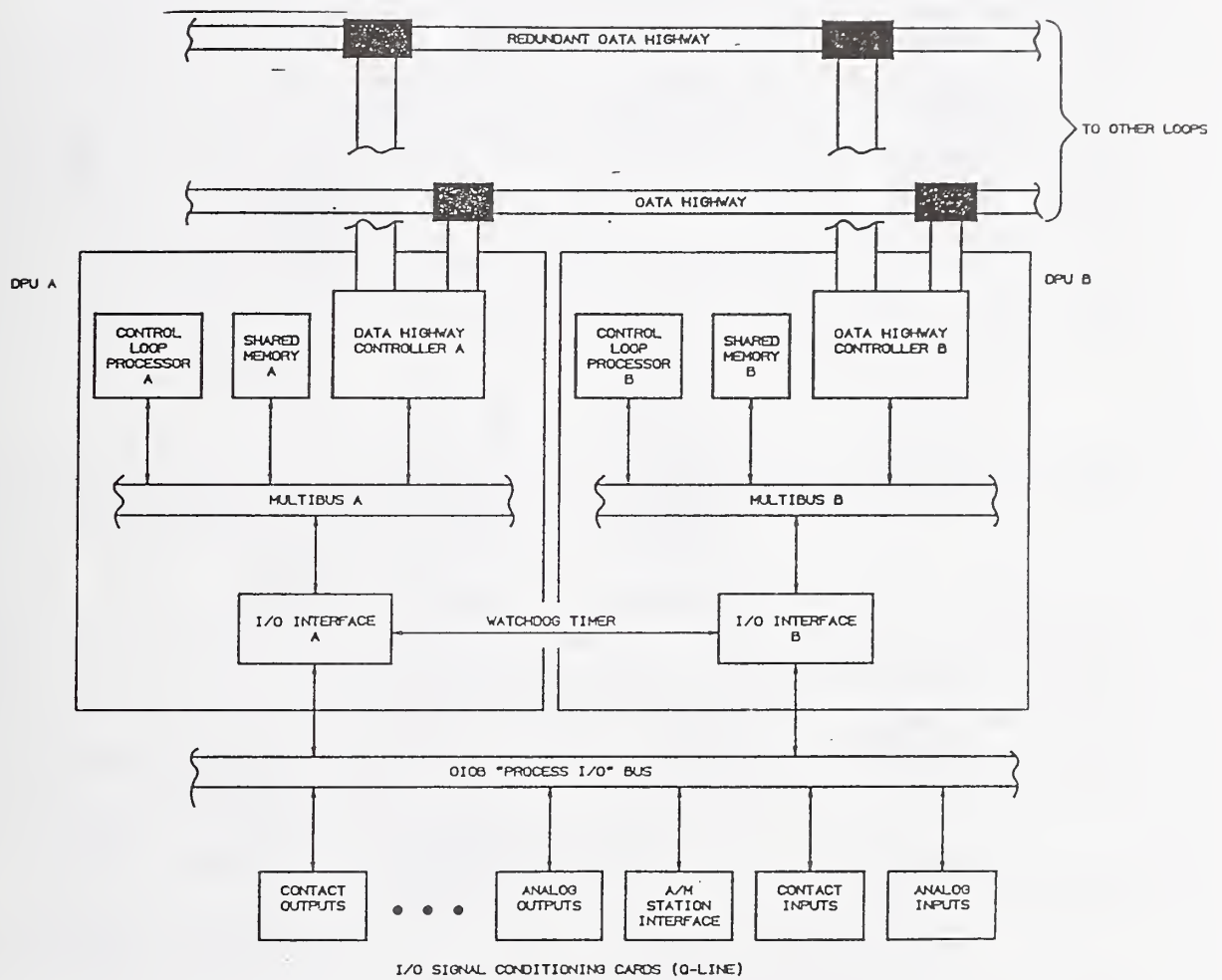
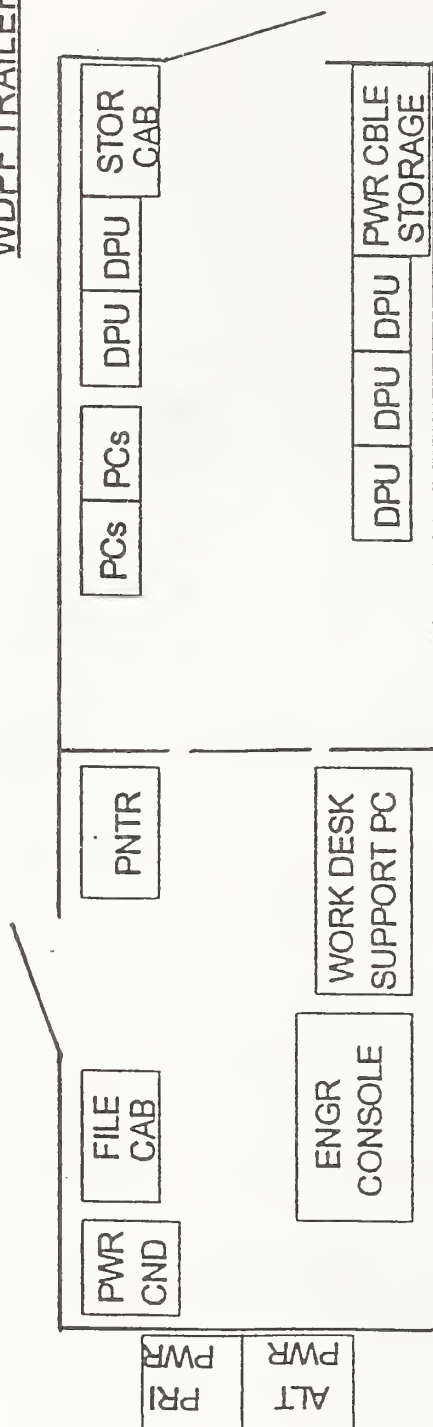


Figure 11. Westinghouse Digital Feedwater Control (WDPF)

WDPF TRAILER



NOTES:

PRI PWR = 600 - 240/120, 25KVA XMFR

ALT PWR = 240/120 @ 80 A

PWR CND - POWER CONDITIONER 15KVA

TRAINING TRAILER

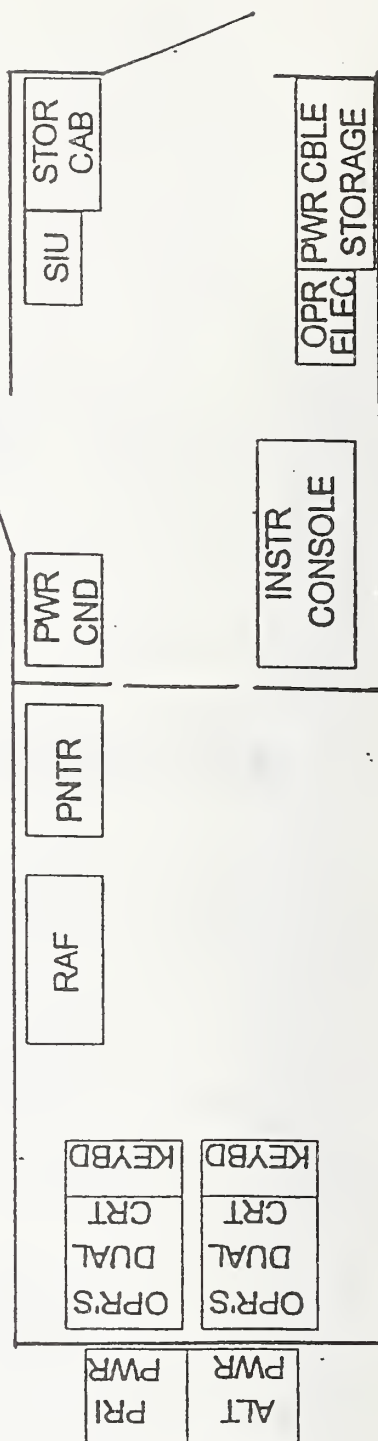


Figure 12. EPRI/Duke Mobile Simulator Trailer Layout

4.1.1 Questions: Mr. A.L. Sudduth

QUESTION: GREG CHISHOLM (Argonne National Lab): Is there any ongoing work directed towards developing a fault tolerant platform that will be qualified for use in nuclear reactor protection systems, that is a network architecture in this process, or independent, and that supports a general scheme of failure detection, isolation, and reconfiguration?

MR. SUDDUTH: I don't know of any work like that going on today. If you look back at the work that NASA sponsored in the late 1970s at SRI and NASA-Langley, you find that people proposed these things. In the case of SIFT, somebody actually went out and tried to build one and sell it. Basically what we do is we go to vendors. We go to the Baileys and the Foxboros and the Westinghouse guys and we say, "What are you building and how well does it work?" I think if the nuclear industry undertakes a process whereby they decide to custom develop something for themselves, 10 years from now you'll still be sitting in this room talking about how are we going to make it work. So, you've got to do the same thing. You're going to have to go to the people who are selling these things in the commercial arena, look at what they offer, perhaps you could influence their design by saying, "Look, you know, if you just tweaked it here and tweaked it there you could make it better and you could sell one to us." But you're not really going to be able to go out and develop something unique to this industry, nor do I really think that's necessary, because what we have now from our vendors is perfectly adequate for the uses we're going to put it to.

QUESTION: BRIAN TOLLEFSON (BG&E Calvert Cliffs): When installing the Westinghouse system, how long did it take, in general, to fine tune the system, and during that time period how many trips occurred and what were the nature of the problems?

MR. SUDDUTH: Boy! When we first started this work back in the mid-1980s we really didn't know what we were doing and we had a really hard time. It was a difficult effort. We were helped by the fact that the plants that we initially started installing this equipment on were in extended shutdowns, that is they were taken out of service for two or three year periods of time in which complete renovation of the whole plant was going on. The digital system would go in 6 months to a year before the station was returned to service and then the operators and everybody would have a chance to play with it for a while.

That worked pretty well and generally we were able to get the systems back up and running. But, of course, you never actually finish one, and we're still tending some systems that have been in service for 5-6 years now.

Our new approach, which is the simulator-based approach, we hope will alleviate that. For the plant I'm currently working on, there is a 12-week outage from the time the unit comes off the line until it comes back with a complete digital system. The system has to work basically the first time that it's plugged in. The way we're going to guarantee that it does is that we test it in the simulator, and the simulator test probably has been going on almost a year. There is a very good possibility--we've got our fingers crossed anyway--that the number of errors in the system will be very low when it actually goes into the working plant, but it's a good question.

The real answer is these systems have a real steep learning curve. It's very difficult to learn how to implement them properly, to get the software right, and to get the hardware installed correctly, so that you don't have problems. We have, finally, after almost 8 years, learned how to do the Westinghouse system right, and so we have a lot of confidence that we'll be able to install it in the future with extremely short outages.

QUESTION: MIKE ENGINEER (Fluor Daniel/M&O): What kind of data highway failures have you experienced? During transience have you experienced saturation of data highway communication? Have there been any problems with data collisions or mysterious false-trips?

MR. SUDDUTH: I assume due to data highway problems?

We have had only one major catastrophic highway problem, and that happened when a worker struck an arc on the highway conduit with a welding machine. That caused such a disruption in the system that the whole world went down. Somebody asked me, "Well, how can you run a plant if you don't have any hard panels? What does the operator do when the computer goes away?" Well, it's a relatively simple question. In the fossil station we look around and decide what position everything in the plant ought to be in, in order to make the plant safe, and then if the computer goes away everything goes to those positions. Now, if there's something that we can't set that way we do have to put it on a hard panel. There are a certain limited number of controls--as I mentioned, maybe a 2-foot square hard panel--which we give the operators, but basically that's not been a problem.

I guess I should mention, we did have one other highway problem that occurred at our Catawba Nuclear Plant. There they have the Westinghouse Eagle Digital Control System and the problem there was a configuration management problem. When they did a repair to the system and plugged in a new processor board that board did not have the right hardware rev level. Before the board was even energized it corrupted both highways and brought the system into a big transient. The operator was then able to recover a B-water level, or a steam generator level, and the plant was tripped. So, you've got to be careful about a lot of things.

We've not had any problems with data collisions because the Westinghouse System uses the Token Bus architecture so there are no questions. However, you can saturate a highway--not from a transient state--because the highway basically passes the same amount of information regardless whether you've got a transient or not. It's really all the same. Every variable gets passed on the highway every time, all the time, regardless of what's going on in the plant. But eventually you can get so much stuff on the highway that you can't make the highway work any more, and then you have to do this tremendously boring task called splitting the highway which takes about 3 months and is real difficult to do.

No mysterious false trips in the system that we know of. We know exactly what's caused all the problems in the digital system.

4.2 Software Aspects for Safety-Critical Systems: Dr. John C. Cherniavsky

Software Aspects for Safety-Critical Systems

Dr. John Cherniavsky
National Science Foundation

[as edited from transcript]

Dr. Cherniavsky represented the National Science Foundation (NSF) and the other government agencies outside of the NRC and NIST which are interested in the use of high integrity systems.

NSF is an independent agency of the Federal Government established about 40 years ago to promote and advance scientific progress. It only sponsors research; it does not conduct any research. Last year, NSF received 57,000 proposals and made 20,000 awards for a total of about \$2.3 billion. Since NSF only has 1,200 employees, it makes extensive use of advisory groups and used 60,000 outside reviewers to make 200,000 reviews last year. NSF makes awards to academic, industrial, non-profit and Government recipients. NSF also works with other agencies, particularly ARPA, the Department of Energy and NASA.

NSF contains seven directorates. The Computer and Information Science and Engineering (CISE) Directorate funds the work of interest to this group. CISE has a budget of \$230 million. CISE contains six divisions. The Computer Computation Research (CCR) division supports software engineering among other areas. CCR has a budget of \$46 million. The software engineering program has a budget of \$4.36 million. This is the home of most of the computer science research, software research, and research in software V&V.

This funding primarily goes to support university-based research though a small amount goes to small business via the Small Business Innovative Research (SBIR) Grants. They typical size of a grant is \$50,000-\$60,000 per year.

There have been some other opportunities in the software engineering area. Three or four years ago, there was a Formal Methods Initiative which attracted at least 60 proposals of which four were funded. NSF has also supported the Center for Research in Real-time Intelligent Complex Computing Systems at the University of Massachusetts at Amherst and the ARCADIA project.

NSF has also provided some limited software safety support. There is not a large university-based research community in this area, and NSF would like to encourage the development of such a community. NSF has supported

- individual researchers like Nancy Leveson and John Knight
- the early formal methods, and temporal logics work
- mutation analysis tool development at Purdue
- research in the underlying mathematics necessary in assuring the correctness of systems

This last may be the most important in the long term.

NSF has had discussions with other organizations including NIST, Mitre, and Argonne to identify targets of opportunity to strengthen research in high integrity systems. One proposed project was a Center for Software Safety Studies funded at a level of \$2-\$4 million a year over the long term. Another proposal was to fund postdoctoral support for software safety researchers in real environments where they can get their hands on real system problem. NSF will continue pursuing joint agency initiatives in this area with NRC, NASA, the Department of Energy and NIST.

One current initiative is the High Performance Computing Communications Initiative. This has identified grand challenge problems (for example, weather prediction, vision research, computational fluid dynamics, and urban flow problems). The CCR division would like to look at grand challenges in the complex systems area. It is just as important to build complex systems as it is to be able to build systems that will solve grand challenge problems. Some examples of complex systems are aircraft control systems, the space station, multimedia interactive telecommunications networks, banking and stock systems, and automated factories. These systems share many characteristics. They are large, real-time, and interactive, and the failure of these systems has serious financial and safety consequences. NSF is currently investigating an initiative that would ask for proposals for the underlying science that would help us build these systems correctly.

Dr. Cherniavsky can be reached on EGOS. His email is jcc@nsf.gov, and his phone is (202) 357-7349. Please call or write if you have any questions or suggestions for NSF in this area.

4.3 Human Aspects for Safety-Critical Systems: Dr. Lewis F. Hanes

HUMAN ASPECTS FOR SAFETY-CRITICAL SYSTEMS

Lewis F. Hanes
Consultant and
Adjunct Professor
Industrial and Systems Engineering Department
The Ohio State University
2023 Wickford Road
Columbus, Ohio 43221
(614)-487-8655

ABSTRACT

This report addresses crew, i.e., human, aspects of the widespread introduction of digital technology into nuclear power plant control rooms. Such changes as increased levels of automation, intelligent electronic displays, and compact work stations influence crew situation awareness, workload, etc. The overall impact is expected to be enhanced safety, although such challenges as automation complacency and access to information in multi-dimensional space must be handled. In considering design certification for safety, it is desirable that guidelines be available. Since digital technology is developing more rapidly than accepted guidelines, regulatory emphasis is placed on the design process and verification and validation. The lack of adequate guidance creates a need to assess knowledge from other applications already experienced with digital technology (e.g., foreign nuclear and fossil plant upgrades, commercial aviation), and to perform application research on such topics as automation and information management.

INTRODUCTION

Digital technology is having a major impact on the design of upgrades for existing nuclear power plant (NPP) control rooms, and advanced control rooms for evolutionary and passive plants. Control room hardware, software, and crew roles and tasks are being affected. This document is concerned with the crew, i.e., human, aspects of these design changes.

Exhibit 1 provides an overview of the topics presented in this report. Changes caused by digital technology impacting on crew roles and tasks are described. These changes affect factors influencing the crew as it performs required tasks. Based on these changes, safety is enhanced, but also, challenges to safety must be addressed. In reviewing the design changes due to digital technology, the Nuclear Regulatory Commission (NRC) utilizes guidance documents. Regulatory positions are influenced by the availability and applicability of such documents. The nuclear power industry has limited experience with the effects of digital technology on human performance, and the guidance

documents are not complete. Therefore, it is important to transfer knowledge from related applications and to perform high priority applications research.

EXHIBIT 1. Document Overview.

- * Changes with Digital Technology
- * Factors Influencing Task Performance
- * Potential Safety Issues
- * Regulatory Positions
- * Research Needs

CHANGES ATTRIBUTABLE TO MODERN TECHNOLOGY

Important changes in crew roles and tasks, and human-system interfaces (HSI) are occurring in advanced control room designs. A review of upgrades, plants recently becoming operational, and advanced control rooms under design reveals some of the changes (references 1,2,3,4,5,6,7, and 8). These changes are presented in Exhibit 2.

EXHIBIT 2. Representative Changes with Digital Technology.

- * Increased Levels of Automation
- * Intelligent Electronic Displays Providing
 - Integrated & Graphical Information
 - Operator Aids & Advice
 - Computerized Procedures Integrated with Plant Data
 - Alarms that are Prioritized & Filtered
 - On-Line Access to Extensive Stored Data
- * "Soft" Controls on Displays
- * Large Dynamic Overview Displays
- * Compact Work Stations

AUTOMATION

Automation is used extensively in operating NPPs. The crew is not involved directly in many control tasks that can be handled more reliably by automatic control systems. New advanced control room designs incorporate even higher levels of automation. For

example, some advanced plants are being designed for automatic control of startup, power level change and shut down (8).

Three types of automation can be described: control, information and management (9). Control automation involves "closed-loop" operation of NPP components and subsystems. An initializing event causes control actions to occur without operator participation.

Information automation involves providing the crew with information about the state of the NPP, the process, systems, subsystems, etc. Information automation can involve at least two situations. System designers determine what information should be available for display depending on plant and process state. This information is formatted and presented to the crew automatically when certain plant and process conditions exist. For example, alarm information is presented on annunciator displays when an alarm set point is exceeded. In a second situation, a crew member requests that information be displayed. The process of collecting the requested information, formatting it, and presenting it may be performed automatically.

Management automation directs control automation in the performance of the requested change. The crew establishes goals which are accomplished by automated systems. An example is automatic power level change. An operator communicates that power should be increased from 78% to 85%. The management automation system commands automated control systems to take appropriate actions. The information automation system provides the crew with the information needed to monitor progress of the power change.

INTELLIGENT ELECTRONIC DISPLAYS

Electronic information and display capabilities are being incorporated in upgrades (1), new control room designs (2,3,4,5,6,7), and have been included in operating plants, such as Toshiba plants in Japan (8). Electronic information and display capabilities are made possible by modern computers, electronic display devices, electronic data bases, high speed data communications networks, intelligent software and logic, graphic display hardware and software, etc. When configured properly, these technological capabilities provide major changes in the form of data and information provided crew members.

Hard-wired indicators in conventional NPP control rooms can show only the sensed data to which they are connected. Since there are many sensors and parameters of importance to the crew during the various phases of plant operations, large control boards are filled with indicators. The reconfigurable (soft) electronic displays permit the sensor and parameter values to be presented

only when required by the crew. The number of indicators can be reduced dramatically. In fact, the Nuplex 80+ Advanced Control Complex is being designed to provide 70% fewer indicators and 60% fewer alarms for the crew members to handle as compared to earlier Combustion Engineering control complexes (5). This reduction in display devices provides the opportunity to design much more compact work stations.

The display configuration in many conventional control rooms requires crew members to move physically around the control center, locate and read many indicators sequentially; to remember these display values; to recall from human memory and/or reference documents additional data; and to integrate and analyze these data mentally to create information upon which to act. It is impressive how well trained crew members can perform this activity. With information automation, however, the needed data are collected, integrated and analyzed, and presented in a unified textual, graphical, or combined form with little or no crew involvement.

There have been numerous efforts to design and provide intelligent operator aids and advisors (e.g., 8,10,11). These devices are intended to provide crew members with diagnostic, problem solving, and response planning support.

Another electronic information and display feature that provides an opportunity for improved crew performance is electronically displayed procedures that are integrated with operating plant data.

There are situations in a conventional control room when many events are occurring simultaneously and in rapid succession. An example is during the onset of a scram. Visual and auditory alarms, flashing displays, and similar kinds of events pepper the crew with data. A well-designed electronic information and display unit draws attention to events that are important, but suppress data that interfere with crew task performance. Much work has been devoted to developing alarm system guidelines (12) and new alarm systems (1, 13).

The availability of data stored electronically in local data bases or available over communications networks may enhance the quality of crew performance. Improved crew performance can occur, of course, only when the crew member's task can benefit from access to the data, the data can be obtained and used in a timely manner, and the process of obtaining the data does not interfere with the other tasks being performed.

"SOFT" CONTROLS

Advanced control rooms are being designed with touch screens

(e.g., the French EdF N4 control room, reference 14). These panels are easy to use for selecting menu options, and for pointing at objects of interest. Touch controls are "soft" in the sense that a given control surface area can represent different controlled variables depending on state of the process and plant. In conventional plants controls are each dedicated to one function. The capability provided by "soft" controls to reduce the total number of controls in an advanced control room facilitates the use of compact work stations.

LARGE OVERVIEW DISPLAYS

Large display panels that can be seen throughout the control room are being included in new control room designs (e.g., 4,5, and 6). These displays typically provide an integrated plant overview and high level alarms. These displays should facilitate crew discussions of the overall state of the plant and process. In addition, the large displays should permit non-crew members to become familiar with and monitor plant status without interfering with the operating crew.

COMPACT WORK STATIONS

Electronic displays and "soft" controls are being incorporated into compact work stations (e.g., 3,4,5, and 6). These work stations typically permit a seated operator to access extensive information about the plant and process, and to control those elements of the plant that he or she is authorized to control. The work stations contain several electronic displays and input devices, such as the mouse, trackball and touch screen.

Compact work stations eliminate the travel time required with conventional control rooms for the crew members to move between displays and controls on the instrument boards. Relationships between displayed parameters and controls may be more obvious because of the proximity made possible by the work stations. Conversely, the shift supervisor and other crew members cannot observe from a distance the physical location of a worker at the instrument board and know the part of the plant/process being attended.

CREW ROLE AND FACTORS INFLUENCING TASK PERFORMANCE

The control room crew will continue to be an integral part of upgraded and new NPP systems (e.g., see 1,2,3,4,5,6,7 and 8). The plant will not be designed for fully autonomous control, in which the crew has no role in operation, monitoring is limited to fault detection, system goals are self-defined, and the crew normally has no reason to interfere (see 9). The role of the crew, in fact, will continue to involve monitoring, supervising, and backing-up automated systems. The upgraded and new designs, however, will require less manual control.

The crew role, as described above, requires the crew to perform a variety of tasks. The nature of these tasks will be impacted by the application of digital technology. For example, higher levels of control automation with associated integrated graphical display presentations may make the crew's task primarily one of monitoring and detecting deviations from plan. In contrast, the crew's task associated with the same plant state with conventional control and display may be more response planning and execution of control actions.

Performance of the crew is influenced by the nature of the tasks and the methods by which modern technology is incorporated into the human-system interfaces. Some of the factors influencing task performance that are affected by digital technology are listed in Exhibit 3.

EXHIBIT 3. Some Factors Influencing Task Performance.

- * Situation Awareness
- * Workload
- * Attention
- * Alertness
- * Crew Skill & Coordination
- * Anthropometrics & Biomechanics
- * Physical Environment
- * External Communications & Structure

Situation awareness is concerned with maintaining within the crew knowledge of the state-of-affairs of special or critical significance in the context of ongoing activities. Knowledge of importance may relate to systems and process status and trends, ongoing tasks being performed, the interactions between states, trends, and tasks, and all within the immediate past, present, and near-future time frames (see 15 and 16). The use of digital technology to provide integrated graphic displays in which data from a variety of sources are combined in a meaningful manner may enhance crew situation awareness.

Workload involves the mental and physical activities of the crew with regard to its capabilities. Highly automated control systems may result in a very low workload, possibly causing alertness problems. Conversely, failure of an automated control system may result in a workload level that exceeds the crew's capability to respond properly. The result may be a crew performance problem, possibly impacting on safety.

Attention is concerned with the crew concentrating on one aspect

of the current situation when that aspect involves information that should be known at that point in time. As an example, advances in alarm prioritization and filtering should help direct crew attention to relevant elements of the situation (12 and 13).

Alertness involves maintaining in the crew a wide-awake attitude.

Crew skill is concerned with proficiency in performing required tasks. Crew coordination deals with providing capabilities for the crew members to act in a harmonious combination, when required, and to perform tasks and activities in an acceptable and timely fashion. Coordination may involve communications, leadership, interpersonal and group climate, conflict resolution, and decision-making. The commercial airline industry program in cockpit resource management has been successful in improving crew coordination (e.g., 17 and 18).

Anthropometrics and biomechanics involve human body dimensions and movement patterns, respectively. Such dimensions are important in the design of compact work station. Improper work station configuration may result in fatigue and difficult-to-see-and-reach displays and controls.

The physical environment of an advanced control room must be designed so that room lighting does not cause glare and reflections from display surfaces. Noise levels must be controlled so that the crew members are able to communicate verbally (important for crew coordination). Ambient temperature, air flow, and other factors may influence the level of alertness, especially during night shifts (19).

External communications and structure is concerned with the support capabilities available and the methods by which information is communicated to the crew. Technical support centers and emergency operating facilities are examples of support and information sources. An issue with digital technology is the method by which information is communicated to the crew. Will voice communication be the primary method, or will the information be delivered on display surfaces?

POTENTIAL SAFETY ISSUES

Changes in crew tasks and the human-system interfaces in upgraded and advanced control rooms are expected to impact on plant safety. The prediction is that overall impact will be positive, based in part on the experiences in other industries (e.g., commercial air transport, see references 9 and 20). It is important to consider both enhanced safety and challenges to safety resulting from the application of modern technology.

ENHANCED SAFETY

Exhibit 4 lists capabilities provided by digital technology that have the potential to enhance safety.

EXHIBIT 4. Potential Safety Issues: Enhanced Safety.

- * Automation Reduces Workload & Error Opportunities
- * Intelligent Displays
 - Direct Attention & Support Situation Awareness
- * Intelligent Aids
 - Support Problem Solving & Decision Making
- * Computerized Procedures Reduce Workload & Errors
- * Large Displays & Compact Work Centers
 - Support Crew Coordination

Higher levels of automation create opportunities for improved performance. With control automation the number of human execution errors should be reduced. Control actions and sequences of control actions performed automatically are no longer performed by crew members. Thus, opportunities for human errors of execution are eliminated.

Automation should reduce the crew workload. In the case of control automation, the operator's role becomes one of monitoring the progress of the control sequence, as contrasted with the efforts required for manual control. Information automation has the potential of reducing workload, also. Presentation of appropriate information on the displays with minimum thought or action by the crew reduces workload and provides a high level of interface transparency. Three potential advantages of reduced workload include the following.

- * Shifts the emphasis of the crew members from being equipment operators to being system managers. The crew members will have more workload capacity available to develop and maintain an overall perspective on the plant and process (situation awareness), and to implement decisions through the management automation system.
- * Improves crew situation awareness by having available appropriate information provided by the information automation system, and spare mental capacity resulting from reduced workload to place information in proper perspective.
- * Reduces the number of crew members required to control the plant safely through the reduced workload, although regulatory requirements may not permit this advantage to be implemented. In this regard the Electric Power Research Institute's (EPRI) Advanced Light Water Reactors

(ALWR) man-machine requirements specify the main control room design be such that a single reactor operator can handle normal operations (7).

A well-designed intelligent display system should enhance the level of situation awareness within the crew members. For example, important relationships between data can be shown graphically. Explicit rendition of these relationships may make obvious the overall situation and the events occurring in that situation. This overall view supports the role of the crew in managing and directing operations.

Intelligent display systems can provide information that might otherwise be ignored. Information automation can retrieve and present relevant data that the crew might not have time to access or knowledge of availability with conventional display.

The intelligent display systems may direct crew attention to information of importance. Color and other coding cues can be used to highlight the information of importance. Information that does not need to be attended to at that point in time can be suppressed.

Intelligent operator aids and advisors have the potential to reduce the mental workload by performing faster data collection, analysis and interpretation activities. Crew members may have a better understanding of the situation because of the increased ability to monitor ongoing events. The crew should not need to be involved in details being handled by the electronic capabilities. In addition, crew performance quality should be improved. The operator aids and advisors should provide the crew with the best possible information upon which to take actions. The crew, because of lack of knowledge or under the stress of the situation, may not develop information of equivalent quality.

Computerized procedures provide the potential to reduce workload and errors. It is sometimes difficult for a crew member to identify and locate the correct procedure in a paper manual. Another problem is that it is difficult to move between several procedures in multiple event situations. Computer control and presentation of procedural information should reduce the opportunity for human error because the wrong procedural step is not displayed when an action is required. The mental workload should be reduced because the crew member has to attend only to the procedure of importance at that point in time.

Large overview displays and compact work stations should support crew coordination. The crew may be in close proximity to each other because of the small sizes of the compact work stations. This arrangement should facilitate communication. In addition, the overview display viewable by all crew members provides a common view of the plant and process states. This should

facilitate decision-making in which all crew members contribute knowledge.

CHALLENGES TO SAFETY

Exhibit 5 lists issues that may present challenges to safety.

EXHIBIT 5. Potential Safety Issues: Challenges to Safety.

- * Increased Isolation of Crew from Plant & Process
- * Automation Complacency
- * Access to Information in Multi-Dimensional Space
 - High Workload to Obtain Needed Information
 - Keyhole Effect & Fixation Errors
 - Display Thrashing & Navigation Difficulties
- * Human Errors with Automated Systems

The higher level of automation with the associated increase in monitoring activities may cause the crew to be more isolated from the plant and on-going process than with conventional control rooms. The crew may be less involved mentally with plant and process state and transitions. This should not be a problem if adequate situation awareness and alerting mechanisms are provided. However, if technology is not utilized properly to provide this information to the crew, then crew performance problems are possible in event of an emergency.

Automation complacency is closely related to the isolation issue discussed above. Complacency may involve lulling a crew into a state that induces it to over-rely on the automation system. A provocative discussion of some of the problems with automation experienced in the commercial air transport industry is contained in reference 21.

Automation may create additional problems for crew members. It is stated in a report on automated air transport cockpits that, "Pilots who fly the glass cockpit often say that they have never been busier, even though automated cockpits are supposed to relieve workload. As it turns out, the new cockpits realign work more than relieve it," (18, p.50). Billings (9), in an extensive evaluation of aircraft automation, provides similar findings. John K. Lauber, a member of the National Transportation Safety Board, is quoted in reference 21 (p. 67) as saying, "We have left some gaps and run ahead of our ability to teach humans" how to use sophisticated cockpit equipment such as flight management and flight director systems properly. Although the technology and reliability of automated systems are impressive and have

benefited the airlines, "what is missing are principles, rules and guidelines defining the relationships between that technology and the humans who must operate it."

It should be emphasized that NPP control rooms and aircraft cockpits, and the control responsibilities of control room crews and cockpit crews, differ in many ways. For example, high workloads occur in automated cockpits when air traffic control reroutes the flight or alters the runway assignment in the final few minutes prior to landing. Although an analogous situation in NPPs is not obvious, other situations can be described in which crew workload could be higher than in conventional control rooms. For example, the crew may need information not currently provided by the display system. The crew may need to think about where the information is stored, how to retrieve it, and operate numerous controls to display the desired information. In contrast, the desired information may be on the display board in a convenient location with the conventional control room.

Another potential problem with automation is that crew members may not have the necessary skills to perform manual actions if the automated system malfunctions or fails. Of course, training, procedures and operator aids can provide the knowledge required if manual intervention is required.

High levels of automation may create a problem related to low workload and operator alertness. In fact, the levels of automation currently in use in NPPs result in extended periods of time during which crew workload is low. A survey at five Swedish NPPs showed that control room operators estimate their major task to be purely passive monitoring (22). Overall, the operators estimated that 55% to 68% of working hours involved this task, while controlling the process required only about 5% of their time. The operators reported an even higher proportion of time during the night shift being devoted to passive monitoring (between 63% and 73%). With increased levels of automation being incorporated into advanced control room designs, unless meaningful operator activities are planned, passive monitoring will occupy an even larger percentage of operating crew work time.

A low workload level is associated with reduced levels of alertness. Workers who are not alert, given the opportunity, tend to make more errors, and be more accident-prone (19). In fact, a low workload level occurring during the night shift may increase the probability of operators dozing on the job (e.g., the occurrence of sleep recorded through EEG-measurements among Swedish NPP operators) (22).

Digital technology capabilities open the window to the crew for tremendous amounts of information. Access to this information in a timely fashion, however, may present challenges to the crew.

Some questions have been reported in using electronic information and display capabilities (23, 24, and 25). Examples of problems include:

- * Inability to access desired displays and controls, and/or excessively long selection paths to get to the desired display (the interfaces to the data are not transparent, causing increased mental workload).
- * High mental workload related to display selection and window management.
- * Crew members getting "lost" in the display structure.
- * Display "thrashing," where relevant plant parameters and controls are distributed across multiple displays forcing crew members repeatedly to switch among displays to - accomplish a given task.
- * Tunnel vision and keyhole effects where the crew member becomes narrowly focussed and loses situation awareness.
- * Mode errors, where a display is misinterpreted and an unintended action is taken.
- * Fixation errors, where an incorrect situation assessment is formed based on incomplete information.

Human errors are of concern with conventional and with upgraded and advanced control rooms. The key, of course, is the consequences of human error. Some types of error are mentioned above. Many human errors associated with operating controls to display desired information would have limited consequences. The error can be corrected. Errors occurring in response to an emergency plant condition, however, may have more severe consequences and corrections may be more difficult. For example, crew actions based on erroneous expectations of automatic system sequences could have an impact on safety. As with the commercial aviation industry, the nuclear power industry must be concerned about potential human errors caused by higher levels of automation and information access.

REGULATORY POSITIONS

The process by which advanced control room designs are being reviewed and certified by the NRC is very different from the control room reviews performed following the Three Mile Island Unit 2 accident (26). The NRC Action Plan developed in response to that accident required NPP licensees and license applicants to perform detailed control room design reviews (DCRDRs) to identify and correct design deficiencies (27). The NRC issued guidelines for use by utilities in conducting DCRDRs (NUREG-0700). This document consists of human factors guidelines adapted to NPP control rooms, and additional guidelines as required.

Exhibit 6 identifies some of the factors involved in advanced control room design certification.

**EXHIBIT 6. Regulatory Positions: Advanced Control
Room Design Certification.**

- * Human-System Interfaces (HSI) Not Designed
- * NRC Reviewing Design Process & Results
- * Guidance Documents
 - NRC NUREGs
 - DoD Standards & Specifications
 - EPRI Guidelines
 - IEEE, IEC, & ANSI Guidelines & Standards
 - NASA Guidelines

Detailed control room and human-system interface designs are not yet available for the new evolutionary and passive reactor plants. This is creating problems for the NRC and the nuclear power industry. NUREG-0700 does not contain adequate guidance with regard to higher levels of automation and electronic information management and display. Other NRC and industry consensus standards and guidelines are limited in applicability. The human-system interface technology is advancing rapidly, and the human factors guidance documents have not kept pace.

The NRC has adopted a control room design certification process that involves evaluation of the preliminary design and the implementation plan that describes the human factors program elements required to develop a detailed design specification (26). The design elements addressed in this review process include:

- * Human factors engineering (HFE) program management.
- * Operating experience review.
- * Systems functional requirements analysis.
- * Allocation of functions.
- * Task analysis.
- * HSI design.
- * Plant and emergency operating procedures.
- * HFE verification and validation.

There are some guidance documents available that are useful. These documents include NRC NUREGs (e.g., 0700 and 0800), military standards (e.g., AR 602-2, MIL-STD-46855 and MIL-STD-1472D), EPRI guidelines (e.g., NP-3659 and NP-3701), IEEE guidelines (e.g., 1023), international guidelines (e.g., IEC 964), ANSI standards (e.g., ANSI HFS-100), and NASA standards (e.g., NASA-STD-3000). This list of guidance documents is not intended to be complete. Rather, the list demonstrates that some guidance is available, but that the guidance has not been synthesized. The control room designers and NRC reviewers do not have a document equivalent to NUREG-0700, although Brookhaven

National Laboratory is preparing a NUREG document to provide guidance. A problem with this proposed NUREG is that only limited guidance is included about automation and electronic display and information management.

Guidance documents are available to provide reasonable guidance for three of the design process elements (see Exhibit 7). Even

**EXHIBIT 7. Regulatory Positions:
Guidance Document Applicability to Design Process Elements.**

- * Reasonable-Guidance
 - Human Factors Engineering (HFE) Program Management
 - Operating Experience Review
 - System Functional Requirements Analysis
- * Limited Guidance (Areas Not Fully Addressed)
 - Allocation of Functions (Automation Trade-offs)
 - Task Analysis (Cognitive & Decision Tasks)
 - HSI Design (Limited Technology Experience)
 - Plant & Emergency Operating Procedures (Computerized)
 - HFE Verification & Validation (Measurement & Extent)

through technology impacting on automation level and the human-system interface is changing rapidly, the HFE program management process and methods for system functional requirements analysis are reasonably robust. The available guidance has been applied in many different system development efforts in a variety of application areas, including military systems incorporating modern technology. The process and methods have been found to be somewhat insensitive to detailed technology concerns.

The operating experience review element is not impacted significantly by digital technology. This element involves review of the literature and operator interviews. Methods for reviews and interviews are known, and not complicated.

Only limited guidance is available for the remaining design process elements (see Exhibit 7). The functions allocated to the crew are influenced by automation decisions. There is continuing research by NASA (18,28, and 29) and the Germans (30), among others, related to automation effects. Billings (9) has developed limited guidelines for human-centered automation, but these are under review.

Although task analysis has been part of the HFE field for many years, it is only recently that cognitive task analysis has come into existence. Cognitive task analysis is especially useful in

describing decision-making activities. Because of limited experience with cognitive task analysis only limited guidance is available.

Guidance in the HSI design and operating procedures elements is limited because of rapid technological changes. Adequate guidance documents have not been developed to reflect fully such features as intelligent operator aids and computerized procedures integrated with plant process data.

Because of the limited guidance available, the nuclear power industry and the NRC are placing great importance on the HFE verification and validation (V&V) element. Since guidance for some of the other design process elements is somewhat limited, V&V provides an opportunity to determine if the system design is safe. Two of the important limitations in guidance available for V&V relate to measurement and extent of the evaluation. Many different human and system parameters may be measured during the V&V activity. A key question not answered completely in guidance documents is, What should be measured? It is very expensive to collect, analyze, and interpret measurement data on complex systems, such as control rooms. The evaluator needs guidance on important performance indicators, especially as related to automation and modern HSIs.

A related issue is the extent of the V&V. The possible conditions that can be tested are many. It is not feasible to evaluate every possible condition and scenario. Only limited guidance is available on how to select those conditions and scenarios that should be subjected to V&V.

RESEARCH AND OTHER ACTIVITIES

It is clear from the discussion above that modern technology is going to have a major impact on the crew's tasks, and on the human-machine interfaces in the control room. Technological innovations in automation, electronic information and display, and compact work stations clearly provide the opportunity for the crew to concentrate on managing and directing plant operations.

The crew need not be as directly involved in detailed control activities as in the past. The crew members should have a greater awareness of the state of the plant and process at all times, be supported by electronic aids when difficult problem-solving and decision-making are required, and have adequate workload capacity to devote to high priority issues. Thus, the performance of the crew and the entire NPP system has the potential of being much better than with conventional NPPs.

These desirable performance results, however, will not occur if the nuclear power industry introduces these technology features without adequate concern for the needs and wants of the various

types of people who will be operating the plants. For example, news stories describe an airline crash in which a mode error may have occurred. It is suggested that the flight crew thought the autopilot was in one mode when, in fact, it was in another mode (31).

These observations about problems with technological features should not be viewed as pessimistic. Other related activities, such as fossil fuel power plants, electronic dispatch control centers, process control plants, military aircraft, and commercial airliners have been upgraded successfully. Although some human performance problems have been encountered, the overall results have been positive. For example, a recent report (20) states that no commercial airline aircraft loss has occurred, to date, from a fully coupled, automatic landing. It seems reasonable that the nuclear power industry should be even more successful in its move to new control room technology.

There are several research and other activities that should be considered in making the transition to upgraded and new control rooms. Accomplishing these efforts and incorporating the results into control rooms will increase the likelihood of successful operations. Exhibit 8 contains a list of some of these efforts.

EXHIBIT 8. Research and Activities Needed to Support Transition to Digital Technology.

- * Knowledge Transfer (Lessons Learned)
 - Foreign Nuclear & Fossil Plant Upgrades
 - Commercial Aviation, DoD & NASA
- * Automation: Crew Involvement & Skill Maintenance
- * Information Management, Presentation & Control
- * Situation Awareness & Decision Making
- * Human Errors in Design Process
- * Verification & Validation

A large amount of relevant knowledge applicable to NPP control room upgrades and new designs is available. It is important to review, access, and document relevant experiences, lessons learned, opportunities for enhanced performance, and problems related to human performance in nuclear power and other industries in which modern technology is in use or in test. Examples of information sources include:

- * nuclear - EdF N4, Japanese ACR designs, CANDU, Sizewell B, Swiss experience with the AWARE computerized alarm handling system.

- * Commercial aviation - 747-400, MD-11, A320.
- * U.S. Navy - Tactical Decision Making Under Stress project which was created in response to the USS Vincennes incident in which the ship's crew accidentally shot down a commercial airliner.
- * NASA - programs dealing with automation.
- * U.S. Air Force - Pilot's Associate program.

The knowledge available is of importance to all planned control room upgrades and new designs. Therefore, it may be desirable for several elements of the nuclear power industry and concerned government agencies to work together to develop the lessons learned and knowledge transfer.

Research is needed to develop guidance and guidelines on how to keep the crew alert, knowledgeable about system goals, aware of plant and process state and availability of resources to handle abnormal and accident conditions at all times. This activity is needed because high levels of control automation may adversely affect the potential for acceptable human performance. In addition, guidance and guidelines are needed on maintaining crew skills so that in situations of automated system failure, the crew can intervene successfully.

Guidance and guidelines are needed on how crew members should control information displays and access needed information. This activity is important because crew members should not be distracted from their primary tasks by the need to think about and operate controls concerned with data access and presentation.

Situation awareness is important with increased levels of automation. A problem is that clear guidance on how to provide such awareness is not available for NPP applications. Similarly, operator aids to support decision making are feasible technically. Guidelines are needed on their use.

Guidance and guidelines are needed on how to detect and reduce human errors in the design process. With control and information automation, some opportunities for human errors may be displaced from control room operators to automated system and electronics information and display designers.

A meaningful evaluation program to verify and validate the upgraded and new control rooms is needed. The industry needs to be sure that the challenges have been handled successfully. As with the knowledge transfer activity, it may be desirable for several elements of the nuclear power industry and the government to work together to develop and implement a V&V program. Important changes in control room design caused by technological advances in automation, information management and control, "soft" control, compact work stations, and large overview displays are not unique to any one advanced control room design.

Some of the V&V issues are generic to nearly all designs. Because of the potential magnitude of the V&V effort, a common approach is worth consideration.

CONCLUSION

Modern technology is being introduced into NPP control rooms. The capabilities provided by this technology will enhance crew performance, resulting in increased plant safety, reliability and availability. Challenges to crew performance exist, however, because of the important shift in the crew's tasks, and methods for communicating with the remainder of the system. The nuclear power industry should learn from relevant experiences in and out of the industry. The industry should also support development of guidance on the especially challenging issues facing the introduction of modern technology.

ACKNOWLEDGEMENT

Material presented in this report is based, in part, on an unpublished presentation entitled, "Challenges and Opportunities of Modern Technology on Crew Performance," made by the author at the American Nuclear Society Topical Meeting, "Nuclear Plant Instrumentation, Control and Man-Machine Interface Technologies," Oak Ridge, TN, April 18, 1993.

REFERENCES

1. J. Easter, and L. Lot. Backfitting a fully computerized alarm system into an operating Westinghouse PWR: A progress report. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York; IEEE, 1992.
2. B. Lee, K. Chang, and J. Yang. Korean Standard Nuclear Plant: Safer, simpler, easier to build. "Nuclear Engineering International," Aug. 1992, pp. 29-35.
3. J. Malcolm, G. Hinton, and J. Pauksens. Human machine interface design in new CANDU control centers. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
4. K. Iwaki. Progress of I&C system and control room design in TEPCO nuclear power plants. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.

5. D. Harmon. Nuplex 80+: An evolutionary approach to meeting ALWR requirements. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
6. J. Carrera, J. Easter, and A. Negus. Human engineering considerations in the man-machine interface design for the AP600 plant. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
7. R. Fink, and E. Rumble. ALWR man-machine interface requirements. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
8. K. Monta, J. Itoh, and M. Makio. An advanced man-machine system for PWR nuclear power plants. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
9. C. Billings. Human-centered aircraft automation: A concept and guidelines. "NASA Tech. Memo." No. 103885, Aug. 1991.
10. J. Ketchel, and J. Naser. A human factors view of new technology in nuclear power plant control rooms. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
11. K. Kang, S. Cheon, and S. Chang. Development of an expert system for performance evaluation and diagnosis in nuclear power plants. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
12. R. Fink, R. Williges, and J. O'Brien. Appropriate choice of alarm system technologies: EPRI research. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
13. D. Harmon, and T. Starr. Alarm and status processing and display in the Nuplex 80+ Advanced Control Complex. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
14. M. Peyrouton, and M. Pirus. Progress on N4 I&C Architecture. "Proceedings of the Topical Meeting on Nuclear Plant Instrumentation, Control, and Man-Machine Interface Technologies." La Grange Park, IL: American Nuclear Society, April 18-21, 1993.
15. N. Sartar, and D. Woods. Situation awareness: A critical but ill-defined phenomenon." International Journal of Aviation Psychology," 1991, Vol. 1, pp.45-57.

16. R. Pew, Y. Tenney, M. Adams, A. Huggins, and W. Rogers. A principled approach to the measurement of situation awareness in commercial aviation. BBN Report No. 7542 prepared for NASA Langley, June 1991.
17. R. Helmreich, J. Wilhelm, J. Kello, W. Taggart, and R. Butler. Reinforcing and evaluating crew resource management: Evaluator/LOS Instructor Reference Manual. NASA/University of Texas Tech. Manual 90-2, Rev. 1, June 26, 1991.
18. D. Hughes. Pilots, Research Studies Give Mixed Reviews to Glass Cockpits." Aviation Week," March 23, 1992, pp.50-51.
19. M. Moore-Ede. Observing human operator alertness at night in power plants. "1988 IEEE Fourth Conference on Human Factors and Power Plants." New York: IEEE, 1988.
20. P. Proctor. Industry Group Begins Worldwide Safety Push. "Aviation Week," June 7, 1993, pp.99-100.
21. E.H. Phillips. Pilots, Human Factors Specialists Urge Better Man-Machine Cockpit Interface. "Aviation Week," March 23, 1992, pp.67-68.
22. K. Dahlgren. Shiftwork, work scheduling and their impact upon operators in nuclear power plants. "1988 IEEE Fourth Conference on Human Factors and Power Plants." New York: IEEE 1988.
23. E. Roth, R. Mumaw, and W. Stubler. Human factors evaluation issues for advanced control rooms. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
24. D. Woods, S. Potter, L. Johennesen, and M. Holloway. Human interaction with intelligent systems: Volume I - trends, problems, new directions. Columbus, OH: The Ohio State University CSEL Report 1991-001, May, 1991.
25. D. Woods. The price of flexibility. "Proceedings of International Workshop on Intelligent User Interfaces." ACM, Jan. 1993.
26. W.T. Russell. Advanced Reactor Control Room Design and Licensing. "1992 IEEE Fifth Conference on Human Factors and Power Plants." New York: IEEE, 1992.
27. A. Ramey-Smith. Nuclear Power Plant Control Room Design Reviews: A Look at Progress. "1985 IEEE Third Conference on Human Factors and Power Plants." New York: IEEE, 1985.

28. J.P. Jenkins. Human Error in Automated Systems: Lessons Learned from NASA. "Proceedings of the Topical Meeting on Nuclear Plant Instrumentation, Control, and Man-Machine Interface Technologies." La Grange Park, IL: American Nuclear Society, April 18-21, 1993.
29. B.W. Henderson. NASA Ames Pushes Automation Toward Human-Centered Design. "Aviation Week," March 23, 1992, pp.69-70.
30. M. Mecham. German Center Studies Cockpit Automation. "Aviation Week," July 26, 1993, p.38.
31. J.M. Lenorivitz. French Government Seeks A320 Changes Following Air Inter Crash Report. "Aviation Week," March 2, 1992, pp. 30-31.

4.3.1 Questions: Dr. Lewis F. Hanes

QUESTION: **FRED PAULITZ (NRC):** What is being done to separate information that maintenance should have and that may be different from what the operator needs regarding both system and component failures?

DR. HANES: I can't really respond as to exactly what is being done, but I can comment on what should be done. Certainly there should be a task analysis of the operator information needs with regard to the maintenance, to the status of systems, and to the status of components. Where there is a need for the operating crew to have this information available, it should be made available.

With regard to the maintenance people, again, there ought to be a task analysis of what the information needs are of the maintenance people and where there is an overlap, there should be some consolidation. People should address how the information is being presented to make sure that there aren't any conflicts, that both people share the information, and that access to that information, control of that information, is handled in a proper manner. So, really task analysis and then evaluation of the results of those analyses is the way, in my judgement, that it should be handled. I really can't speak to how the various vendors are handling this.

5 SOFTWARE ENGINEERING FOR HIGH INTEGRITY SYSTEMS

The technical session on software engineering focused on techniques for improving the software development process. The premise of all the speakers in this session was that software developed using the methods generally in use today does not meet the needs of high integrity systems. The speakers identified two general areas of concern. The first two speakers focused on the need to provide accurate specifications. The last two speakers moved the discussion into the need to design, implement, and test systems to meet the specifications.

Dr. John Knight (University of Virginia) focused on the need to separate the functions of systems and software engineers, and to ensure an effective interface between them. His point was that software engineers do not have the system or application knowledge to make decisions about the behavior of the full system. Deciding what the system should do under each set of circumstances is the job of the systems engineer. Software engineers need to receive a precise system specification including all the assumptions and constraints that the systems engineer expects the software to maintain.

Dr. John McHugh (Portland State University) focused on the need to provide a non-ambiguous communication mechanism for the systems and software engineers. This mechanism should be some form of formal specification understandable both to system and software engineers. Dr. McHugh acknowledged the difficulty of learning formal specification languages and understanding formal specifications. While the use of explanatory text can mitigate some of these problems, it introduces a new problem in defining which of the two specifications, the formal or the English language one, controls.

Mr. Robert M. Poston (Interactive Development Environments) focused on the need for and benefits of providing tool support for developing a formal specification. Developing a formal specification is difficult and labor intensive. Mr. Poston discussed the use of tools that make the process less "user-hostile" by allowing the developer to create the specification in diagrams which the tool can transform into an easily understood notation. The developer can then check the produced specification. Another advantage of a formal specification in a defined notation is that a test generation tool can take that specification and produce test cases.

Dr. Barbara B. Cuthill (NIST) moved the discussion from the requirements specification to the design and implementation portions of the software lifecycle. Specifically, she discussed general features of object-oriented design (OOD) and C++ development. Dr. Cuthill enumerated risks and benefits of using OOD and C++ with respect to specific criteria important to safety-critical systems such as functional diversity. While she did not form a final conclusion, she presented many of the issues that need to be addressed as OOD and C++ become more widely used in industry.

The need for traceability of the requirements, design, code and test cases back to a set of non-ambiguous specifications provided a common thread that all the speakers identified as important and discussed in some form. Dr. Knight began by discussing the need for systems engineers to establish complete specifications. Dr. McHugh continued by discussing how systems engineers can provide non-ambiguous specifications. Mr. Poston discussed the methods for generating test cases traceable to the specifications while Dr. Cuthill discussed methods for designing and coding the software while maintaining traceability. While each of the speakers also addressed other important issues in the development of safety-critical systems, each one returned to this theme of traceability and maintaining the link between the phases of the software development lifecycle.

5.1 Interaction Between Systems and Software Engineering in Safety-Critical Systems: Dr. John Knight

Interaction Between Systems and Software Engineering in Safety-Critical Systems

John Knight
University of Virginia

[as edited from transcripts]

There are three areas of concern:

1. When is software to be considered safe?
2. What, exactly, is the role of the software engineer?
3. How do systems, or sometimes applications, engineers and software engineers interact with each other?

My views in this area are different from many others. He thinks that we are experiencing some loss of reliability or dependability because important details tend to be forgotten.

One problem is that the definition of *software safety* is intuitive; software is safe if it does no harm. The problem with this definition is that software engineers need to know precisely what software safety is and what is expected. Software engineers have to know when software is good enough and when it is *not* good enough. Others such as regulatory agencies and the legal system also need to know what to require and how to respond to accidents to lessen the chance of reoccurrence. It is very important that we have a clear, precise definition of software safety.

Software operates in a complete system, and, like most components of a complex system, is never a hazard in isolation. This does not mean that software safety can only be defined in the context of system safety as others have said. Software engineers are not qualified to deal with systems engineering issues such as defining the nuances of the safety actions required for nuclear power systems. Software engineers should not decide the actions a system will take for unspecified inputs. The terms "hazard" and "risk" are inappropriate for software specifications for a safety-critical system; the specification should contain the required treatment of the hazard. To prevent software engineers making these decisions, they need a clear, precise delineation of responsibility. The systems engineer should decide what the software has to do. The software engineer should decide how to implement it.

Unfortunately, many software engineers do not receive complete or useful functional or non-functional specifications. For example, in the aerospace industry, there are requirements that the probability of failure be less than 10^{-9} per hour. Sometimes the software engineers receive precise probabilities like this and sometimes they do not. The systems engineers determine these requirements; however, it is impossible to demonstrate that software meets a failure rate this small. Verifiable levels of quality should appear in more safety-critical specifications.

The specification should support distinguishing responsibilities. The specification should include desired system behavior and undesired events. Components of the specified system should have required assurance levels which may differ by component. In the resulting system, an undesired event should be traceable to a defect in the implementation which is the software engineer's responsibility or a defect in the specification which is the systems engineer's responsibility.

The use of a specification that will require and associate with it various assurance levels will provide a better specification of a safety-critical system. It is not enough to use the system's history to tell if it is safe, because knowing after an undesired event is not good enough. Instead, the software must meet the specification with a certain probability, and the software engineer has to verify that. These are two distinguishable activities: the implementation activity yielding the software and the verification activity assuring the software's conformance to the functional and non-functional specifications. If safety is the satisfactory completion of that verification, then there is a criteria for formally judging software to be safe. It seems reasonable to ask that software be subjected to appropriate analyses like other engineered artifacts.

The clear, precise specification is the communication mechanism between the application and the software engineers. Software engineers should not keep quiet if they see a problem, but society should not depend on software engineers to find functional deficiencies. Application engineers who are concerned with hazards and the ensuing risks, have a responsibility to build complete, formal, unambiguous specifications including all the functional and non-functional requirements.

One problem with this approach comes if the specification pushes the non-functional requirements such as the dependability levels up to this 10^{-8} - 10^{-9} per hour figure. There is no dependable development technology for, no way to guarantee and no way to demonstrate this level of dependability.

Another problem is verifying that the specification itself is adequate. We need better specification capture techniques and better specification analysis techniques. At Virginia, we are trying to deal with these issues. We have defined the notion of a specification capture process. There is a great opportunity for progress in this area because specification capture is typically non-rigorous. We would like to define a process for capturing specifications that is rigorous and dependable, repeatable, and sufficiently rigorous for mathematical analysis. We want to separate software safety issues of concern to the software engineer and specification safety issues of concern to the systems engineer. The introduction of formal specifications has introduced formal analysis techniques but without constraints on the requirements, yet. We want to push rigor into the specifications capture process to have greater confidence in the specifications.

We tried to build a formal specification for a safety-critical system of some magnitude and we started with systems engineering techniques like fault trees and failure analysis, and we found this very difficult. For example, getting software elements into a fault tree caused 2 or 3 major revisions in the specifications which should not have happened, but it did point out the need for more rigorous specifications. One useful technique we found was information flow modeling. Many safety analysis techniques rely on energy flow; therefore, it's appropriate that computer-based systems use information-flow. By tracing information flow paths from a source to a sink, the engineer can find all the potential places that information can be distorted and these locations dictate the specifications on the software needed to react to that distortion.

In conclusion, there are serious safety requirements in more and more applications. Software engineers are not tied to any particular industry and may have to deal with safety issues in any of them. It is very hard to build software well, and building safety-critical software is even harder. We are trying to formalize some elements of this so that we know exactly what we are supposed to build. The major concern is that the software engineer is only qualified to deal with software engineering issues and should not have the responsibility to define systems issues. It is also time to think about licensing software engineers if they are going to build software for safety-critical applications. Finally, formal specification is the right thing to do.

5.1.1 Questions: Dr. John Knight

QUESTION: MARK SERHAL (Wolf Creek Nuclear Oper.): In principle, what is the difference between software specifications developed by application engineers and other specifications developed for hardware components?

DR. KNIGHT: In principle, I don't think there is any difference there. That's actually a very helpful question. One of the things that has been occurring is that there is quite a lot of opposition to formal specification of software because some of it involves mathematics, and a lot of people feel that that's not appropriate. Yet, when it comes to hardware specification it seems to me that engineers in many industries are quite happy with the mathematics of electrical circuits. They're quite happy with the formalness of specifying things using many different kinds of mathematical notation, and software engineering has really, I think, been somewhat behind in that.

So, in response to that question, I think that's exactly the point; there really shouldn't be any difference. The applications engineers understand what they want from any component and specifying it in a manner that is mathematically complete is all that I'm really asking for.

QUESTION: DR. LANCE A. MILLER (SAIC): There is a flaw in your argument. V&Ver's only can do testing, not systems engineering. You say that system engineers are not qualified to assess software for problems.

DR. KNIGHT: No. I said that software engineers were not qualified to make decisions about the application. I have no comment about the systems engineers because I don't know that field.

DR. MILLER: My point is that the V&Vers are the only ones qualified to look for risks and hazards in software. The systems engineers are not qualified to do that.

DR. KNIGHT: Well, the role of verification and validation is really to provide the confidence that you require between the implementation and the specification. The validation step is, of necessity, informal because you can't be sure that what you're looking at is what you wanted. Validation, the process of showing that what you have is what you really want, by definition is informal, and you're stuck at that point. The systems engineer has to look at it and make, basically, a very educated guess about what's going on.

When it comes to verification, we're trying to show that the implementation does what the specification said, and that could be undertaken by basically anybody who understands both sides, the implementation and the specifications.

QUESTION: RAY DiSANDRO (Philadelphia Electric Co.): Prior to licensing a software engineer, what assurance does the customer and the regulator have that a software vendor constructed their software modules in the safest way, i.e., no nested do loops, a qualified math package, and so on?

DR. KNIGHT: Well, I'm not a vendor of software, but I do have an opinion about that. I think it's really quite depressing that software comes with no warranty whatsoever. When you purchase it, you usually have to purchase some sort of maintenance contract with it, and the software industry has somehow gotten away with this. I don't think any other engineering discipline has managed to do that. If other people were to manufacture products and sell them with the degree of quality that the software industry does, and without a warranty, they'd go out of business.

It's quite remarkable that this continues and that the community puts up with it, and I have no response to this question other than that we should all demand better.

QUESTION: WILLIAM D. GHRIST (Westinghouse Electric) If the software of a large system is to consist of a number of loosely coupled modules, rather than monolithic blocks of code, then many of these modules will have functions that are related to the computer system design rather than to the application domain, for example, I/O processing software. Who is responsible for the functional specification of these functions and how does this relate to the separation of responsibilities?

DR. KNIGHT: Another great question. That really is a topic that I should have addressed, but it would have taken a little too long. I will try and encapsulate it here. That really raises the difficulty of trying to get a point over in a rather closed fashion when it's a much more elaborate point than that.

The issue is really the difference between specification, design, and implementation. The question is addressing the problem of a large system, in which we have perhaps many separate processes running which are connected over a network and in which the distinction between the implementation, the design and the specification is rather blurred. The specification then becomes something that is sort of mired in with the design. There are elements of such a system for which we want a formal specification like the interface between the I/O subsystem and something which uses it.

Now, is that part of the design? Well, some would say it is. Is it just specification? Well, no, clearly it isn't. But really it comes down to the central argument that I think we've been trying to make here this afternoon that what we want is a vehicle of communication between those responsible for different elements. If somebody is building an I/O subsystem and they are going to provide that interface to other developers, then what we're arguing for is a formal specification of what that interface is; therefore, applying these kinds of ideas in sort of a recursive way. This gentleman clearly understood what I was getting at and nailed me right to the mast on that one, and that was a great question. Thank you. I hope I answered it.

QUESTION: JIM DUKELOW (Battelle Pacific Northwest Labs): You have lifted a burden from the software engineer and placed it on the system engineer. Are they qualified? Will they understand software well enough to write good specs? Should they, the systems engineers, be licensed to build safety-critical systems.

DR. KNIGHT: Well, I don't like to think of this as having lifted any burdens. I like to think of it as having defined some new ones. We're asking for more formalism on everybody's part.

The software engineer has to analyze the specification. The applications engineer and other kinds of engineers, given the previous question, have to be prepared both to read, write and analyze these kinds of things. So, it's not really a case of moving a burden. It's a case of perhaps defining a burden, that we were thinking of subconsciously or ignoring before, a little more precisely. I don't think that the systems engineer or the applications engineer are being dumped on or given some new kind of responsibility that they are not really trained for. It's really a case of making them think more carefully about what their role is.

And in dealing with the question of interaction, all we're really asking for is that the applications engineer be able to comprehend these specifications and understand the limitations of what's going on. It's not a tremendous imposition to understand some formal notation sufficiently well to communicate. It's not the case here that we're demanding that applications engineers understand all the nuances of software. That's precisely what we're trying to avoid.

QUESTION: You said a reliability requirement in commercial aviation for events that must be extremely improbable indicated a failure probability of less than 10^{-9} . Under what protocol or reliability model does the FAA accept such a claim?

DR. KNIGHT: The requirement is an imposed requirement at the digital system level. It's not a software requirement. And nothing is said about how it's going to be demonstrated. It's up to the manufacturer to do that. Currently, I don't think any of them are doing anything other than claiming that they meet it. Everybody knows that it's very hard to demonstrate; so, there's very little that can be done.

On the regulator's side, it's a statement that this is what is supposed to be met, and on the developer's side, there's generally frank admission that it's just not possible to demonstrate that kind of number. I hope that answered the question.

DR. NASER: If I could add just one more thing, because I asked the same question of one of the aerospace developers of how you do this, in part it is a sort of a consensus process. They've agreed that if I go through a certain process then that satisfies the requirement and then that's as far as it goes.

5.2 The Role of Formal Specifications: Dr. John McHugh

The Role of Formal Specifications

John McHugh*

13 September 1993

The purpose of proof is understanding; the purpose of specification is communication.

Abstract

The role of formal requirements specifications is discussed under the premise that the primary purpose of such specifications is to facilitate clear and unambiguous communications among the communities of interest for a given project. An example is presented in which the failure to reach such an understanding resulted in an accident at a chemical plant. Following the example, specification languages based on logical formalisms and notations are considered. These are rejected as failing to serve the communications needs of diverse communities. The notion of a specification as a surrogate for a program is also considered and rejected. The paper ends with a discussion of the type of formal notation that will serve the communications role and several encouraging developments are noted.

1 Introduction

Any useful complex system is the concern of a number of diverse communities. In [6], Parnas identifies the reviewers required for each phase of the development of nuclear power plant computer systems. In almost every phase, the reviewers are drawn from more than one discipline. For these reviewers to evaluate a common work product, the product must be represented in a way that is comprehensible to all the parties. In addition, each party must reach the same understanding of the meaning of the product. Consider the following example which is based on an actual incident [5] from the early days of computerized process controls.

The specification for the plant read, in part: "In case of an alarm, hold the variables constant and call the system operator." The relevant portions of the system are shown in figure 1. The computer controls valves that admit catalyst into the reactor vessel and cooling water into the condenser. The computer can receive alarm signals from various components of the system, including the low oil level alarm shown as coming from one of the gear boxes in the system.

When the system was first put into operation, it failed with the following scenario:

1. The reactor was charged

*Tektronix Professor, Computer Science Department, Portland State University, Portland, OR

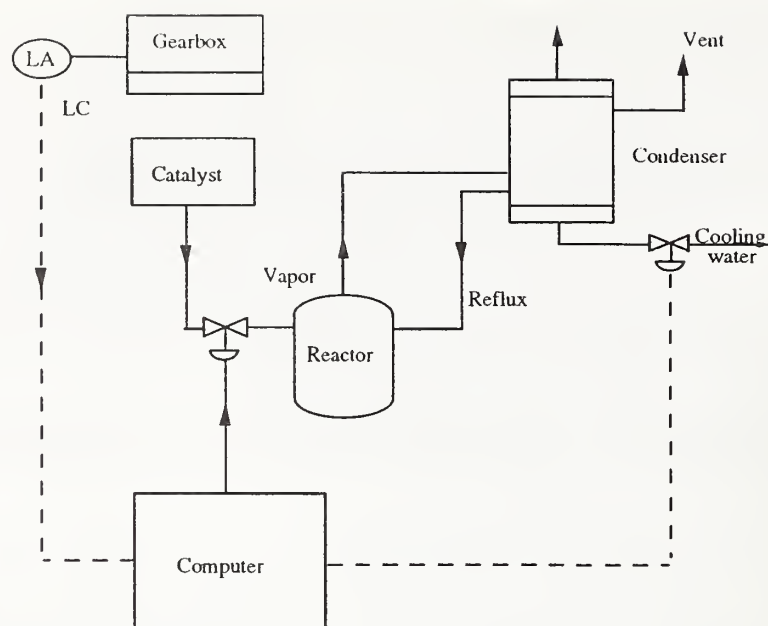


Figure 1: Fragment of a Chemical Plant

2. The catalyst valve was opened to start the reaction
3. The cooling valve was closed to allow the reaction to reach operating temperature
4. A low oil alarm was sensed. This later turned out to be a spurious condition.
5. The valves were held as indicated, the catalyst valve open and the cooling water valve closed, while the operator was notified of the (spurious) low oil condition.
6. The reaction proceeded unchecked.
7. The reactor overheated and vented its contents.

The proximate cause of the mishap was the failure of the chemical engineer who designed the system and the programmer who implemented it to reach a common understanding of the term “variable” as used in the specification fragment. To the engineer, the variables of the system are the temperature and pressure in the reaction vessel. To the programmer the variables were the valve positions under the control of the program. The two views are linked by the control laws of the reactor system and the chemistry of the particular reaction under control, but these were not part of the specification. While this story appears to exhibit an almost unbelievable lack of understanding on the part of both the engineer and the programmer, many developers of specification languages, especially those in purely academic settings seem determined to create languages that can easily lead to similar situations.

2 Standards for Requirements - Does Formalism Fit?

IEC Standard 880 [1] says, in part,

A2.9.1 The software functional requirements shall be presented in a manner which is easy to understand for all user groups. The presentation shall be sufficiently detailed, free from contradiction, and non-redundant as far as possible. The document shall be free of implementation details, complete, consistent, and up-to-date.

A2.9.3 The software requirements document is intended to be used by:

- its authors, i.e. plant specialists;
- the customer, client, or final user;
- the software system development team;
- the software system verification team;
- assessors and licensing personnel;

These meta requirements or requirements on requirements emphasize the communications role by explicitly calling for a representation that is understandable by all parties to the system design and evaluation and by explicitly naming a diverse set of user groups. The additional meta requirements; freedom from contradiction, completeness, etc. are characteristics that we usually ascribe to what computer scientists refer to as "Formal Specifications." These specifications are typically couched in languages based on predicate calculus, set theory, or possibly a higher order logic. While precise and supporting analyses based on theorem proving, the formal specification languages of the computer scientist often fail to satisfy the communications role that is the primary concern of IEC 880 and similar standards. The reasons for this situation are not completely clear. A contributing factor is that the background of many of the members of the formal methods community is in mathematics and logic rather than in engineering. This seems to create a bias towards notations based on predicate calculus and other terse symbologies. This is unfortunate since even most programmers' eyes tend to glaze over when confronted with an \exists or a \forall and while many in the reviewing community have substantial mathematical backgrounds, their mathematics is not directly based in logic. The result is that notation hinders communication rather than facilitating it. Languages such as "Z" which is particularly rich in notation have achieved limited success, especially in Britain and Europe, primarily through extensive educational efforts. Even with these efforts, it appears that relatively few potential users who have taken a "Z" course actually adopt the notation.

Interspersing formal language with informal prose is a substantial aid to improving the readability and understandability of a formal description, especially a terse or highly symbolic one. Indeed, much of the success that "Z" has had may be due to the practice of embedding "Z" specifications in expository material. It has been suggested that parallel formal and informal specification or requirements documents be developed and maintained. This brings to mind the adage:

Never go to sea with two chronometers; take one or three.

If we have both prose and formal definitions, one must be considered to be *the* standard. Typically, prose is not completely unambiguous. If the prose version is considered to be the standard, it is likely that the formal version represents one (of possibly many) allowable interpretations. On the other hand, if the formal language is the standard, it is incumbent on all users to understand it so as to avoid reaching an understanding of the prose that is at odds with the formal language. In this case, the prose can, at best serve as illumination, rationale, and example material. In either case, we must allow for errors in translation and be prepared to appeal to the primary standard in order to resolve any problems that arise.

The precise semantics associated with formal specification languages is a specific antidote for ambiguity. In addition to avoiding ambiguity, we would like requirements specifications to be consistent and complete. By consistency, we mean that the specifications do not contain any contradictory requirements. Completeness is more subtle. Intuitively, a set of requirements is complete if it covers all possibilities that might arise in the operation of any system built to satisfy the requirements. Failing to specify the behavior for some portion of an input range can be detected by analysis of the requirements, but possible situations that are not reflected at all in the requirements set cannot be detected by formally analyzing the set. For example, if the requirements state that a system must operate over a given range of temperatures, but do not say what its behavior should be outside that range, analysis of the requirements should be able to identify the omission. On the other hand, if the requirements make no mention of a need for the system to operate in a corrosive atmosphere, no amount of analysis of the requirements will detect the omission. Formal specification languages do not automatically ensure either consistency or completeness, but they often provide ways in which a given set of requirements can be analyzed to ensure that it is consistent or complete in the sense of specifying a total behavior.

A formal specification that contains a contradiction is said to be unsound. The logician's concern is that the contradiction could be used to generate a false hypothesis which, in turn could be used to prove any desired proposition using the property of logical implication that says that **FALSE** implies **Anything**. As a consequence of this fear, some theorem prover based systems require an extreme form of completeness that leads to over specification. In developing requirements we are usually careful to avoid specifying a solution to the problem. We want to leave the discovery of an appropriate implementation to a later time and, usually, to a group of designers who are closer to the details of the particular components that are available to work with. At the requirements level, we are willing to take the risk that there is no way to realize a particular function and do not feel the need to explicitly demonstrate that this risk is unfounded by including in the requirements the details of one way to realize the function. Tools that enforce this level of detail are of little use in formalizing requirements since it makes us specify in great detail precisely those things about which we care least. Since there is no way within the formalism to indicate which portions of the specification are critical and which ones are not, the immaterial becomes an essential part of the system to be preserved as the system evolves or is moved to a new host.

An extreme case of over specification is the use of an existing implementation as a *de facto* formal specification for a new implementation. Brooks [4, Ch. 6] discusses the consequences of this approach in some detail, noting that in emulating the IBM 1401 on the IBM 360 it was necessary to preserve undocumented functionality as much of the 1401

programming community relied on these accidental behaviors of the original 1401.

3 Animation and Synthesis

Many developers of formal specification languages provide facilities for executing their specifications, either directly (animation) or by the mechanical production of executable code from the specification. In either case, this is not appropriate at the requirements level as it necessitates a high degree of over specification because the specification must embody a complete solution to the requirements. Even at the design specification level, this approach is dubious for safety critical systems with real-time constraints as it is unlikely that either the animator or the synthesizer will be able to meet non-functional requirements for performance, robustness, etc. In the case of synthesis, the synthesizer may contain additional domain specific knowledge that will be embodied in the resulting program, but is not easily available for examination and review. David Parnas makes an interesting observation about the execution of specifications in a recent paper [7]. In the paper, he discusses the role of mathematics in programming, taking the engineering viewpoint that design is a creative activity and that mathematics is primarily useful to analyze and validate designs.

Program derivation from requirements appears analogous to deriving a bridge from a description of the river and the expected traffic. Refining a formal specification to a program would appear to be like refining a blueprint to produce a bridge. Engineers always make a distinction between the product and the description of it. This seems to be lost in the computer science literature on programming and software engineering.

4 What do we really need?

If we believe that the primary goal of specification is improved communication among designers, customers, users, developers, and regulators, then it is clear that some form of mathematically based specification and requirements representation is useful. Only a mathematically based notation can ensure freedom from ambiguity and only an unambiguous notation can be successfully analyzed for properties such as completeness and consistency. As we have seen, the mathematical jargon usually associated with specification languages directly rooted in a formal logic seems to hinder communication rather than facilitate it. Fortunately, there are alternatives.

Parnas uses formal mathematical specifications in his work, but he attempts to cast them in a form that facilitates communication. His tabular specification form has been used successfully in the design of the shutdown system for the nuclear power plant at Darlington, Ontario [2, pp 46-69]. Leveson has developed a highly readable notation for specifying the requirements for complex systems. This notation has been used to develop the requirements specification for a collision avoidance system for the FAA [2, pp 150-162]. The same study [3] gives other examples, some successful and others not, of the use of formal specifications in the development of computer systems. This study is interesting in that it points out the role of non technical factors in the application of formal methods to a variety of projects. The successful projects seem to have a high level of organizational

vision and a willingness to experiment with untried techniques as well as a realization that business as usual will not produce the needed results.

The study provides evidence that formal specifications can serve the primary role of facilitating precise communications that we have outlined. The IEC 880 meta requirements that requirements specifications be free from implementation details and that they be up to date are subjective and can only be met through suitable administrative and managerial procedures. The meta requirements that specifications be complete and consistent can largely be met by analyzing formally represented requirements with suitable tools.

5 Conclusions

Formalism for its own sake is useless. The formal methods community has often been its own worst enemy, placing emphasis on issues that are of little importance to its customers. Ignoring the communications aspect of specifications will, at best result in software that does the wrong thing perfectly.

Determining what software ought to do seems to be the most difficult part of the development process. The precision of formally based specifications can aid in this process. Once a suitable requirements specification has been developed and agreed upon, the ability to analyze formal specifications and to show that implementations are consistent with them offers added advantages.

References

- [1] International Electrotechnical Commission. Software for computers in the safety systems of nuclear power stations. IEC Standard, 1986. Publication 880.
- [2] Dan Craigen, Susan Gerhart, and Ted Ralston. An international survey of industrial applications of formal methods, volume 2 case studies. NISTCGR 93/626, U.S. Department of Commerce, National Institute of Standards and Technology, March 1993.
- [3] Dan Craigen, Susan Gerhart, and Ted Ralston. An international survey of industrial applications of formal methods, volume 1 purpose, approach, analysis, and conclusions. NISTCGR 93/626, U.S. Department of Commerce, National Institute of Standards and Technology, March 1993.
- [4] Frederick P. Brooks, Jr. *The Mythical Man-Month*. Addison Wesley, 1975.
- [5] T. Kletz. Human problems with computer control. *Hazard Prevention*, pages 24–26, Mar/Apr 1983.
- [6] David L. Parnas. Software for computers in the safety systems of nuclear power stations. Final Report for contract 2.117.1, Computing and Information Science, Queen's University, Kingston, Ontario K7L 3N6, March 1991. Based on IEC Standard 880.
- [7] David L. Parnas. Mathematics of computation for (software and other) engineers. In *Third International Conference on Algebraic Methodology and Software Technology, AMAST*, The Netherlands, June 1993. University of Twente.

5.2.1 Questions: Dr. John McHugh

QUESTION: DORELLE RAWLINGS (Sorrento Electronics): Why can't both prose and formal definitions be the standard? The intended meaning must be consistent with both forms of specification so the two forms provide constraints on the meanings of the specification.

DR. McHUGH: In principle, yes; in practice it usually doesn't work that way. Every time that I have seen an attempt to maintain both definitions as joint standards the question of this doesn't say the same thing as that arises at some level of detail or subtlety, and ultimately you wind up having to decide which one is right and which one has to change. It's often a much more difficult process to do it in practice.

QUESTION: Nuclear power safety systems are simple. Do you really need formal methods?

DR. McHUGH: Okay. First of all, I'd like to point out that the existence of this question is the counter-example to Bob Poston's statement that everybody agrees that formal methods and formal specs for these systems are desirable. I can only offer the evidence that for a conference that has a fairly strong emphasis on the formal aspects, at least 200+ people showed up to discuss it. I have never built a nuclear reactor safety control system. If they are all that simple I don't understand why it takes years to license them and years to evaluate them.

QUESTION: I have a question I'd like to address because I have been involved in the building of a nuclear reactor protection system using digital methods, and I think maybe the problem is that nuclear reactor protection, the actual nuclear reactor protection functionality, is not all that complicated, but that doesn't mean that the systems aren't.

DR. McHUGH: Right.

QUESTION: What you find if you actually get out into the real world and try to do this stuff is that the complexity is not in implementing the protection functions. The complexity is in implementing software that supports the architecture of the system itself, things like input/output processing and that sort of thing, which are not domain-specific but are specific to the system.

DR. McHUGH: I think that addresses it in a very good way. I mean, if I were to give this programming, as an example, in an introductory computer science class, the students would come up with a very naive view of the problem solvable in some tens of lines of code. But, in fact, if you tried to demonstrate that the system would work in the real world with the real sensors, the timing constraints, the possibility of failures in various components and so on that you have to deal with, the protection of that simple concept becomes a much more complex one. Once it starts to get more complex, you start running the risk of getting it wrong and getting something that will not work when you need it to, or work when you don't need it to, or all of the other things. While conceptually we're dealing with something which is very simple--let the man with the ax cut the ropes so the rods drop in at the right time--in fact we're dealing with something that is much more difficult to realize than that.

QUESTION: There are many formal methods, (e.g., VDM, Z). Which one is the most cost-effective, reliable and practical? How much does the formal method buy you for system reliability in the practical world? Can you provide some real-world examples?

DR. McHUGH: Okay. I'll take those in reverse order. For the real-world examples what you should do is you should obtain the study published at NIST, *An International Survey of Industrial Applications of Formal Methods*, NIST GCR 93/626. Dan Craigen (ORA Canada), Susan Gerhart (Applied Formal Methods) and Ted Ralston (Ralston Research Associates) conducted this study. They looked at applications of formal methods in a variety of commercial and semi-commercial projects and have written a rather nice report that I've seen abstracts of and presentations from, and I suggest you get a hold of that. That will give you an idea of the scope of some of the things that have been done.

The best figures that we have on quantitative improvements in reliability come out of some of the cleanroom experiments that have been done at IBM Federal Systems Division and by a facility at the University of Maryland. They typically see an order of magnitude decrease in errors that manifest themselves at the system level compared to the previous techniques that IBM had been using for those kinds of developments at a marginal increase in system development cost. I mean not a statistically significant increase, but right in the same ball park as business as usual. Why everybody doesn't go for that order of magnitude I've never been able to understand.

As far as effectiveness and cost-effectiveness, give me a break. We don't know the answer to that question for any kind of software development methodology. Nobody is willing to put up the money required to develop enough replicates of large-scale systems using different methodologies and comparing the results. I mean we've got lots of anecdotal evidence that lots of things seem to provide some improvement, but I can't give you solid comparisons. I wish I could. If somebody wants to help me develop a dozen 2-million line systems as replicates of each other to get some firm measurements on, I'd love to do it. I haven't been able to figure out anybody who is willing to pay for that kind of thing, and it's really hard to do controlled experiments in this area. Things don't scale. Everything works pretty well for a 100 line or a 200 line program. And when you go up into the few hundred thousand lines it's hard to get comparisons because they cost too much to do two of them, or three of them, or 200 of them, to get control on the experiments.

So, there is a bit of literature emerging, but not a whole lot, in those areas. The people who are using them seem to be happy. Read the report that Dan and Susan and Ted put together.

QUESTION: **GUSTAV DHALL** (OECD Halden Reactor Project): To show the safety of a spec, would it then be necessary to model the process in the same formal language as the spec? In this way one can prove the safety assertion on the specification.

DR. McHUGH: That would be a nice luxury. Typically you don't have it. But, yes, what you'd really like would be a model of the environment to play your model of the system against and prove the properties of them. The trouble is that we don't speak the same language when we talk about modeling the physical system. I don't know how to represent, in the kinds of specifications that I use for programming language logic, the physics of the system that is being

controlled by the process control. Typically what we do is abstract from some other kind of model of the exterior system some input/output behaviors and then verify that our model specification of the system control behavior matches the output behavior that we think we ought to have given the input behavior. But there is the Oracle problem. If we're not real sure about what it ought to do, or what the answers ought to be exactly, it's very hard to validate it that way, and that's part of the requirements capture problem. Since we don't have a complete understanding of the universe that the program is supposed to interact with, the expected results come down to a lot of peoples' best judgement, a lot of discussion and a lot of careful educated guesses at the very top level of this thing. Lower down it's easier.

QUESTION: STEVE HARPER (SAIC): To what extent do the suites of ADA-9X address the wrong language portion of [Dr Royce's] presentation, especially if I believe one is developed for a safety-critical system?

DR. McHUGH: As the co-author of the safety and security annex for ADA-9X, I can give a little bit of background on that. In the early requirements phase for ADA-9X the requirement that the language have a formal semantic definition, which was a requirement in the original ADA requirements and not met by ADA-83, was reiterated. The distinguished reviewer panel, which was basically in control of what got in and what didn't, soundly defeated that on the grounds that it was useless and impossible and that they had done very well without it in the past and they would continue to do very well without it in the future.

There was some work that led to what was called the ADA-9X language precision team which did some formal studies--Dan Craigen was part of that team, I was part of that team--to look at specific issues, and I think that made some very useful contributions to the development of the ADA-9X language.

The concept of annexes came along later. The ground rules for annexes were fairly stringent. They could not change the meaning of anything in the base language. Most of the annexes were added in terms of particular packages, which is not an appropriate mechanism for safety or security where what we're really looking at is assurance. The best thing that has come out in the annex as it stands now is an ability to cause certain features of the language not to have to be supported for some applications. There is a pragma that basically will allow you to say that, "My program isn't going to use tasking, so you don't have to provide any tasking support." Now, exactly how this is going to be validated remains up in the air and the language lawyers are having a field day with some of the things. But the things that are in the current annex are directed more towards providing predictable execution and better assurance for the run-time behavior of ADA-9X programs than they are to new features in the language.

5.3 Specification-Based Testing: What is it? How Can It Be Automated?: Mr. Robert M. Poston

SPECIFICATION-BASED TESTING: WHAT IS IT? HOW CAN IT BE AUTOMATED?

by Robert M. Poston

The Primary Testing Reference

Software testing should begin with a written requirements specification. A specification states how software is expected to behave and describes operational characteristics (performance, reliability, etc.) of the software. A specification serves as a reference or base to test against, giving rise to the name, specification-based testing. Should analysts or designers fail to write a specification, then testers are obliged to write their own specification to test against. Specifications written by testers may be called test plans or test objectives.

For a long time, we in the software industry tried to avoid writing specifications. We wanted to code without taking time to write any description of how software was supposed to behave. Today we realize that we cannot develop software — that will work as expected — without a written specification. Not only do testers need specified-behavior information in order to detect deviations from expectations, but other software practitioners must have that information, too. Designers must know how software is expected to behave when they need to change or repair designs; programmers must understand what programs are supposed to do, so they can write code that performs as expected and so on. In a well-planned software project, specified-behavior information is not left to float around in people's heads. Rather, the information is recorded systematically and made available to all who need it in a requirements specification from the first day of development to the last day of maintenance. The written specification then becomes the definitive reference for software development and testing.

Other Testing References

Some experts believe that software testing should be based on code. Others think it should be based on profiles of the software under test or on statistics about the software. Code-based, profile-based, and statistically-based testing all have advantages. However, each of these testing methods

also has a serious limitation. No one of these methods, by itself, can produce test cases that demonstrate whether or not software works as specified. Using these methods without a specification will lead to inconclusive testing. Only when combined with specification-based testing do these methods prove useful.

Consider code-based testing, for example. What good is it to test code against itself? Code-based testing merely shows that code is exercised. But does the code work the way it is supposed to? Have any functions been omitted, or does the code perform functions that were never intended? Only by referring to a specification will testers know that software works as specified.

Most testing experts advising our industry today, whether advocates of specification-based testing or not, say that a requirements specification is necessary for testing (and of course, for developing) software. John Musa, a leading proponent of software reliability testing, stresses that testing should start with an operational profile. An operational profile is a description of inputs to software under test. The speed, volume, and probabilistic distribution of inputs are defined in the profile. All this information comes from a requirements specification.

William Howden, well-known spokesman for functional testing, says that functional testing depends on a test oracle. An oracle is a person or tool that judges if software passes or fails a given test. To make this judgment the oracle must be familiar with the requirements specification.

Tom McCabe, an early supporter of control flow or complexity-based testing, tells his clients that tests should be derived for the purpose of exercising every statement, branch, and control flow path in the code. McCabe also says that in order to judge whether or not software passed a test, testers must know what the software is expected to do. Testers must have a requirements specification to test against.

The list of testing experts whose techniques depend on requirements specifications goes on and on: Elaine Weyuker championing data flow testing and Richard DeMillo advancing mutation testing, etc. The message here is that no matter what school of testing or software development we follow, we need a specification.

Specification-Based Testing

Since the specification is the definitive reference for software testing, many practitioners are moving to specification-based testing. Increasingly, developers and testers are asking about the specification-based testing process and how to automate each step in the process.

What is the testing process?

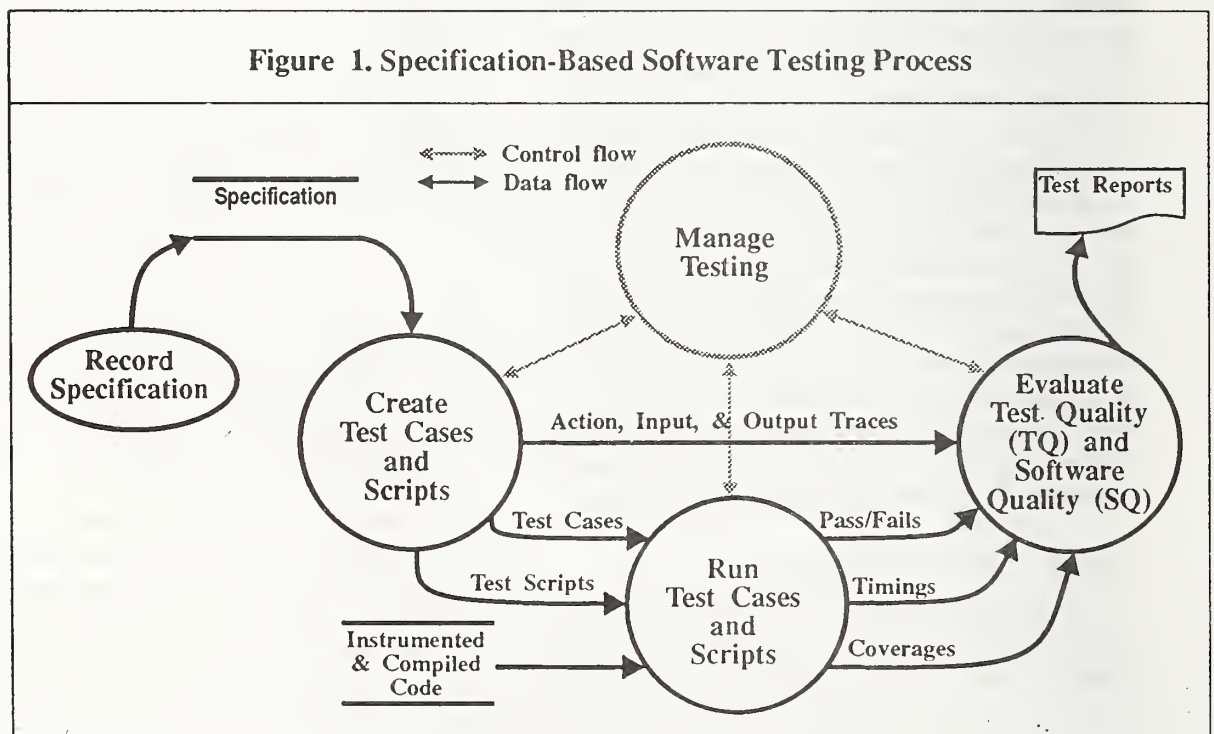
The testing process is easy to remember. Start, as illustrated in Figure 1, with a well-written requirements specification that carefully describes how software is supposed to behave. Create test cases directly from the specification. Run or execute the test cases. Then evaluate the test cases to make sure they have covered all functions, inputs, outputs, and structures of the software under test. Create, execute, evaluate!

How do we get a correct and testable requirements specification?

Correct and testable requirements information is unambiguous, consistent, and complete. A word in a requirements specification is unambiguous if it has one, and only one, definition. A requirements specification is said to be consistent if each of its words is used in one, and only one, way. Consider for example, the word "report." In a requirements specification "report" must be used either as a noun or a verb. To use "report" as both a name and an action would make the specification inconsistent. Completeness from a tester's point of view means that requirements contain necessary and sufficient information for testing. Every statement must have a defined input, function, and output.

We once faced two big jobs when developing a requirements specification. First, we had to figure out what the requirements were during a problem or needs analysis. Then we had to record correct requirements in a way understandable to other people or tools. We had to know what to write about and then we had to write carefully what we knew: both difficult tasks.

Today we can cut the work of recording requirements dramatically through the use of auto-



mated tools. But we still need to know what to record. If we do not understand what a software product is supposed to do, no tool can provide that understanding for us. Ill-conceived requirements are still the number one cause of unfinished or seldom-used software products, and that problem cannot be automated away.

On a brighter note, as soon we have an idea of what our requirements will be, tools can help us organize our ideas and prevent gaps in our thinking. As we record requirements information in tools, they can verify correctness and testability. Therefore, people who use automated tools during requirements specification are more likely to produce correct, testable requirements than those who develop requirements manually.

(Note: From here on, various kinds of automated tools will be discussed. The superscript, ^{TD}, that appears after certain tool names or types indicates that more information about these tools is listed in the Tool Directory at the end of this paper.)

How do we record requirements with tools?

Requirements information can be captured by tools in one of two ways: textually or graphically. We can write the information in sentences and paragraphs, or we can draw the information in pictures and diagrams. In either the textual or graphical method, we can record information informally without rules or formally according to rules. Discussed next are tools that enable analysts to capture requirements information textually or graphically, with or without rules.

Recording textual requirements informally

The most common computerized requirements recorders are the text editor and word processor. In workplaces where people use text editors or word processors to write specifications, requirements usually are recorded in a natural language such as English. The only rules applied to the writing are the syntactic or grammar-and-punctuation rules of the natural language; semantic rules that restrict meanings and use of meanings, such as a word may have only one meaning or a word may be used in only way, are not applied. So we say the natural language used in the specification as well as the

specification itself are informal, because neither adheres to semantic rules.

Informal specifications may appeal to developers and end users who are not familiar with formal languages. If the natural language of developers and users is English, they will need no training to read an informal specification written in English. On the other hand, these people may have difficulty reading a formal-language specification without training.

Unhappily, the benefit of easy-to-read informal specifications is outweighed by a major disadvantage. An informal specification does not incorporate semantic rules, yet automated verifiers as well as tools such as design, code, and test generators depend on semantic rules to parse and verify information. The information in an informal specification, then, cannot be checked by tools for correctness. Neither can that information be processed automatically for output to other tools. Informal specifications condemn us to manual verification, testing, and development.

Recording textual requirements formally

We need not reject text editors and word processors for specification writing, however. We can use those tools successfully for capturing specifications in English, if we apply a few simple semantic rules to our writing. Once we write the specification in rule-restricted English, we can employ a tool called a language verifier ^{TD} to check that the content of the specification conforms to the rules. By applying semantic rules to specification writing, we begin to formalize the specification, making it ever more tool usable.

An easy way to apply semantic rules to our writing is to use a formal, textual specification language called the Semantic Transfer Language (STL) that appears in IEEE Standard 1175, 1991.¹ The STL is immediately familiar and comfortable to anyone who reads English. A sample STL sentence appears in Figure 2. The STL's greatest advantage lies in its expressive power, but we also notice right away how simple and efficient the STL syntax is. An STL sentence begins with a capitalized keyword and ends with a period. Action is the keyword in the sentence in Figure 2. STL sentences may contain any number of clauses in any order. Clauses start with a keyphrase and end with a semi-

colon. Keyphrases are italicized for easy identification in Figure 2. Commas separate items or words in a list as seen in the "uses" clause. These are the only syntactic rules specifiers need to know to write STL sentences.

Figure 2. Sample STL Sentence	
Action	A01
<i>is allowed in state</i>	normal ;
<i>receives event</i> item	warning_interrupt ;
<i>transmits event</i>	safety_action1 ;
<i>uses data</i> item	water_temperature_reading, water_temperature_base ;
<i>produces data</i> item	safety_action_report ;
<i>has duration time maximum</i> 3 ;	
<i>has duration time unit</i>	"second".

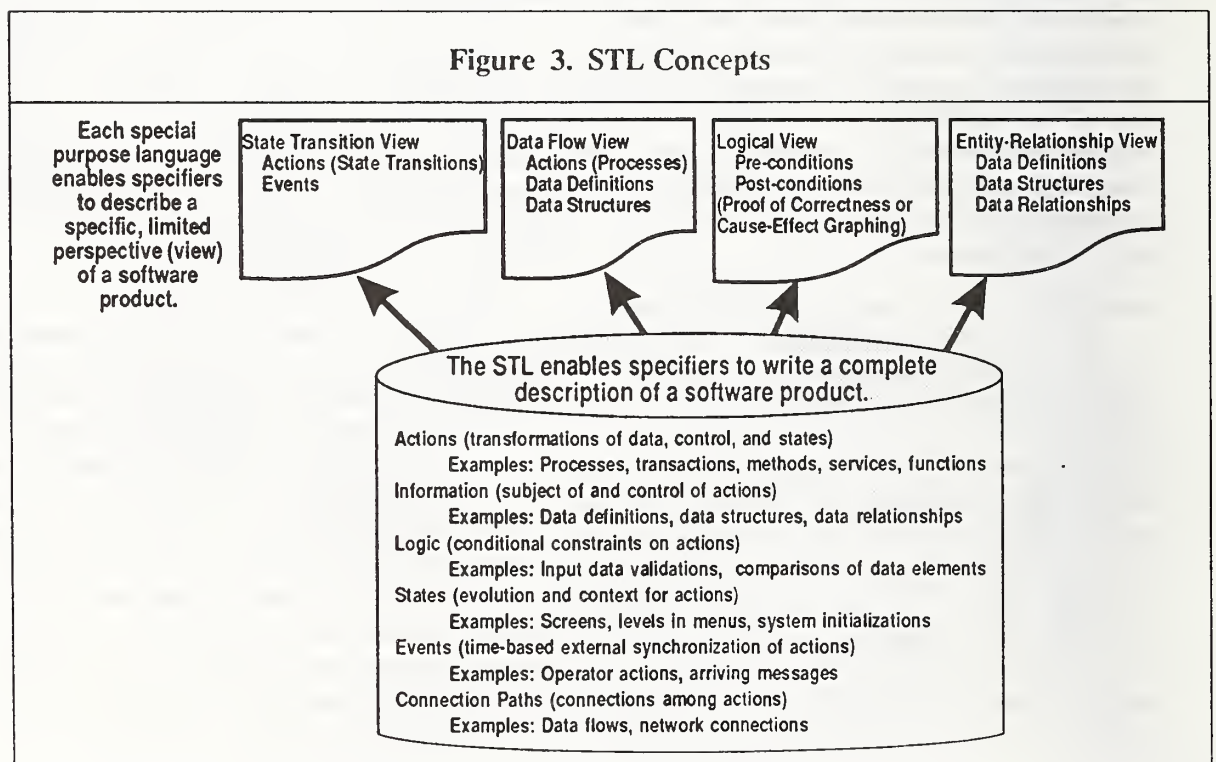
Whereas syntactic rules concern the form and structure of a language, we know that semantic rules bear on the meanings a language conveys. Vocabulary is the vehicle through which meanings are

expressed. Ideally for software developers and testers, a formal language would have a large enough vocabulary for specifiers to say anything necessary about software behavior. Then developers and testers could draw on a formal-language specification for complete behavioral information. But tool developers cannot build big enough scanners and parsers (yet) to accommodate unlimited vocabularies. Therefore, formal languages processed by tools are restricted in vocabulary size.

With vocabulary limitations in mind, the creators of the STL designed a tool-manageable language with a large enough vocabulary to enable requirements writers to describe software behavior thoroughly. As shown in Figure 3, writers can describe actions, information (which includes data and relationships among data), logic, states, events, and connection paths with the STL. Other formal languages are available in our industry, but none is as well-suited to automated, specification-based development and testing as the STL.

Recording graphical requirements informally

Bubbles and lines started to fill thousands of CASE-tool (Computer-Aided Software Engineer-



ing-tool) screens in the mid-1980s, when many of us first realized we could tool draw attractive pictures of our specifications, instead of writing them in boring old text. Graphical specification was a fresh, fun-to-practice idea, but as with many things new, we had to work out some kinks in our implementation of the idea.

In their early days of inexperience, most CASE tool users recorded graphical specifications informally. With little regard for formal technique, they started to draw data flow diagrams. The diagrams contained abstract or logical representations but no explicit descriptions of software behavior or operation. With data flow diagrams, software practitioners could see processes and data flow clearly as well connectivity among processes. But it was difficult to build or test software (that would work as expected) from these diagrammed specifications, because they did not contain detailed behavioral or operational information. Specifically, the diagrams did not contain data definitions that provide information about structure, relationships, and values of data. Neither did the diagrams contain the condition (logic), event, timing, state, performance, or error handling information that software developers need. CASE tool users and developers had to stop and rethink where they were going with graphical specification.

Recording graphical requirements formally

Tool users and developers have arrived now at a better understanding of how to specify requirements graphically. To gain this understanding, users and developers first had to re-examine the data flow diagram. In the data flow diagram, a process could be represented with a circle, a unit of data in motion with an arrow, and data retained over time with a pair of lines. The circle, arrow, and line-pair symbols (i.e., the graphical vocabulary), and what these symbols could represent, comprised a simple graphical language called the data flow language. The language could be supplemented with process specifications and data dictionaries. But the data flow language, even when augmented with process specifications and data dictionaries, was so vocabulary-limited that it permitted only partial specification of software behavior. Other graphical

languages with greater expressive power were needed.

By the late 1980s the software industry was inundated with popularized graphical languages — each targeted at a specification niche. For example, the decision tree was offered as a diagram language to help specifiers express logic. The state transition and event trace diagram gave specifiers a way to record information about context (i.e., states) and time (i.e., events). Entity relationship and entity history diagrams were niche graphical languages good for specifying relationships among units of data.

Unfortunately, each niche language was created in isolation from the others with no common semantic rules to govern meanings among the languages. The result was compartmentalized languages in which a piece of data could mean one thing in one kind of diagram such as a decision tree and something entirely different in another kind such as a state transition or entity relationship diagram. Sharing information across niche languages was difficult.

In the 1990s, another crop of graphical languages came to industry attention. This time they were the object languages: Object-Oriented Analysis; Object-Oriented Structure Design; Object-Modeling Technique, etc. These languages are composites of the niche languages. Although semantic rules do not prevail among the composite languages, each composite, usually containing two to five niche languages, is fairly self-complete and is overlaid by a set of semantic rules. These rules make the composite, object languages quite formal, verifiable, and tool usable.

In light of tool-processable object languages, many specifiers are looking at graphical specification anew. Specifiers now can create a testable, graphical specification with a CASE tool that utilizes an object language.^{TD} They can draw pictures of software to be built and supplement the pictures with textual descriptions according to the semantic rules of an object language. Once specifiers have recorded requirements in a tool, they can direct the tool to produce a specification in either a text-and-picture hard copy ready for distribution and review or a text-only file suitable for processing by tools. At least two CASE tools offer push-button test case

generation directly from processed specification information.^{TD}

How are test cases created from a specification?

Until the advent of powerful information-capturing tools in the late 1980s, most professionals tested software after it was defined, designed, and coded. They exercised a more or less finished product to see if its major parts would work. When people accustomed to this test-after-coding approach first come to specification-based testing, they may be confused by the idea that testing should start by creating test cases from a specification at the front end of software development before a product is designed. Indeed, specification-based testing changes the way we view the software life cycle and how the testing process fits into the cycle.

Test execution (i.e., running test cases) and test evaluation still have to begin after software is developed. However, by shifting the creative work of planning, designing, and preparing for testing to the front end of the software development life cycle, test cases can be developed as software is developed. (See Figure 4.) The parallel activities usually

enable the project manager to cut time from the development schedule. Yet, the tester finds more time to design high quality test cases, because testing activities were started early.

This front-end testing approach does not work without a test case generation tool^{TD}, however. In the days before this tool was available, if we had developed test cases against requirements too soon in the development life cycle, requirements would have changed (as they invariably do) and rendered the test cases invalid. All the work that went into designing, generating, documenting, and tracing test cases probably would have been lost with the change of a requirement. Either we would have had to discard the cases and start from scratch to develop them again in accordance with the changed requirement, or we would have started a painstaking effort to try to modify the test cases.

Then came the automated test case generator. (A test case suitable for automatic processing is described in Figure 5.) The test case generator takes formally-recorded specification information, treats it as though it were a knowledge base or data base, and applies test design rules to this base to create test cases automatically. A test case generator is often compared to a compiler. Much as a compiler

Figure 4. New Schedules for Software Testing

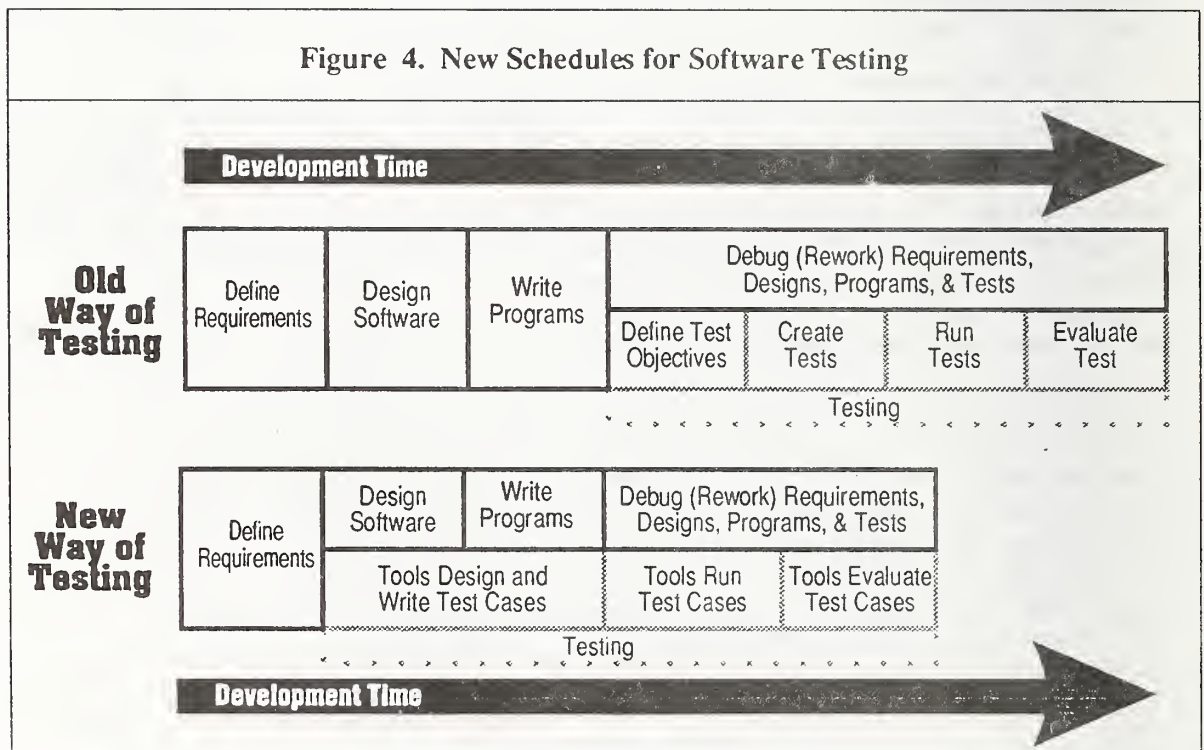
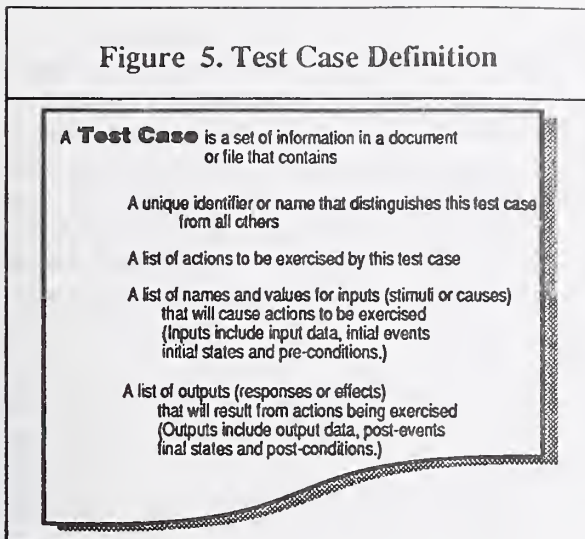


Figure 5. Test Case Definition



reads and analyzes source code, a test case generator reads and analyzes specification information. A compiler uses algorithms (i.e., rules) to produce object code; a test case generator uses rules to create test cases. Functional testing, boundary value analysis, cause-effect graphing, and state- and event-directed testing are among the groups of rules or test design techniques that a test case generator incorporates. Complex manipulations of data go on inside a test case generator, then, but the user sees very little of such goings-on. The user just inputs specification information, pushes a couple of keys, and awaits an output of test cases.

Automation makes large productivity and quality gains possible during test case generation. Industry statistician, Capers Jones, reports in his 1991 book titled *Applied Software Measurement* that software testers in the United States produce an average of 30 to 300 test cases per staff month.² And, as Jones points out in his book, the quality of manually-developed test cases is questionable. (Are such test cases complete? Are they redundant? Are they even worth running?) In contrast, a good test case generator will produce 300, very reliable test cases in less than 30 seconds without any human intervention.^{TD}

Finding errors early has long been considered a good idea in the software development community. Specification-based test case generators have taken that idea one step further into error prevention. If people know how a product will be tested before they start to build the product, they usually

will create a product that will pass its test cases. With automated generators and front-end development, reliable test cases can be available as soon as information is specified. So from the outset of their work, designers can use those cases as references against which to develop correct, specification-consistent designs free of most probable errors. Designers can then deliver their high-quality designs to programmers who can practice error prevention during coding.

How should test cases be run?

To run or execute test cases, we invoke software under test and apply inputs (documented in test cases) to the software. When the software processes the inputs, it will produce outputs known as actual outputs. We compare the actual outputs to expected outputs to see if software passes or fails its test cases. We may expect to see certain outputs, because we have seen such results when similar software ran in the past. We may use a mathematical calculation or a scientific method to predict outputs the software will produce. When actual and expected outputs from one test case are equal or are within allowed tolerances, we say software passes that test case. When actual and expected outputs from one test case are not equal or close to each other, the software fails to pass that test case.

Test execution is an important activity that lends itself to automation. Automated test execution is particularly useful for running old tests on new or changed code to see if new or changed code affects old code (regression testing).

The following scenario shows how testers benefit by moving from manual to automated test execution. Suppose a tester needs to test a function in a data base containing a mailing list. For the function to delete an entry, data must be available to delete. The tester normally prepares to test the delete function by adding a particular name such as David Jones to the data base. By entering the name, David Jones, into the data base, the tester has set up the initial data state for testing the delete function. The tester has changed the data base from its original or reference data state. Then the tester must set up the function state for "delete" by getting to the screen where the delete function is accessible. This setup or initialization effort is necessary

before a test case for the delete function can be run.

Now the tester is ready to exercise a test case for the delete function. The tester executes the delete function by pressing a key. The name, David Jones, disappears from the screen. The test case has been exercised.

However, the tester should have questions at this point. Did the delete function remove the name, David Jones, from the screen but not from the data base? Did the delete function erase all Joneses from the data base? Did it delete the entire data base? Did deleting David Jones interfere with other names in the data base? To answer these questions, the tester must clean up.

Cleanup in test execution involves two activities: checking side effects or confirming results and returning the software under test to its original state. When testers check for side effects, they find answers to the questions just asked. By querying the data base for the name, David Jones, the tester can confirm that David Jones is no longer present in the data base. By comparing (usually with a comparator program) the data base as it existed before test execution to the data base after execution, the tester can detect any unintentional changes in the data base.

After checking for side effects, the tester returns the data base to its reference or original data state. Most testers return to the reference data state by reversing any changes made during test execution. Had the tester found, for example, that all the Joneses were deleted during test execution, the tester would have to re-enter all Joneses that were in the original data base. Another way to return to the reference data state is to restore the data base by copying it from off-line to on-line storage.

Manual execution of one test case has just been described in three steps: setup, exercise, and cleanup. Some testers exercise test cases in a prescribed sequence so that one test case sets up for the next one or cleans up after the previous one. Running test cases in a planned sequence minimizes execution work but does not eliminate it.

Any way we approach it, manual test execution is hard work. It also is expensive, inefficient, and unnecessary. Most commercially-available test execution tools, commonly called capture/replay or re-

gression testing tools ^{TD}, will record setup, execution, and cleanup commands in scripts for automatic replay. Working from scripts, automated test execution tools can run thousands of test cases without intervention from the tester. Even better, outputs from test case generators can now be executed automatically by capture/replay tools. In one button push we can get integrated tools to create test cases and run them for us while we have a cup of coffee. ^{TD}

How do we evaluate test cases?

Once we run test cases, we must find out if all the test cases working together did an adequate job of testing. This brings us to the final activity in the testing process labeled in Figure 1 as "Evaluate Test Quality and Software Quality."

Evaluating one test case is easy. Software either passes or fails the test case. But how do we know that the test cases we ran satisfied the purpose of specification-based testing? Did the test cases demonstrate requirements, find failures, and exercise code?

To measure how well test cases demonstrate requirements, testers often complete four actions. First, they count how many requirements must be demonstrated. This count may be labeled **TR** for total requirements to be demonstrated. Next, testers trace each requirement to all test cases which exercise that requirement. Then they count all test cases that software has processed correctly (i.e., passed test cases). When all test cases which exercise one requirement pass, the requirement passes. The count of all passed requirements is called **PR**. Finally, to report how well requirements were covered, testers may calculate a requirements coverage factor by dividing **PR** by **TR**. This four-step procedure yields an objective measurement of how thoroughly requirements are demonstrated.

A little more work is needed to measure how well test cases find failures. While most testers count failures, many do not know how many failures were in software before testing began. When testers do not know how many failures were present to begin with, they cannot make an accurate assessment of how well they have probed for failures. Neither do most testers know how many failures were left in software to be passed on to

customers. If testers cannot predict how many failures customers will find, testers cannot evaluate test quality in terms of failure coverage. To assess accurately how well test cases cover failures, testers are beginning to apply software reliability theory. John Musa and co-authors, Anthony Iannino and Kazuhira Okumoto, enlighten testers about this theory in their book called *Software Reliability*.³

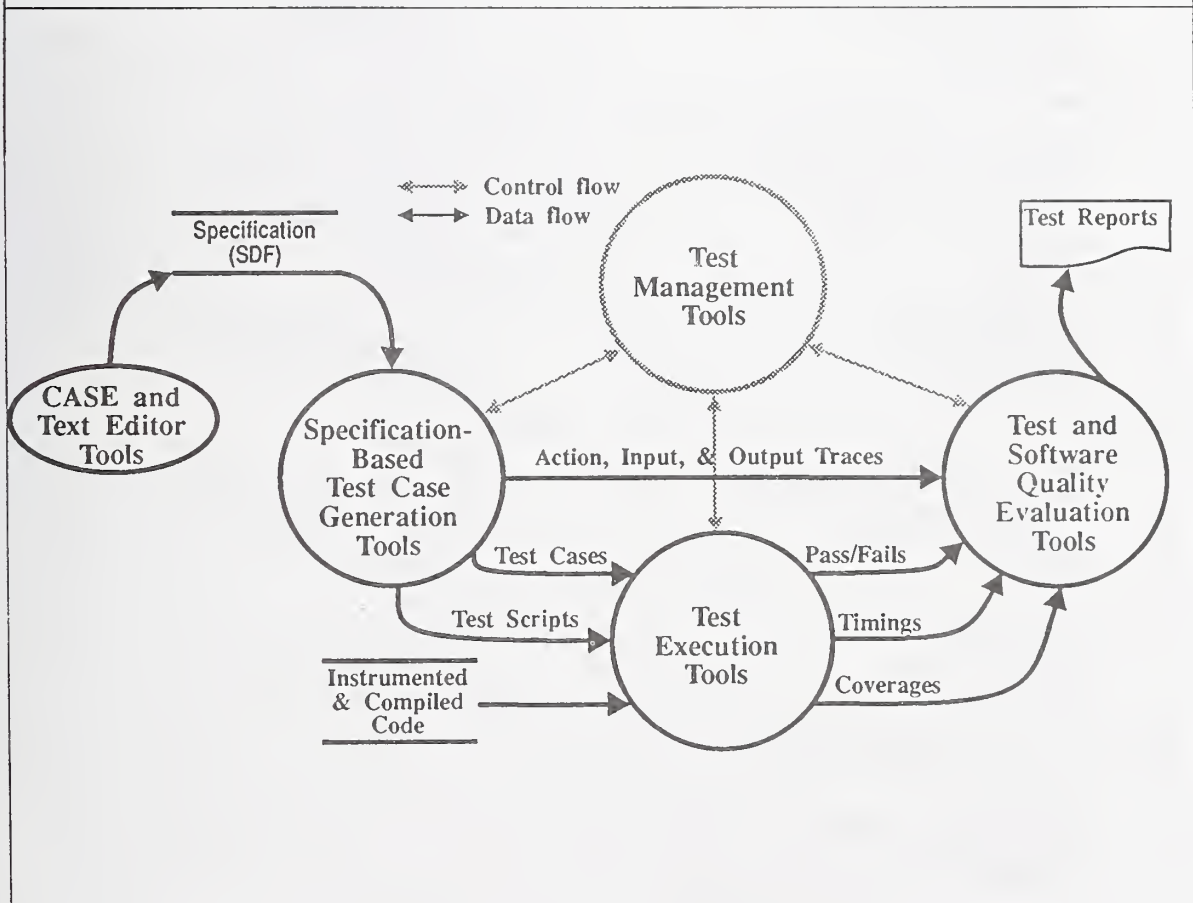
The last type of test quality evaluation that many testers perform is called code coverage measurement. In this effort testers use tools called instrumentors^{TD} and dynamic analyzers^{TD} to measure which statements, branches, and paths in the code were and were not exercised by test cases. Any code not exercised by specification-based test cases is called extra code.

To help us evaluate testing, our industry offers three important automated tools: Tracers to help

measure requirements coverage;^{TD} Reliability Predictors to help measure and predict failure coverage;^{TD} and Analyzers to help measure code coverage.^{TD} These tools are for sale from commercial suppliers. Some are offered free of charge from selected vendors and educational institutions.

In Figure 1 the activities in the specification-based testing process are shown without tool support. Illustrated in Figure 6 is the same testing process improved by automation. A separate tool is available to back every testing activity shown in Figure 6. Our industry's job for the future is to make each tool perform its function more effectively and to make all these tools work together more efficiently throughout specification-based testing.

Figure 6. Automated Specification-Based Testing



Tool Directory

Following are examples of particular types or categories of tools mentioned in this paper. In most categories there are other similar tools available in the software industry.

Language Verifiers

1. T
Interactive Development Environments
595 Market Street, 10th Floor
San Francisco, CA 94105
800-888-4331
2. SoftTest
Bender & Associates
P.O. Box 849
Larkspur, CA 04939
415-924-9196

CASE Tools Utilizing Object Languages

1. Software through Pictures (StP)
Interactive Development Environments
595 Market Street, 10th Floor
San Francisco, CA 94105
800-888-4331
2. Excelerator
INTERSOLV
3200 Tower Oaks Blvd.
Rockville, MD 20852
301-230-3200

CASE Tools with Test Case Generation Capabilities

1. Software through Pictures (StP)
Interactive Development Environments
595 Market Street, 10th Floor
San Francisco, CA 94105
800-888-4331
2. Teamwork
Cadre Technologies
222 Richmond Street
Providence, RI 02903
401-351-5950

Test Case Generators

1. T
Interactive Development Environments
595 Market Street, 10th Floor
San Francisco, CA 94105
800-888-4331
2. SoftTest
Bender & Associates
P.O. Box 849
Larkspur, CA 94939
415-924-9196

Capture/Replay or Regression Testing Tools

1. AutoTester
AutoTester Inc.
8150 North Central Expressway
Dallas, TX 75206
214-363-6181

2. QA Robot
Software Quality Automation
1 Parker Street
Lawrence, MA 01843
800-228-9922

Capture/Replay Tools Integrated with Test Generators

1. XRunner
Mercury Interactive
3333 Octavius Drive
Santa Clara, CA 95054
408-982-0100
2. CAPBAK
Software Research Inc.
625 Third Street
San Francisco, CA 94107
800-942-7638

Requirements Coverage Tracers

1. RTM
Interactive Development Environments
595 Market Street, 10th Floor
San Francisco, CA 94105
800-888-4331
2. Teamwork/RqT
Cadre Technologies
222 Richmond Street
Providence, RI 02903
401-351-5950

Reliability Prediction Tools

1. SQA Manager
Software Quality Automation
1 Parker Street
Lawrence, MA 01843
508-689-0182
2. RELIA
Free Software Foundation, Inc.
675 Massachusetts Avenue
Cambridge, MA 02139
617-876-3296

Code Coverage Tools

1. Tcov
Free Software Foundation, Inc.
675 Massachusetts Avenue
Cambridge, MA 02139
617-876-3296
2. TCAT-PATH
Software Research Inc.
625 Third Street
San Francisco, CA 94107
800-942-7638

References

- ¹IEEE Std 1175, 1991, Trial-Use Standard Reference Model for Computing System Tool Interconnections, IEEE Press, New York, NY, 1992.
- ²Capers Jones, Applied Software Measurement: Assuring Productivity and Quality, McGraw-Hill, Inc., New York, NY, 1991.
- ³J.D. Musa, A. Ianino and K. Okumoto, Software Reliability: Measurement, Prediction, Application, McGraw-Hill, Inc., New York, NY, 1987.

About the Author

Robert M. Poston became president of the PEI Division of Interactive Development Environments in April 1993 after serving as president of Programming Environments, Inc. from 1981 to 1993. Prior to his presidencies, he held technical and managerial positions in the computer industry for 15 years. He is well-known as the originator of the software testing tool called T. Mr. Poston has been prominent in IEEE activities since 1970. From 1984 to 1987 he served on the editorial staff of IEEE Software magazine. Recently, he headed the standards effort to develop IEEE Std. 1175. Mr. Poston is the recipient of numerous professional and industry achievement awards and is recognized around the world for his lectures and seminars on software engineering subjects.

5.3.1 Questions: Mr. Robert M. Poston

QUESTION: A.L. SUDDUTH (Duke Power): Could a logic diagram in standard format linking blocks of elements from a standard library be considered a software specification? If not, what additional features would be required of a standard logic diagram in order to make it a specification? Can we take advantage of domain-specific knowledge in producing a formal specification methodology, or must formal specifications be generic to the domains?

MR. POSTON: Yes! Okay, "Could a logic diagram in standard format linking blocks of elements from a standard library be considered a software specification?"

If that logic diagram were expressed in a formal syntax the answer is yes, because that boils down to a subset of predicate calculus and that boils down to basic decision tables. So, it would have to be formatted rigorously, but then it could be considered a specification. Decision trees work. They're not the best, but decision trees do work.

"If not, what additional features could be required of a standardized logic diagram in order to make it a specification?"

First, express it in a formal language, or translate it. Hopefully your decision tables are in trees, or your logic diagrams are in magnetic form. If they're not, then this is wrong. If they are, reformat them into a table format, put them into a decision table, and hopefully add a little more. You know, those upside-down A's and those backward L's, they're really not that dangerous, particularly for people that deal with logic diagrams. You're only adding a few constructs. So, move them to a formal language.

"Could we take advantage of domain-specific knowledge?"

Well, yes. Domain-specific knowledge usually comes out in how the logical equations are expressed. You express things like temperature range (e.g., in this area, below this, above that, that's a safe area) and you've already put domain-specific knowledge into that decision tree. Whether or not you could apply domain-specific knowledge using an AI tool to help you express that knowledge, that's a completely different subject. It's a much more powerful subject. And there is a lot of research going on in that area.

QUESTION: GREG MILLER (EG&G Falcho): You're using software to generate specifications, create test cases and so on. How do you justify the adequate reliability of this host of tools? Must you also validate the tools on a case-by-case basis and, is this done manually?

MR. POSTON: I am a tool vendor. I will sell you a tool if you're not careful. Given that as background, never trust the tool. Always check it.

QUESTION: WILLIAM D. GHRIST (Westinghouse Electric): If we could produce a tool that would take a specification and produce test cases that guarantee that the code conforms to the spec, shouldn't we be able to produce a tool that would generate code that is guaranteed to conform to the spec?

MR. POSTON: No. They're nowhere near the same. Code generation has got a thousand and two trade-offs that test cases don't. Well, take a simple sort routine. How fast is it? How much memory does it take up? Test cases don't have that problem. Generating test cases is infinitely easier than generating code for the general purpose applications. They are nowhere near the same.

5.4 Applicability of Object-Oriented Design Methods and C++ to Safety-critical Systems: Dr. Barbara B. Cuthill

Applicability of Object-Oriented Design Methods and C++ to Safety-critical Systems

Barbara B. Cuthill
National Institute of Standards and Technology

September, 1993

Abstract

This paper reports on a study identifying risks and benefits of using a software development methodology containing object-oriented design (OOD) techniques and using C++ as a programming language relative to selected features of safety-critical systems development. These features are modularity, functional diversity, removing ambiguous code, traceability, and real-time performance.

1 Introduction

This paper reports the results of a preliminary study identifying risks and benefits of applying object-oriented design (OOD) and C++ relative to selected key features of safety-critical systems development. Safety-critical systems are typically real-time systems for which the failure to operate correctly could result in loss of life or substantial property damage. The features important to the design, implementation and testing of such systems include modularity, functional diversity, removing ambiguous code, traceability, and real-time performance. OOD and C++ can provide benefits to developing systems with these features but can also introduce new risks to incorporating these features if not used carefully.

The IEEE's *Glossary of Software Engineering Terminology* [54] defines *object-oriented design* as "a software development technique in which a system or component is expressed in terms of objects and connections between those objects." It defines an *object* as "an encapsulation of data and services that manipulate that data" and *encapsulation* as the "technique of isolating a system function within a module and providing a precise specification for the module" [54]. OOD is ultimately a simple but powerful paradigm of creating types for objects or *classes* consisting of narrow well-defined interfaces to blocks of data structures and functions operating on those data structures. These interfaces limit access to the data and functions of a class and define its external behavior.

OOD should improve the readability, quality and maintainability of software. A domain expert should find OO designs and programs easier to understand because OOD can incorporate domain models, such as those developed as part of a domain analysis, into the design. For example,

OOD can model real-world objects such as sensors and hardware controllers as separate software components with defined behavior. OOD should improve the quality of software because developers can reuse encapsulated portions of the analysis, design or tested code in new systems. Since the interface to an object should be the only means of accessing the object, module interfaces should be simpler making component integration easier. Easier component integration allows for the easier extension and maintenance of an OO system by limiting module interaction.

OOD is only one part of the software development life cycle. Without programming language support for OOD, programmers will have difficulty maintaining the OOD restrictions on system design. There are several general purpose languages with OO features (i.e., C++, ADA) and special purpose OO languages (i.e., SMALLTALK, CLOS, Eiffel). This study focuses on C++ because it is an extension of a language popular for safety-critical systems development. C++ [4, 5] extends C with constructs for the creation and manipulation of objects and classes.

2 Key features of Object-Oriented Design and C++

2.1 Background in Object-Oriented Development

The design phase of the software development life cycle starts with products of the requirements analysis for the target system and develops a design through several transformations from a sketchy design meeting the requirements into an easily codeable form. OOD, if part of an overall OO development methodology, begins with the results of an object-oriented analysis (OOA) of the system requirements and, after several refinements, provides the design for implementing the system as an object-oriented program (OOP). (See Figure 1) OOA, OOD and OOP should flow smoothly. A good analysis can become the initial design. Similarly, with programming language support, the final design should be trivial to code. With compatible methods, tools, languages and notations, the process of going from analysis through design to code should not have boundaries. Unfortunately, the actual transitions are rarely smooth since there are few OO methodologies supporting more than a small part of the lifecycle [27].

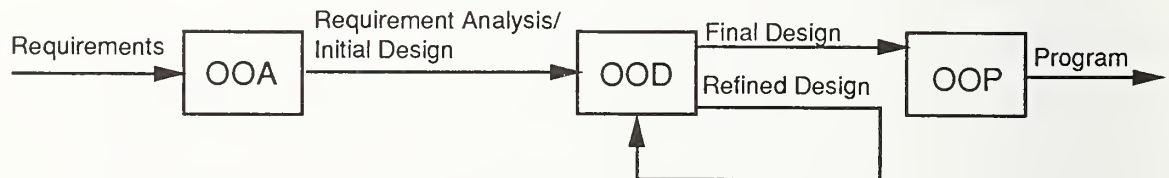


Figure 1: Connections among OOA, OOD, and OOP.

De Champeaux et al. [28] created a template for OOD methods describing the most common elements of those methods. De Champeaux's template decomposes OOD into three interleaved *phases*, a class design phase, a system design phase and a program design phase. The class design phase is a bottom up process of defining the system's objects and classes, describing their internal and external behavior and composing complex objects from simple objects. Objects and classes may mimic real-world objects (e.g., hardware controllers) or represent and manipulate

collections of data in the domain (e.g., lists, queues). The system design phase is a top-down process of defining partitions of the system according to control information supplied by the requirements, and needed class interactions. This phase also defines the resource management facilities required. The program design phase applies performance and resource constraints to the design. The program design streamlines resource use and communication requirements and replaces poorly performing components. Replacement can include combining classes or partitions or revising the operations required in a class or partition. Figure 2 illustrates these design phases and how each effects the design. While an OOD method should contain all three phases, each method interleaves these phases differently, even interleaving the design phase with parts of the OOA or OOP phases.

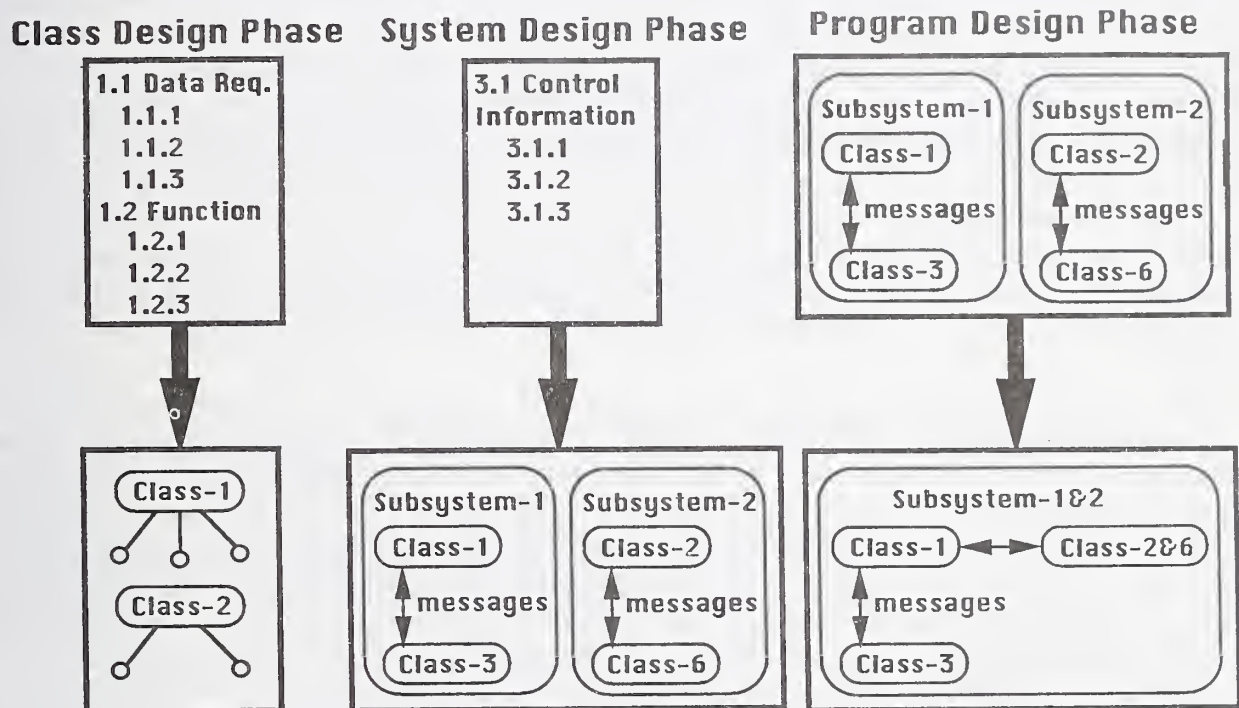


Figure 2: Phases of Object-Oriented Design.

Good OOD interleaves the design phases to develop pieces of the design in a bottom-up fashion using knowledge of the overall requirements to put those pieces together into subsystems or the final system. OOD provides the opportunity for a structured mix of bottom-up and top-down methods. This combination better captures the information in typical requirements documents which mix abstract and detailed information. Lubars, Potts and Richter [63] present a case-study using OOD methods for designing the mission planning subsystem of the Tomahawk Weapons Control System [TWCS]. This exercise applied OOA and OOD methods to a system with a large set of requirements (approx. 500 pages). The study used bottom up techniques to develop the individual classes from the data and low-level requirements and top-down techniques to partition the system and organize the classes into subsystems from the high-level and control requirements.

This study illustrated the usefulness of OOD for combining the two techniques in a large real-time system with complex requirements.

2.2 Current Limitations on the use of OOD and C++

While OOD and C++ can enhance software development, they are not cost free and cannot enforce or replace good software development practices. Developers will not use any difficult methodology consistently unless they see advantages to using and receive support for doing so. Because generating a good, consistent OOD is difficult, a developer could ignore or deliberately bypass OOD and C++ constraints. As with any substantial technological advance, organizational adoption of OOD requires substantial investment in training in the chosen methodology and the tools to use that methodology. Developers with experience in starting new object-oriented projects strongly endorse extensive training in OO methodology as well as in specific OO languages and tools [1, 2, 45, 71]. Software tools for generating and browsing through classes substantially aid the OOD process, but selecting a set of compatible tools appropriate to an OOD method and learning to use the tools well requires considerable time and effort. One expert in the area recommends six months of immersion in OOD training [2]. As with any adoption of new technology, management must commit to the change and be willing to accept failure on the first projects using the methodology [30, 70]. In addition, OOD does not reduce the need for careful design reviews and a quality assurance program to enforce the use of good development practices.

One potential problem in moving an organization to using OO development methods is the proliferation of such methods, notations and languages. Many OO methods provide only sketchy explanations of how to use the method and none cover the full lifecycle [27]. Consequently, each OOD method has its own notation and object model providing slightly different characteristics and default operations for objects. These differences may become very important especially when developers need to change or compare models or notations [73]. Many surveys of object models and OOD methods are available [e.g., 26, 27, 28, 29, 30, 31]. A study submitted to the International Standards Organization (ISO) [56] surveys the OO notations. This paper focuses on selected fundamental features of OOD but does not provide details about any particular OOD method.

Like OOD methods, there is no standard C++ yet. Consequently each C++ compiler accepts a slightly different version of the language; however, the variations are minor with most developers accepting Stroustrup's definition [4]. Unfortunately, many C++ compilers are actually preprocessors producing C code giving a C programmer the opportunity to bypass the C++ language constraints completely. This study describes C++ features which are broadly agreed on and does not discuss the details of what one compiler will accept and another will not. This study also examines only the C++ code and does not consider the C or assembly code produced during compilation.

This study examines key features of OOD and C++ on which there is broad agreement. The OOD features are common to almost all OOD methods. Most C++ compilers implement the C++

features; however, the C++ features are specific to C++ and do not generalize to other OO languages.

2.3 Key Features of Object-Oriented Design

Most OOD methods support the creation of an OOD consisting of the definition of groups of class hierarchies and objects and controlling software for manipulating the objects using the provided interfaces [e.g., 2, 9, 11, 12, 20, 26, 28, 29, 31]. The classes, objects and controlling software should be traceable to the requirements and specific elements of the code. There is general agreement that the following features define significant characteristics which as a group differentiate OOD from non-OOD methods:

Encapsulation - A mechanism for limiting the accessibility of a data structure or group of data structures to a set of operations manipulating the data and a set of operations providing a well-defined interface to that data. The data and the operations accessing the data are treated as a single unit.

Abstraction - A mechanism for recognizing and representing the underlying similarity of objects and operations in a given domain. Class and hierarchy definitions typically provide data abstraction in an OOD.

Inheritance - A mechanism for deriving new specialized definitions from existing, general definitions by incorporating the general definition and extending or adding detail to it.

Refinement - The process of adding detail to an object-oriented description of a system without altering the basic form of the description.

Polymorphism - This is the ability to reference instances of more than one class via a single function call or operation. A base class and its derived classes can all provide different definitions of a single function. A function call using a parameter of the base class could apply any of the derived functions depending on the exact type of the parameter.

While the details of each OOD method are slightly different and each implements these features differently, most OOD methods encourage the production of designs with all of these features.

2.4 Key features of C++

Stroustrup [4, 5] designed C++ to extend C to support the key features of OOD and OOP. While there are a number of languages explicitly supporting OO features (e.g., SMALLTALK), C++ has the advantages of general acceptance and compatibility with C for relative ease of learning and use. While an OOD can theoretically be implemented in any language (e.g., X windows toolkit, an OOP, is implemented in C), consistently maintaining OO features with no

programming language support requires disciplined programmers to maintain OO restrictions themselves. It also requires that programmers duplicate or reimplement features that the language could provide. The important extensions provided in C++ to support OOD and OOP follow:

Classes - These are named aggregates of data elements and operations (termed methods). Classes define types of objects.

In-Line Functions - The compiler expands these functions at compile time directly inserting them into the body of the code. This process is very similar to macro expansion; however, depending on the implementation, the compiler may generate short, uncomplicated functions in-line automatically.

Templates - These are class or function definitions which the developer can customize for parameters of specific data types.

Call by reference - Functions can contain reference parameters creating implicit pointers to the calling arguments avoiding the explicit passing or explicit selection of variable's addresses. While many other programming languages have call by reference (e.g., FORTRAN, PASCAL), C does not possess this important feature.

Type-Safe Linkages (or strong typing) - The arguments of a function call must match, directly or through inheritance, the number and type of parameters in the function definition unless the function has specified optional parameters with default values.

Classes are the central mechanism for encapsulating, abstracting and inheriting information and for defining types of objects. Support for classes is an important feature of OOD, OOP and C++. C++ provides a number of options for defining access to the elements of classes, manipulating classes and for supporting class inheritance. C++ features for class definition follow:

Public elements or methods - This is a definition of a data structure or function within a class which is accessible by any routine inside or outside the class. Public elements and methods define the interface for the class.

Private elements or methods - This is a definition of a data structure or function within a class which is only accessible from routines defined within the class or to specifically declared *friend* classes and functions.

Protected elements or methods - This is a definition of a data structure or function within a class which is accessible only to members of the class and to members of any classes derived from that class.

Friend functions or classes - Friend functions and classes can access protected and private data and methods. A class definition can include friend declarations.

Derived Classes - These classes inherit from other classes.

Base Classes - These classes supply data and methods for derived classes to inherit.

Function and Operator Overloading - The same function or operator name can refer to several alternative definitions provided each function definition has a different *signature* (i.e., different sequence of parameter definitions).

Dynamic Run-Time Binding - For overloaded functions, the selection of the correct function definition is held until the types of the calling parameters is identified at run-time.

Mandatory Methods - These are functions in a base class which all derived classes must inherit and may not redefine.

Virtual Functions - These are functions in a base class which all derived classes must inherit but may redefine.

Pure Virtual Functions - These are function signatures in a base class which all derived classes must supply.

2.5 Relationship of key features of C++ to OOD

Since Stroustrup designed C++ to support OOD, it is not surprising that key features of C++ support key features of OOD. Table 1 illustrates this correspondence. C++ programmers can implement abstractions of the central features of objects or functions through classes or templates. Classes in C++ correspond to classes in OOD. Templates allow developers to define general classes and functions which they can later tailor with specific type definitions; however, not all C++ compilers support templates.

C++ support for inheritance includes support for function and operator inheritance and for tailoring the inherited information. The base class specifies functions and data structures that the derived classes inherit and any access restrictions on that information. The base class can also include requirements on what functions the derived classes must contain. For example, a base class can require the derived class to supply a definition for a function (pure virtual function), provide default functions that a derived class can change (virtual functions), or require a derived class to use specific functions (mandatory methods). The base class also controls which data elements are inherited by declaring those data elements protected or public. C++ supports polymorphic function calls allowing several classes derived from a common base class to each provide a definition for the same function name.

C++ support for polymorphism includes type checking function calls to select the correct copy of the function, function and operator overloading and dynamic run-time binding. The linker

Table 1: Relationship between OOD and C++ Features

OO Feature	C++ Support
Abstraction	Classes Templates
Inheritance	Derived Classes Mandatory, Virtual, and Pure Virtual Functions Templates Function and Operator Overloading
Polymorphism	Run-Time Binding Type-Safe Linkages (strong typing) Function and Operator Overloading
Refinement	Base Classes C++ as design notation
Encapsulation	Classes Public, Protected and Private Data Public, Protected and Private Methods Inline Functions

must resolve overloaded function or operator calls at run-time since the specific type of the parameters may not be known until run-time. For example, if the base class animal has derived classes dog and cat, all three classes could define versions of the polymorphic function likes-food. When the program invokes this function with a parameter of type animal, the linker cannot choose the correct version of the function to use until the exact type of the calling parameter (dog, cat or some other animal) is available. If a function call contains arguments that do not match any signature for that function name, the linker should not select a particular version. This restriction requires C++ to use type-safe linkages.

C++ support for encapsulation is primarily support for classes and for access restrictions on information within classes. Classes can have public, protected and private data and functions. These different restrictions allow the developer to define an appropriate interface for accessing the data by only declaring as public those functions which should be part of the interface. The developer can use these restrictions to control access to the data and functions within the class definition. C++ requires the declaration of all access restrictions within the class definition. C++ also supports inline functions allowing the encapsulation of small blocks of code without performance penalties.

C++ supports refinement through very general mechanisms for implementing object models. C++ notation has also been used as a design notation. The flexibility of C++ allows for the implementation of designs developed using a variety of OOD methods.

3 OOD and C++ Reinforcing Features of Safety-critical Software

This study examines how OOD methods effect four software design principles, generally applicable to good software development but especially important for safety-critical software systems.

Modularity - A good system should consist of logically separate, defined components with defined interactions and limited access to data.

Functional diversity - A safety-critical system requires multiple, provably separate sequences of functions to provide independent checks on the data produced.

Traceability - An auditor should be able to map user requirements, statements in the analysis, elements of the design and lines of code to each other.

Removal of ambiguity - Safety-critical systems should not have ambiguous code or code that may not always have predictable effects.

This section contains four subsections, one on each of the above software development principles. Each subsection contains four parts. The first part is on the OOD features which support the incorporation of the design principle. The second part is on the support C++ provides for implementing designs incorporating the principle. The third part is on the potential risks OOD and C++ bring to incorporating the design principle. The fourth is a summary of the advantages and disadvantages of OOD and C++ with respect to the design principle.

3.1 Modularity

OOD Features Supporting -

Information hiding through encapsulation is central to OOD. Encapsulation is a mechanism for encoding abstract data types by packaging the data structures with the functions manipulating those data structures. Encapsulation requires narrow well-defined interfaces to data structures and the operations for those data structures providing more support for modular design and information hiding than traditional structural design methods. Since in a consistent OOD data are only accessible through defined interfaces and only functions encapsulated with the data can manipulate it, consistent use of OOD encourages and even enforces modular programming.

OOD also supports modularity through abstraction. By abstracting common elements of the design into classes and creating inheritance hierarchies with the common features of these classes, OOD reinforces the development of modular systems.

C++ Features Supporting -

C++ supports encapsulation and abstraction by defining classes as a programming language construct. C++ classes can encapsulate data structures by allowing the user to narrowly define the interface to those structures through the use of private and protected data structures and functions within classes. Private or protected data in a class are only available using functions defined within the class or its derived classes. The developer can keep the interface to the data narrow by defining a minimal number of the class functions public (i.e., available outside the class). To improve the efficiency of using functions to examine data structures rather than examining the data structures directly, inline functions allow the definition of small blocks of code as functions without performance penalty.

Risks -

The definition and use of classes in OOD and C++ for developing more modular programs can pose risks for safety-critical systems. One problem identified in [26] is the problem of creating too many classes and objects so that the system becomes unmanageable or has severe performance penalties [63, 71]. Because creating classes is a bottom-up process, it is easy to generate a large number of classes without creating manageable groups of classes. An accurate representation of the domain may include a large number of classes which are unnecessary for addressing the immediate problem. Askit and Bergmans [26] discuss the tendency of developers to create elaborate class hierarchies for reuse when only a small number of those classes are necessary for the current application. While the investment in creating an accurate, complex class hierarchy to represent a domain may pay off in its reusability in future projects, for a particular project, the effort may require a substantial investment including more testing and result in a performance penalty. The performance penalty can be substantial and effect the usefulness of real-time systems (e.g., [71]).

The problem of managing a large number of classes is similar to the general problem of managing a large design; however, top-down structured design methods have evolved to meet this need. Developers using OO methods can mitigate the risk by partitioning or grouping class definitions. Developers have proposed a variety of methods for grouping classes (e.g. ensembles [29], clusters [21] or class categories [9]) or partitioning the design (e.g., into subsystems [9, 10, 29] or layers [44, 52]). Since OOD is still evolving and each of these methods uses a slightly different rationale for creating groups of classes, no one method has universal acceptance. The only consensus is that good OOD practice for large system design alternates between top-down system partitioning and bottom-up class definition to mitigate the risks of either creating an unmanageable collection of classes or a system partition which does not reflect class interaction [26, 63].

One example of an OOD method attempting to mitigate these risks in large system development is the European Space Agency's Hierarchical Object-Oriented Design (HOOD) methodology [8, 15, 16] which was explicitly developed for large, complex, real-time, distributed systems. The HOOD methodology provides guidelines for decomposing abstract *parent* objects into more

detailed *child* objects for representing subsystems and developing an OO class hierarchy from the domain objects. There are available toolkits supporting the HOOD method (e.g., IPSYS HOOD Toolset).

Alternatively, Yamazaki, Kajihara, Ito and Yasuhara [44, 52] developed an in-house OOD method for a large telecommunications company that they are now publicizing to also address the need for better OOD methods for large systems. Their approach creates a layered architecture (e.g., hardware, subsystem, function) and associates entities in the requirements with specific layers. The developer then positions objects corresponding to the entities found during requirements analysis in the appropriate layer.

Another potential problem for developing modular designs is that arbitrary inheritance schemes or overuse of the *friend* declaration can bypass the development of good modular designs. The *friend* declaration bypasses normal access restrictions on class information. Inherited methods which add to the interface for accessing a class of objects make defining that interface more difficult. A developer can create abstractions of classes which capture important common information to develop the inheritance hierarchy providing a more modular design and reuse of common data and functions within the system; however, inherited data structures and functions make defining what an individual class does or can access more difficult. There is no good mechanism, and few auditing tools, for collecting all of the inherited information applicable to a class into a single location for purposes of examining the full definition of what an object does and which other objects can access its information.

Summary -

While there are potential risks to using OOD and C++, both provide strong support for modularity. Encapsulation is a mechanism for information hiding, and information hiding is a large part of why modular designs are desirable in large systems including safety-critical systems. C++ provides constructs supporting encapsulation and allows for the easy conversion of class access constraints into code. A major benefit of OOD and C++ is the explicit support of modular design and information hiding in the methodology. This support encourages the development of modular designs and their conversion into modular programs.

For OOD and C++ to support modularity, developers must make consistent use of their features. No design method or language choice alone will force developers to write modular programs. Developers still need a good software development process including design reviews and a quality assurance process to reinforce the use of OOD encapsulation and the accurate translation of class definitions into C++.

3.2 Functional Diversity/Redundant Software Components

OOD Features Supporting -

OOD supports functional diversity primarily through encapsulation. Encapsulation is a product of the object model which is central to OOD. Conceptually, the object model contains independent objects which exchange messages as needed. The actual operations and data necessary for the operations are hidden from every other object. The object model supports multiple objects or classes of objects which perform the same or similar functions but are completely independent of each other. Encapsulation, by providing a mechanism for limiting access to both functions and data within classes supports functional diversity.

For example, a system can include a pressure sensor class of objects and a temperature sensor class of objects with instances representing the actual states of pressure and temperature sensors. In Figure 3a, these two classes are completely independent of each other. The functions contained in the pressure sensor object cannot access the data or functions contained in the temperature sensor object except through the designated interface functions. In Figure 3a, the only interface functions are the display-pressure and display-temperature functions. Calculations made within the pressure sensor object would be completely independent of and provide a check on calculations made within the temperature sensor object.

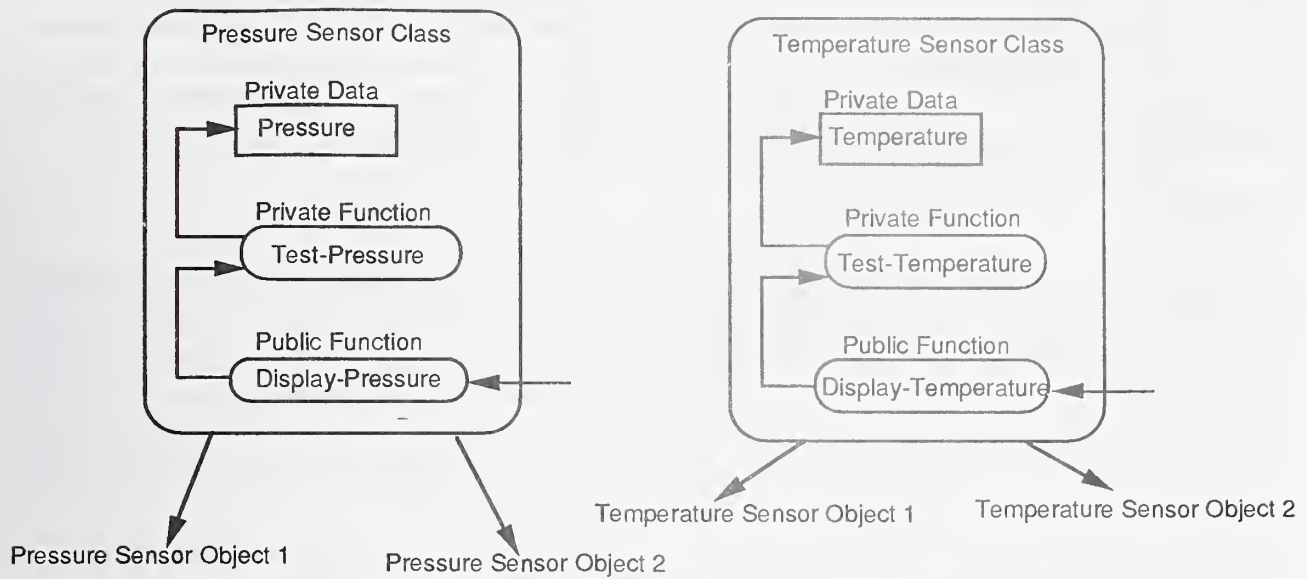
C++ Features Supporting -

C++ supports encapsulation by providing class definitions as discussed in the previous section. By defining data as private or protected within a class definition, the programmer can limit access to that data from functions outside the class definition or from derived classes inheriting from that class definition.

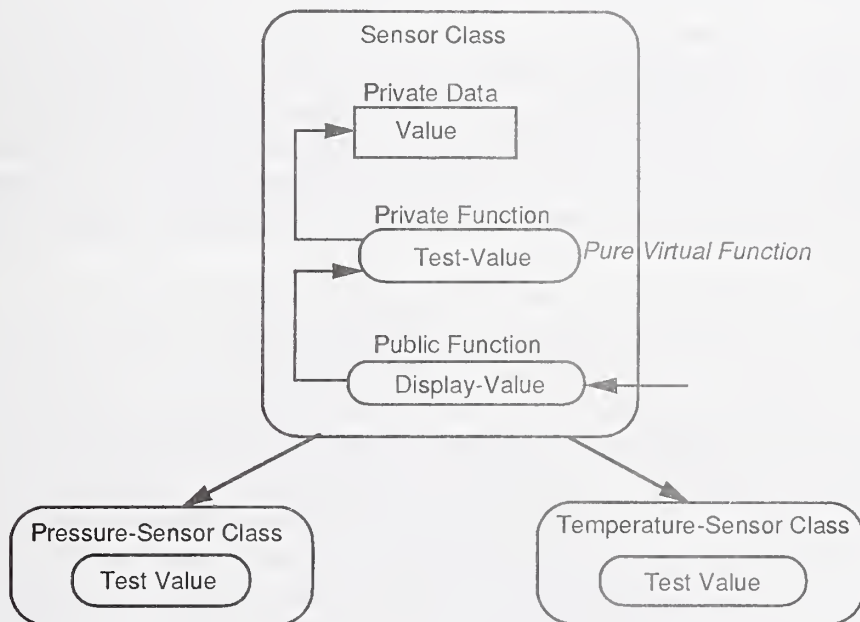
C++ class definitions also include an inheritance mechanism allowing base classes to force the creation of new operations in a derived class. If a function is a *pure virtual function* in a base class, each derived class inheriting from that base class must include a separate implementation of that function. This enforced function creation reinforces functional diversity since a pure virtual function definition requires separate function definitions to exist in each class derived from that base class. For example, in Figure 3b, there is a base class, sensor, with a pure virtual function, test-value. This declaration requires the derived classes pressure and temperature sensor to each provide a separate implementation of test-value. These functions would be polymorphic, possessing the same name but operating on different classes derived from the same base class.

Risks -

While private and protected restrictions within classes and pure virtual functions forcing the creation of new functions support functional diversity, inheritance can also allow unintended connections between functions and data. Classes can inherit data structures and functions from one or more base classes directly and from additional base classes indirectly. Since a class can



(a): Encapsulation and Functional Diversity



(b): Inheritance and Functional Diversity

Figure 3: OOD and Functional Diversity.

inherit from a series of classes through a hierarchy, a careless developer may not be aware of all the data and functions inherited by a new derived class. Overuse of inheritance mechanisms, especially multiple inheritance, and the use of friend declarations can create risks for enforcing both modularity and functional diversity requirements on systems if the developer does not explicitly watch for unintended connections among all the classes.

Summary -

Encapsulation by limiting access to data supports the creation of alternate methods from different classes accessing different data. C++ implements encapsulation through classes which can have private and protected data and functions limiting both access to and inheritance of that information. C++ class definitions can also require the creation of separate functions in derived classes through the definition of pure virtual functions. Alternatively, inheritance and friend declarations can create risks because these create unintended connections between classes and hide those connections in complex class hierarchies.

3.3 Traceability of Requirements to Design and to Code

OOD Features Supporting -

OOD should bridge the gap between the requirements and the code without performing translations or radical transformations of the design. The results of an OOA (object-oriented requirements analysis) should become the basis of the OOD which should become the OOP through a process of refinement and the addition of detail to the original design. There should be no sharp breaks or translations between phases of the lifecycle [21, 28]. To support traceability and the reusability of designs, there is a consensus among researchers that decisions forcing a particular implementation for any part of the design should be delayed as long as possible to make the gap between design and code narrower [44]. A more general design can also be available for reuse early in the design process of a subsequent system [68].

OOA provides the requirements analysis and initial design supporting traceability of the requirements, requirements analysis and initial design for OOD methods. OOA provides this traceability because it mixes bottom-up and top-down techniques just as requirements documents mix abstract and detailed requirements. Some methods [e.g., 14, 62] suggest collecting the significant terms in the requirements and turning the nouns into objects and the verbs into functions. While this approach is simplistic and impractical for large requirements documents, it does provide traceability and is useful for small problems or components of large systems. For large systems and requirements documents, abstract information is useful for organizing classes and breaking the problem into subsystems. The large OOA example mentioned in Section 2.2 [63] also showed the flexibility and traceability of OOA methods because the organization of the analysis as well as the details of the initial classes could be traced to specific requirements.

The OOD method should maintain the traceability of the design to the initial requirements through each refinement of the design to the actual code. Coplien [3] defines the *isomorphic*

structure principle as a goal for OOD methods. This principle states that the implementation structure and design abstractions should be parallel and easy to relate throughout the development process. The preferred method for maintaining this relationship is to refine the design throughout design development rather than radically changing the design. Coplien uses this principle as a criteria for analyzing OOD methods. He also defines a non-sequential series of steps for an OOD refinement process preserving the underlying data model. These steps follow:

- Identify the entities in your application domain
- Identify behaviors of the entities
- Identify relationships between entities, particularly subtypes
- Create a C++ design structure from the entities
- Implement
- Fine-tune
- Test

The developer should iterate through the steps to arrive at the final system; however, different portions of the target system may be at different points in the design process.

Coplien's approach is similar to other approaches in OOD [3]. One of the typical reasons cited for using OOD methods is the traceability of the requirements, design and code to each other [e.g., 3, 63].

C++ Features Supporting -

C++ supports OOD traceability by providing programming language structures for classes and objects and enforcing a variety of restrictions on class construction. C++ has been very popular for OOD because it is flexible enough to allow for the implementation of many of the object models assumed by the OOD methods. Those features of the language identified in Table 1 as supporting the various features of OOD also support the traceability of OOD elements into C++ constructs.

Risks -

A significant handicap to traceability is the lack of OO methods or tools for the full software lifecycle. Most OO methods cover, at most, only one portion of the life cycle [27] typically only the requirements analysis or design portion. This partitioning requires the use of multiple methods, tools and processes at different stages of the software lifecycle. If the developer does not choose the methodologies for his development process carefully, not all the selected methods, tools and processes may be compatible or support OO development.

Even when developers are only working with OO methods, these methods could be based on different object models with different properties for classes and objects. There may be minor or extreme differences among notations, languages, and CASE tools, in the default restrictions on classes, the functions performed on or with objects, and the limitations on inheritance. If the

developer is using tools or methods incorporating two or more different object models, the developer must translate between the two. This translation requires a defined mapping between the object models potentially changing the design or system in the process. This translation also makes tracing a requirement through its incarnations in the OOA, OOD and OOP more difficult since changes in the available properties of objects and classes may change how the design and program incorporate a requirement.

Thompson [73] proposed an object lifecycle scenario in which an object undergoes a number of transformations in the course of its development and use to illustrate the problem of incompatible object models. For example, a developer can use an object definition tool to develop a class, use a simulation tool to simulate the behavior of the class, store the class definition in a database or repository and use a coding tool to develop a C++ definition of the class. All of these tools and storage mechanisms can use different object models. These different models require multiple translations corrupting the original class and object definitions. This scenario reflects the current state of object technology with multiple competing object models embodied in the available tools and proposed in current standards. Selecting tools, methods, and notations which have incompatible object models prevents easy traceability in different phases of the lifecycle.

Summary -

Ideally, OOD supports traceability because it is a process of gradual refinement of the design details without major transformations and it is much easier to trace requirements through a design if the process involves refinement rather than translation. The requirements analysis and design documents should more closely map to the original requirements and domain analysis documents allowing the customer to understand the design better. There have been a number of successful examples of using these techniques to provide traceability. (e.g., [63])

To achieve traceability, the notation, tools, and implementation language must support compatible, if not identical object models. It is very easy to select incompatible tools or methods and have to translate among them. Developers must consider this issue when establishing the development process and selecting the methods, notations and tools to support that process.

3.4 Removal of Ambiguity

OOD Features Supporting -

This is a problem with C coding somewhat mitigated by C++. OOD is not relevant to this problem.

C++ Features Supporting -

C++ eliminates some of the sources of ambiguity present in C while introducing others. The sources of ambiguity problems from C that C++ mitigates follow in Table 2. Primarily, C++ offers type checking features and methods of avoiding explicit identification of addresses or use of pointers.

Table 2: Features Mitigating Ambiguity

Source of Ambiguity in C	C++ Features Mitigating Problem
Easy implicit conversions	Conversions not allowed Operator overloading
Incompatible function calls and signatures allowed	Type-safe linkages (parameter checking) Operator overloading
All calls by value, passing pointers	Explicit memory management functions (ex. constructors, destructors, copy constructors, new, delete) Call by reference

One of the major sources of confusion in C programs was the explicit use of address-of (&) operators and the use and nesting of contents-of (*) operations. These operations were essential because C only supported the passing of parameters by value. C++ supports the use of reference parameters allowing the programmer to eliminate the explicit use of pointers under some circumstances. C++ also allows the creation of constant references or parameters that cannot change during function execution which C did not support prior to the adoption of ANSI C.

Memory management has generally been simplified and hidden from the programmer. The simplification allows the programmer to avoid explicitly allocating and deallocating specific blocks of memory and eliminates some sources of errors since the programmer does not need to calculate the size of a memory block. Explicit unrestricted use of pointers and memory features makes program tracing, auditing and understanding more difficult.

Risks -

Even though C++ mitigates some problems, it introduces some new ones primarily through its support for inheritance, polymorphism and operator overloading through dynamic or run-time linking. If there are several polymorphic versions of functions (i.e., several functions with the same name operating on different types), the linker will have to choose among them or indicate the possible choices depending on the types of the arguments involved. Since the types of the

arguments may not be available until run-time, the version of the function used may not be known until runtime. This is true if the program declares the arguments to be instances of the base class to which all the versions of a polymorphic function apply rather than the derived classes used in the function signatures.

Essentially all polymorphic or overloaded functions are hidden branching statements in the program which an auditor inspecting the code may find difficult to trace. The options for polymorphic function calls and overloaded operators may be buried in several different classes scattered throughout the system. This scattering would make it difficult for the auditor to identify and collect the different options and all the possible branches the linker has available at any one point. If the options are too scattered, the auditor may not even be aware that an operator or function is overloaded.

4 OO and Real-time systems

Most safety-critical systems are real-time systems, and real-time systems have specific temporal requirements. Real-time systems include temporal characteristics associated with distributed components which may vary over time. A real-time control system must *classify* and *encapsulate* the behavior of these characteristics [34]. Classification is important for identifying the characteristics, and encapsulation is important for associating the characteristics with the correct components. A real-time system must use these temporal characteristics to recognize changes in the environment and adapt its control algorithms to these changes.

4.1 Advantages

Object models, as used in OOD methods, show promise for modeling the behavior of real-time distributed systems because object models consist of independent objects exchanging messages which require the receiver to take some action. One definition of OOD is "a design and implementation methodology which attempts as much as possible to compose the software of software objects which correspond to or encapsulate 'real' or 'physical' objects [34]." This definition comes from researchers who view OOD principally as a tool for the development of real-time systems. In many real-time domains, the environment that the system controls or mimics consists of controllers acting independently either serially or asynchronously to accomplish a goal. An actual process may require hundreds of separate controllers each having limited knowledge of the overall system and may require that the system have the ability to dynamically reconfigure itself. OOD can provide better models of the actual controllers and other interacting hardware components in a real-time domain than traditional data models. Encapsulated objects are a direct model for software controllers since each object can have a narrow interface and tightly controlled interaction with the other objects (software controllers). Changes internal to an object should have only a limited effect on the other objects in the system. This is analogous to changes internal to one controller having a limited effect on other controllers. Similarly, the reconfiguration of objects or controllers should have only a minimal effect on other objects or software controllers in the system [46].

The instability of the temporal characteristics of many real-time domains is another reason for choosing OOD over traditional software development methods [34]. Because temporal characteristics may not remain static over the life of the system and may not even be known until run-time, a real-time system must implement adaption algorithms for entities which minimize the impact of temporal changes on other entities. Objects by encapsulating temporal characteristics and behavior provide a mechanism for isolating those characteristics and behaviors to only those elements or objects directly effected.

There are several examples of deployed real-time systems developed using OOD methods [34, 46]. One example is in Kotchev et al. [46] which describes control software for a manufacturing process. This process consists of distinct phases with different types of activities conducted in each phase. The control software is an OOP supporting the deployment of independent controllers transferring control at established points in a process. Another example is the telecommunications software described in [34].

4.2 Problems encountered

The primary problems encountered with OOD in real-time systems relate to performance. Large, complex class hierarchies can produce very slow programs in part because over use of encapsulation and inheritance can add to the overhead in running C++ programs both from additional function calls and the need to dynamically locate the correct version of a polymorphic function. Performance must be taken into account at all phases of the development of real-time object-oriented systems [1, 2, 71].

5 Long Term Advantages of OOD

There are potential long-term advantages to OOD, discussed by a number of researchers, in addition to those cited in support of specific features of safety-critical systems. These long term advantages include the potential reusability of object-oriented code both within projects through inheritance hierarchies and between projects, the reduced cost and effort of maintaining code and the reduced cost of extending the code as the environment changes. These long term advantages should increase software development productivity and software product quality.

5.1 Reusability of Verified Code

One method of improving software productivity and quality and potentially reducing development costs is the reuse of verified software development artifacts in the creation of new systems. Software development artifacts include components of the design, code, and analysis of a system.

Rubin [68] identifies problems with the implementation of indexing schemes to identify specific items for reuse that the use of OOD methodologies can mitigate and some OOD can exacerbate. He identifies as a fundamental indexing problem the development of a representation for reusable artifacts. The lack of a standard notation or representation has hindered reuse efforts, but the definition of a narrow interface to objects as OOD methods provide has made the problem of

defining a representation "tractable" [68]. Further, a narrow interface to the objects simplifies the generation of queries and reasoning about the capabilities of objects. However, inheritance schemes by hiding some of the interface can complicate the representation and reasoning process.

Rubin [68] also identified two qualities of software development artifacts which effect the artifact's reusability. The qualities are the *abstraction level* and the *complexity* of the object. The abstraction level is the extent to which the artifact is dependant on a particular domain or implementation. By encouraging the development of abstract designs and the identification of common features of designs cutting across domains, OOD encourages the production of abstract, therefore, highly reusable components. The use of abstraction at the design or analysis level early in the software development process can potentially provide the greatest long term savings from reuse.

The second quality, the complexity of the software artifacts, is important because as the complexity of the artifacts increases, they become more difficult to understand and reuse. OOD through the encapsulation of data and operations in objects encourages modular development and information hiding. The developer intending to reuse the artifact need only understand its interface to know if a selected object is appropriate for a new application.

Stark [71] is less optimistic about the possibilities of general reuse of common artifacts, but emphasizes the benefits of reusing artifacts within a single domain. Using the experiences of NASA-Goddard's Software Engineering Laboratory (SEL), Stark cites an increase in the reuse of code from a baseline of 20-30% of new systems to 75-80% of new systems in a single domain over a seven year period in SEL projects using OO methods. Reuse of artifacts in projects outside the specific domain was much less common. Stark credits the difference to the developers understanding of the system artifacts associated with their domain and the difficulty of capturing that information for other developers. Stark and SEL conclude that OO methods promote reuse by providing a means to generate designs that accurately reflect the domain. Stark did cite, as a cost of reuse, OOD's neglect of other important issues especially run-time efficiency.

5.2 Ease of maintenance

Maintenance "is the performance of those activities required to keep a software system operational and responsive after it is accepted and placed into production." [55] Using OOD can significantly decrease the effort required to perform all three functions.

OOD tends to isolate faults by encapsulating data and functions within a single object or class. Since objects and classes have narrow interfaces, only changes to these interfaces will effect other classes. Changes within a class will only effect that class. Encapsulation generally limits the scope of errors and the scope of fixes to those errors.

Encapsulation also simplifies isolated improvements to the system. An OO system can, for example, change representation schemes or searching algorithms without effecting parts of the

system outside the class containing the scheme or searching algorithm. Substitution of poorly performing components is easier with only a narrow interface to those components.

5.3 Extensibility

Similarly, developers can easily enhance a system by adding new classes of objects especially if those classes are derived from existing base classes. For example, if the system has a base class *sensor* and derived classes *temperature-sensor* and *pressure-sensor*, it should be easy to add a new class *humidity-sensor* also derived from *sensor*. Any functions which accept sensors will accept humidity-sensors. The new class would have a similar interface to the other derived classes including specific polymorphic functions which would have to be defined for all three classes.

Traditional methods of development do not easily accommodate the evolution of a software system to encompass new features or changes in the environment. OOD potentially can accommodate new objects or variations on existing objects in a given environment, and since good OOD encapsulates the functions operating on a given structure in an object, the effects of changes to that structure should be hidden within the object.

6 Conclusions

OOD is a valuable design methodology for real-time safety-critical systems because it can provide a more direct model of the domain than traditional design methods and can reinforce the use of development principles important to safety-critical systems (e.g., modularity, functional diversity, traceability and removal of ambiguity). OOD by modeling independent controllers as separate objects can isolate the functions and temporal characteristics of those objects and better model the actual hardware under software control. OOD has now been used in a number of large, successful real-time systems especially in the telecommunications area.

OOD also reinforces software development principles important to the development of safety-critical systems. The OOD properties of encapsulation, abstraction, inheritance and refinement reinforce the safety-critical design features of modularity, functional diversity and traceability. Encapsulation and abstraction reinforce the development of modular code with narrow interfaces to separate blocks of data and functions by making information hiding central to OOD. Encapsulation also supports functional diversity because objects encapsulate both data and functions supporting the development of independent functions operating on independent data elements. The inheritance properties of objects can also support functional diversity by forcing the creation of new functions. OOD supports traceability because objects can provide a direct, understandable model of the real world. With careful selection of compatible OO development methods, tools and notations, the requirements, requirements analysis, initial design, intermediate designs, final design and code should all be traceable to each other. OOD is effective because it takes good software development practices and formalizes their use within the design paradigm. Table 3 summarizes the advantages of OOD and C++ development methods with respect to these safety-critical features.

Use of an object-oriented programming language is essential for the consistent use and deployment of an OOD. C++ supports OOD and is compatible with a popular language used in safety-critical systems. C++ provides a number of features (see Section 2.3) which support the easy transference of OOD into code. C++ also provides features making C a safer language primarily by minimizing the need to directly manipulate pointers and memory and by providing type-safe linking.

Even though OOD and C++ provide a number of benefits, OOD and C++ are not a panacea. Table 3 also summarizes the risks of OOD and C++ development with respect to the safety-critical systems features. These techniques will reinforce good practices and the features important to the development of safety-critical systems, but will not supply those features by themselves. In addition both introduce new risks which the careful selection or definition of a development methodology can mitigate. The use of C++ and OOD must be part of an overall software development process reinforcing quality development practices.

Table 3: Summary of the Risks and Benefits of OOD and C++

	Benefits		Risks	
	OOD	C++	OOD	C++
Modularity	Encapsulation	Classes with access restrictions	Class hierarchy complexity Inheritance	Base/derived classes Function & operator overloading
Functional Diversity	Encapsulation Polymorphism	Classes with access restrictions & required function definitions Dynamic run-time binding Function & operator overloading	Inheritance	Function & operator overloading
Traceability	Refinement	Support for OOD	Lack of Compatible OOA, OOD, and OOP methods, notations, languages and tools	
Removal of Ambiguity		Type-safe linkages Call by reference Memory management		Operator & function overloading Dynamic run-time binding
Real-Time Systems Development	Encapsulation Object model Refinement	Classes	Class hierarchy complexity	Performance

Bibliography

C++ Programming

[1] Carson, John H. *The C++ Programming Language*. Lecture Notes, John H. Carson Associates, 1993.

[2] Carson, John H. *Object-Oriented Programming and the C++ Programming Language*. Lecture Notes, John H. Carson Associates, 1993.

[3] Coplien, James O. *Advanced C++: Programming Styles and Idioms*. New York: Addison-Wesley Publishing Co., 1992.

[4] Mancel, D. and W. Havanas. "A Study of the Impact of C++ on Software Maintenance" from *Proceedings: Conference on Software Maintenance*, pp.63-69. Los Alamitos, CA: IEEE Computer Society Press, 1990.

[5] Stroustrup, Bjarne. *The C++ Programming Language, Second Edition*. Addison-Wesley, 1991.

[6] Stroustrup, Bjarne and Margaret Ellis. *The Annotated C++ Reference Manual*. Addison-Wesley, 1990.

Object-Oriented Design Methodologies

[7] Alagar, V.S. and K. Periyasamy. "A Methodology for Deriving an Object-Oriented Design from Functional Specifications" from *Software Engineering Journal*, Vol. 7, Issue 4, pp. 247-263, July, 1992.

[8] Aslett, M.J. "An Overview of the HOOD Method" from *IEE Colloquium on An Introduction to Software Design Methodologies*, pp. 5/1-4. London: IEE, 1992.

[9] Booch, G. *Object-Oriented Design with Applications*. The Benjamin Cummings Publishing Company, Inc., 1991.

[10] Coad, P. and E. Yourdon. *Object-Oriented Analysis*. Prentice Hall, 1991.

[11] Coad, P. and E. Yourdon. *Object-Oriented Design*. Prentice Hall, 1991.

[12] Cox, Brad J. and Andrew J. Novobilski. *Object-Oriented Programming: an Evolutionary Approach*. New York: Addison-Wesley Publishing Co., 1991.

[13] EVB. *An Object-Oriented Design Handbook for ADA Software*. Frederick, MD: EVB Software Engineering, Inc., 1985.

[14] Fayad, Mohamed E., Louis J. Hawn, Mark A. Roberts, and Jerry R. Klatt. "Using the Shlaer-Mellor Object-Oriented Analysis Method" from *IEEE Software*, March, 1993.

[15] HOOD Working Group. *HOOD Reference Manual*. (HRM/91/07/V3.1) Noordwijk, Netherlands: European Space Agency, 1991.

- [16] HOOD Working Group. *HOOD User's Manual, Issue 3.0*. Noordwijk, Netherlands: European Space Agency, 1990.
- [17] Hull, M.E.C., A. Zarea-Aliabadi and D.A. Guthrie. "Object-Oriented Design, Jackson System Development (JSD) Specifications and Concurrency" from *Software Engineering Journal*, Vol. 4, Issue 2, pp. 79-86, March, 1989.
- [18] Jackson, M. *System Development*. Englewood Cliffs, NJ: Prentice Hall, 1983.
- [19] Lieberherr, Kar. J., Paul Bergenstein, and Ignacio Silva-Lepe. "From Objects to Classes: Algorithms for Optimal Object-Oriented Design" from *Software Engineering Journal*, pp. 205-228, July, 1991.
- [20] Meyer, B. *Object-Oriented Software Construction*. Hemel-Hempstead, England: Prentice-Hall, 1988.
- [21] Nerson, Jean-Marc. "Applying Object-Oriented Analysis and Design" in *Communications of the ACM*, Vol. 35, No. 9, September, 1992.
- [22] Rumbaugh, J. et al. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.
- [23] Shlaer, Sally and Stephen J. Mellor. *Object-Oriented Systems Analysis*. Prentice-Hall, 1988.
- [24] Wirfs-Brock, R. et al. *Responsibility-Driven Design*. Prentice-Hall, 1990.
- [25] Yonezawa, A. and Tokoro, M., editors. *Object-Oriented Concurrent Programming*. MIT Press, 1987

Surveys of Object-oriented Design Methodologies

- [26] Askit, Mehmet and Lodewijk Bergmans. "Obstacles in Object-oriented Programming" in *Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications*. ACM, 1992.
- [27] Blair, Gordon, John Gallagher, David Hutchinson, and Doug Shepherd. *Object-Oriented Languages, Systems and Applications*. NYC: John Wiley and Sons, 1991.
- [28] De Champeaux, Doug Lea and Penelope Faure. "The Process of Object-Oriented Design" in *Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications*. ACM, 1992.
- [29] De Champeaux, Doug Lea. "Object-Oriented Analysis and Top-Down Software Development" in *Proceedings of the European Conference on Object-Oriented Programming*, pp. 360-375, 1991.

[30] Fichman, R.G. and C.F. Kemerer. "Object-Oriented and Conventional Analysis and Design Methodologies" from *Computer*, Vol. 25, Issue 10, pp. 22-39, October, 1992.

[31] Wirfs-Brock, R. and R. Johnson. "Surveying Current Research in Object-Oriented Design" in *Communications of the ACM*, Vol. 33, No.9, pp. 104-124, September, 1990.

[32] Ziegler, Bernard P. *Object-Oriented Simulation with Hierarchical, Modular Models*. New York: Academic Press, 1990.

Applications of OOD Methodologies

[33] Arefi, F., C.E. Hughes, and D.A. Workman. "The Object-Oriented Design of a Visual Syntax-Directed Editor Generator" from *Proceedings of the 13th Annual International Computer Software and Applications Conference*, pp. 389-96. Washington: Computer Society Press, 1989.

[34] Bihari, T., P. Gopinath, and K. Schwan. "Object-Oriented Design of Real-Time Software" from *Proceedings: Real Time System Symposium*, pp. 194-201. Los Alamitos, CA: IEEE Computer Society Press, 1989.

[35] Boettcher, C.B. and R. Poster. "Object-Oriented Design of Radar Warning Receiver Application Software" from *Proceedings: IEEE/AIAA 10th Digital Avionics Systems Conference*, pp. 550-554. New York: IEEE, 1991.

[36] Bower, W., C. Seaquist, and W. Wolf. "A Framework for Industrial Layout Generators" from *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, Issue 5, pp. 596-603, May, 1991.

[37] Campbell, W. "A C Interpreter for Scheme - an Exercise in Object-Oriented Design" from *Software Engineering Journal*, Vol. 6, Issue 4, pp. 130-136, July, 1991.

[38] Cappellini, V. and A. Del Bimbo. "Image Processing System Based on Object-oriented Design" from *Sixth Multidimensional Signal Processing Workshop*, pp. 21-22. New York: IEEE, 1989.

[39] Chang, A.-M., P.K. Kannan, and B.O. Wong. "Design of an Object-Oriented System for Manufacturing, Planning, and Control" from *Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing*, pp. 2-8. Los Alamitos, CA: IEEE Computer Society Press, 1990.

[40] DaPonte, P., Nigro, L., and Tisato, F. "Object-Oriented Design of Measurement Systems" from *IEEE Transactions on Instrumentation and Measurement*, Vol. 41, Issue 6, PP. 874-880, December, 1992.

- [41] Evans, C. "Construction of an Object-Oriented User Interface" from *IEE Colloquium on Application and Experience of Object-Oriented Design*, pp. 8/1-3. London: IEE, 1989.
- [42] Fayad, M.E., L.J. Hawn, M.A. Roberts, and J.R. Klatt. "Mission Generation System (MGS): an Application of Shlaer-Mellor's Object-Oriented Method" from *Proceedings: IEEE/AIAA 10th Digital Avionics Systems Conference*, pp. 91-96. New York: IEEE, 1991.
- [43] Ganti, M., P. Goyal, R. Nassif, and S. Podar. "An Object-Oriented Application Development Environment" from *COMPCON Spring '90: Thirty-Fifth IEEE Computer Society International Conference*, pp. 348-355. Los Alamitos, CA: IEEE Computer Society Press, 1990.
- [44] Kajihara, K., S. Yamazaki, T. Yamashita, and M. Ito. "An Application of Object-Oriented Design for Communication Control Systems" from *Proceedings: Fourteenth Annual International Computer Software and Applications Conference*, pp. 43-51. Los Alamitos, CA: IEEE Computer Society Press, 1990.
- [45] Knolle, Nancy T., Martin W. Fong, and Ruth E. Lang "SITMAP: a Command and Control Application" from *Applications of Object-Oriented Programming*. ed. by Lewis J. Pinson and Richard S. Wiener. NYC: Addison-Wesley Publishing Co., 1990.
- [46] Kotchev, B., A. M. Levy, P. Petrov, and R. Patrashkov. "Structuring the application software for Real-Time Process Control" from *Proceedings: EUROMICRO '90 Workshop on Real-Time*, pp. 189-196. Los Alamitos, CA: IEEE Computer Society Press, 1990.
- [47] Lahti, G., I. Ashkenazi, C. West, and B. Morgan. "Embedded Software for the CEBAF RF Control Module" from *Conference Record of the IEEE Particle Accelerator Conference, Vol. 2*, p. 1290-2. New York: IEEE, 1991.
- [48] Menga, G., Morisio, M., and Mancin, M. "A Framework for Object-oriented Design and Prototyping of Manufacturing Systems" from *Proceedings: IEEE International Conference on Robotics and Automation, Vol. 1*, pp. 128-135. Los Alamitos, CA: IEEE Computer Society Press, 1991.
- [49] Mock, Mary. "DoubleVision: A Foundation for Scientific Visualization" from *Applications of Object-Oriented Programming*. ed. by Lewis J. Pinson and Richard S. Wiener. NYC: Addison-Wesley Publishing Co., 1990.
- [50] Taylor, J.M. "Cooperative Computing and Control" from *IEE Proceedings E: Computers and Digital Techniques*, Vol.137, Issue 1, pp. 1-16.
- [51] Wu, Thomas C. "Development of a Visual Database Interface: An Object-Oriented Approach" from *Applications of Object-Oriented Programming*. ed. by Lewis J. Pinson and Richard S. Wiener. NYC: Addison-Wesley Publishing Co., 1990.

[52] Yamazaki, S., Kajihara, K., Ito, M., and Yasuhara, R. "Object-Oriented Design of Telecommunication Software" from *IEEE Software*, Vol. 10, Issue 1, pp. 81-87, Jan., 1993.

[53] Yang, L., M.A. Perkowski, and D. Smith. "Object-Oriented Design of an Expandable hardware Description Language Analyzer for a High-Level Synthesis System" from *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, Vol. 2, pp. 529-538. Los Alamitos, CA: IEEE Computer Society Press, 1991.

Reference Works

[54] "Glossary of Software Engineering Terminology." ANSI/IEEE Standard 729-1983.

[55] "Guideline on Software Maintenance." Federal Information Processing Standard 106, Department of Commerce, National Bureau of Standards, U.S., 1984.

[56] "Survey of Diagrams on Charting Techniques in the Area of Inference-Based Systems and Object-Oriented Programming" New Work Item, ISO/IEC JTC1/SC7/WG11, 1993.

Additional Papers on OOD Topics

[57] Anderson, B. "Process and Reusability in Object-Oriented Programming" from *IEE Colloquium on Application and Experience of Object-Oriented Design*, pp. 4/1-3. London: IEE, 1991.

[58] Atkinson, Colin. *Object-Oriented Reuse, Concurrency and Distribution: an ADA-based approach*. NYC: Addison-Wesley Publishing Co., 1991.

[59] Buescher, T.W. and R.T. Wilkinson. "Requirements Modeling for Real-Time Software Development" from *Proceedings of the IEEE National Aerospace and Electronics Conference*, Vol.2, pp. 613-617. New York: IEEE, 1990.

[60] Coleman, D. and F. Hayes. "Getting the Best from Objects: the Experience of HP" from *IEE Colloquium on Application and Experience of Object-Oriented Design*, pp. 6/1-3. London: IEE, 1991.

[61] Cooling, J.E. and T.S. Hughes. "The Emergence of Rapid Prototyping as a Real-Time Software Development Tool" from *Second International Conference on Software Engineering for Real Time Systems*, pp. 60-64. London: IEE, 1989.

[62] Lieberherr, K.J., and I.M. Holland. "Tools for Preventive Software Maintenance" from *Proceedings: Conference on Software Maintenance*, pp. 2-13. Washington: IEEE Computer Society Press, 1989.

- [63] Lubars, Mitchell, Coline Potts, and Charles Richter. "Developing Initial OOA Models" from *Proceedings: 15th International Conference on Software Engineering*. IEEE, 1993.
- [64] Manel, Dennis and William Havanas. "A Study of the Impact of C++ on Software Maintenance."
- [65] McGregor, D.R. "Reusability - the Major Promise and Challenge of the Object-oriented Approach" from *IEE Colloquium on Application and Experience of Object-Oriented Design*, pp. 7/1-6. London: IEE, 1991.
- [66] Mitchell, R.J. "Improving Object-Oriented Software Design" from *IEE Colloquium on Advances in Optimisation*, pp. 2/1-4. London: IEE, 1989.
- [67] Randell, B., and J.-C. Fabre. "Fault and Intrusion Tolerance in Object-Oriented Systems" from *Proceedings: 1991 International Workshop on Object-Orientation in Operating Systems*, pp. 180-184. Los Alamitos, CA: IEEE Computer Society Press, 1991.
- [68] Rubin, K.S. "Reuse in Software Engineering: an Object-Oriented Perspective" from *COMPCON Spring '90: Thirty-Fifth IEEE Computer Society International Conference*, pp. 340-346. Los Alamitos, CA: IEEE Computer Society Press, 1990.
- [69] Satoh, Ichiro and Mario Tokoro. "A Formalism for Real-Time Concurrent Object-Oriented Programming" from *Proceedings: Conference on Object-Oriented Programming Systems, Languages, and Applications*, pp. 315-326. ACM, 1992
- [70] Smith, Dennis, Cliff Huff, Ed Morris, and Paul Zarrella. *Software Engineering Environment Evaluation Issues*. Technical Report. Pittsburgh, PA: Software Engineering Institute, 1993.
- [71] Stark, Michael. "Impacts of Object-oriented Technologies: Seven Years of SEL Studies" from *Proceedings of the Seventeenth Annual Software Engineering Workshop*, NASA Goddard Space Flight Center, 1992.
- [72] Walker, I.J. "Requirements of an Object-Oriented Design Method" from *Software Engineering Journal*, Vol. 7, Issue 2, pp. 102-113, March, 1992.
- [73] *Workshop Report: Workshop on Application Integration Architectures*. NIST Special Publication, NIST, 1993.

5.4.1 Questions: Dr. Barbara B. Cuthill

QUESTION: JAMES CHELINI (Raytheon): You stated that C++ is compatible with a language popular for writing safety-critical systems, I assume C. Is C chosen because it's good for developing critical systems, or because it is widely available in school and commercial use and, by default, has been used in critical systems? What limitations would you impose on the C++ OOD implementation?"

DR. CUTHILL: C is being used. That's the point. It's there. It's what's being used. Whether that was the best choice, it's what we have.

When using C++, there have to be limitations on the use of inheritance, function overloading, polymorphism. These things create ambiguities and their use has to be limited. Possibly it can't be supported at all in safety-critical systems. I think that's an issue that has to be looked at further.

QUESTION: DR. JOHN McHUGH (Portland State University): With the exception of explicit inheritance, every advantage that you attribute to C++ is done, and done better, in ADA83, which is standardized and much more portable. ADA9X will have inheritance. C++ allows all the faults of C. Why would anyone use it?"

DR. CUTHILL: I have not looked at ADA. What we need is an examination of C++, ADA9X, EIFEL. We need a study of all of these and their benefits for safety-critical systems. We need that kind of a study, we need that kind of research. It's not there.

DR. McHUGH: Those studies have been done.

DR. CUTHILL: Well, I'd like to know your references.

DR. McHUGH: Okay.

MS. DOLORES R. WALLACE (NIST): Barbara was actually asked to look at C++, which is what she did.

QUESTION: How do you measure objectively the increase in modularity using OO techniques versus other approaches, and how do you model the trade-off with performance prior to implementation?

DR. CUTHILL: There are simulation languages, packages, that will let you look at performance trade-offs. Also, the problem is overly complex inheritance hierarchies- nd there are complexity metrics for checking for that. As for the first part, measuring the increase in modularity, I don't have a good answer for that except that encapsulation makes information hiding central to the design process. So, it is a key part of the design method and is supported by the method, which should make the designs more modular and provide better enforcement of modularity.

6 METHODS FOR REDUCING RISKS IN SOFTWARE SYSTEMS

The technical session on risk reduction provided interpretation on both the problems in achieving and assuring the safety of high integrity software and possible solutions to the problems. The speakers, Dr. Winston Royce (TRW, Inc.), Ms. Anne-Marie Lapassat (Commissariat à l'Energie Atomique), Mr. H. Ronald Berlack (Configuration Management International), Dr. Lance A. Miller (Science Applications International Corporation), Ms. Charlotte O. Scheper (Consultant), Mr. Kyle Y. Rone (IBM), Dr. William Everett (AT&T), and Mr. Roger U. Fujii (Logicon, Inc.), provided insights from their experiences in defense, nuclear, space, and communication industries.

Dr. Royce identified the following six areas of concern with regard to safety-critical systems, and elaborated on their meaning during his presentation:

1. Safety-critical systems are implemented in the *wrong languages*.
2. There are *not enough tools* for safety-critical systems development.
3. There is insufficient analysis and distribution of *error measurement* data.
4. There is *no organizational certification*.
5. There is *no people certification*.
6. There is *infrequent graspability* of the full system functionality.

Ms. Lapassat, whose presentation preceded Dr. Royce's, and the six speakers immediately following Dr. Royce, provided some techniques for addressing most of these problems.

While several speakers discussed the need for and availability of good tools, the only speaker in this session to focus on tool development was Ms. Lapassat. She discussed simulation, testing and auditing tools for independent evaluation of the software in NPPs developed for the French nuclear regulatory agency. These tools focused on testing the control system to see if it met the required timing constraints by simulating real-time, normal and abnormal operation.

Other speakers discussed tool availability and usage in relation to support for specific processes. Mr. Berlack focused on the importance of SCM and that organizations need to have both the tools and processes in place to support CM. Dr. Miller focused on the need for organizations to support V&V and to use automated V&V test tools whenever possible.

Two complementary techniques were discussed as ways to manage the complexity of large software projects: CM and reuse. Mr. Berlack strongly endorsed the use of CM as a means of communicating between the systems and software engineers and maintaining traceability. Good CM tools can provide a mechanism for tracing the impact of one change on the overall system. Dr. Miller reiterated this need for CM to trace and allocate requirements to all the development artifacts (e.g., specifications, code) as a means of supporting V&V.

Ms. Scheper described typical approaches for certifying and reusing software components. She provided an alternative approach to simplifying complex systems through the certification of reusable components. Ms. Scheper developed a certification framework for software for high

integrity systems. Software components are maintained with the requirements they meet. The framework grades the requirements and component based on the level of confidence that the component meets the specification, the level of criticality of the software, and the level of assurance used to test the software.

Three of the speakers discussed the need for and use of empirical data on the software development process and software error measurement. Mr. Rone discussed the need for organizations to use models of error discovery. Dr. Everett addressed the question: "Can we apply software reliability engineering techniques to safety-critical systems?" Dr. Everett discussed the test acceleration methods that isolate safety-critical functions for extensive testing to achieve higher estimates of reliability. Dr. Miller also discussed the use of models to estimate the numbers and types of errors remaining in a system. Many speakers and members of the audience agreed that public availability of error data would provide valuable information but that it is not realistic to expect companies to release this data.

While no speaker directly addressed Dr. Royce's call for certifying an organization's capability to produce safety-critical software, several speakers discussed the need for organizations to implement repeatable processes supported by CM. These capabilities are necessary to establishing a corporate ability to produce software. Both Mr. Berlack and Mr. Rone identified process definition as a required precondition for an organization to produce useful metrics data. Without a defined process, it is not clear what the collected data means across different projects. Mr. Rone linked the production of usable metrics data to quality, cost, and schedule planning.

The speakers also linked process definition, metrics collection, project planning, and V&V activities to CM. Mr. Berlack explicitly discussed the link between CM and the establishment of consistent planning, traceability, formal releases, change management, status accounting and auditing. All of these capabilities are important for an organization to define its software development process. Mr. Fujii and Dr. Miller discussed the need for CM and specifically traceability to support a V&V process.

Mr. Fujii discussed software V&V in the context of the system, i.e., software V&V is a systems engineering discipline that evaluates software as part of the entire system, including hardware, human operators, and other interfacing software (e.g., operating systems, printers). He provided guidance on estimating the cost of software V&V based on two concerns:

1. The criticality of system-specific functions and other system parameters (e.g., security, usability, maintainability, and performance).
2. Risks of the development environment (e.g., system architecture maturity, processor technology suitability, development methodology, maturity of development tools and aids, staff skills, schedule, and software application maturity).

The criticality analysis process requires traceability from system functions to all other components to define the system behavior and to trace the implementation of the system functions through all system documentation and code.

Mr. Fujii described a system safety framework that assists in estimating how much of the software to analyze and test. In modern systems the interaction of software with the hardware, human operators, and other software elements is more complex and interwoven in the total system solution than existed previously. The system performance functions must be specified and analyzed. Software V&V must also analyze the allocation of the system requirements to ensure that critical requirements are traceable and are allocated so as to make integration and testing less difficult and time-consuming.

No speakers in this session addressed the other two issues that Dr. Royce mentioned: language selection or developer certification. However, speakers in other sessions, the panelists in the last session, and many audience members discussed and debated these issues.

In this session, like the previous technical session, two major themes were the importance of maintaining and verifying the traceability of specifications across a complex system, and maintaining the system context for the software. Mr. Berlack discussed the use of CM as a vehicle for communication between the systems and software engineers because it provides the mechanism for tracing the elements of development artifacts back to the specifications. Dr. Miller emphasized the importance of traceability to V&V activities. Mr. Fujii emphasized the need for systems engineers not only to be able to trace the specification to the design implementation but to be heavily involved in the software V&V process since the systems engineers know how the full system, not just the software, should behave. Ms. Scheper also emphasized the need to understand the requirements on software in the context of a full system to be able to select reusable software components and catalog those components correctly.

**Digital Systems Reliability
and Nuclear Safety Workshop**

**Automated Tools for Safety-Critical
Software**

Anne-Marie Lapassat

LETI(CEA-Technologies Avancées)
DEIN-CE/S F91191 Gif sur Yvette Cedex

September 13-14, 1993 Rockville, MD

Abstract

In France, the licensing process of nuclear power plants includes a detailed survey of the nuclear reactor Instrumentation and Control and in particular of the Protection System which is classified as a Safety Critical System.

The Safety of such a system is mainly insured by the quality of the development process including separated teams for development and validation. However, before acceptance, an independent evaluation is realized by Institute for Protection and Nuclear Safety (IPSN), comprising a critical examination of the documents, an assesment of the code quality, an identification of the critical software components, a choice and preparation of tests cases, and dynamical verifications of consistency and robustness.

To help that Independant Verification &Validation (IV&V) process, the tool CLAIRE-OST was developed at CEA-DEIN to cover dynamical verifications. Based on the priciple of test by simulation, it is independant of the specification methods and languages used by the Safety Critical System Provider and it is flexible enough to allow different types of modelling and categories of test.

Further development consists of helping the generation of test cases, based on proof techniques and formal models of specifications.

1 Requirements of IV&V

Applications to be evaluated:

Safety Critical I&C systems are real-time applications and one important requirement for the evaluation is the possibility to verify temporal specifications, watch-dogs and synchronizations bewteen components.

They are multi-microprocessors applications and the verification of links or independancies are of interest. The real installation is made of a great amount of microprocessors and hardware components.

They are fault-tolerant, they must detect and manage defaults on the installation, the instrumentation and themselves.

IV&V environment

The team realizing IV&V, has not generally a physical item of the system to be verified and at the begining, they could only inspect the documents, that is the main reason which oriented us toward a simulated environment.

They may have successive and quite different types of systems to analyse.

For example, in the case of the french nuclear park, which is quite homogeneous, IPSN has to manage in parallel evaluations of SPIN P4 evolutions and of SPIN N4, plus other systems, all of them using different hardware technologies, architecture type, and languages. SPIN P4 is based on a M6800 technology and asynchroneous communications for hardware part, manual methods of development and assembler language for software. SPIN N4 is based on M68000, local networks for hardware, and the C language code used for developing the software is generated from a formal specification.

2 Objectives of CLAIRE-OST Tool

The objective was to produce a software tool allowing:

- to simulate a code with an exact representation of its execution on the target microprocessor;
- to represent every nominal interactions between the software on its microprocessor, and its environment such as inputs from acquisition systems, and outputs toward display and actuators. The temporal aspects as well as functional ones had to be represented;
- to describe and simulate abnormal interactions such as failures on acquisition systems . . . , in order verify the behaviour of the code when defaults occur;
- to introduce defaults on the microprocessor and its memory and to verify their detection;
- to represent and play test sequences;
- to introduce an observer able to compare expected values to test results;
- to keep preparations and results of test in order to replay the same test sequences in particular after evolutions of the software.

The result was an event simulator associated with emulators of microprocessors, a system for modeling and simulating real time environment.

The first version of OST (Tool for Simulation and Test) accommodated only one emulated microprocessor, the second one allowed emulation of several codes and their environment.

A graphical editor allows one to model and generate the complete test environment according to the rules of our real time model.

The complete CLAIRE toolset is represented by the following figure where dotted lines represent non-automated functions:

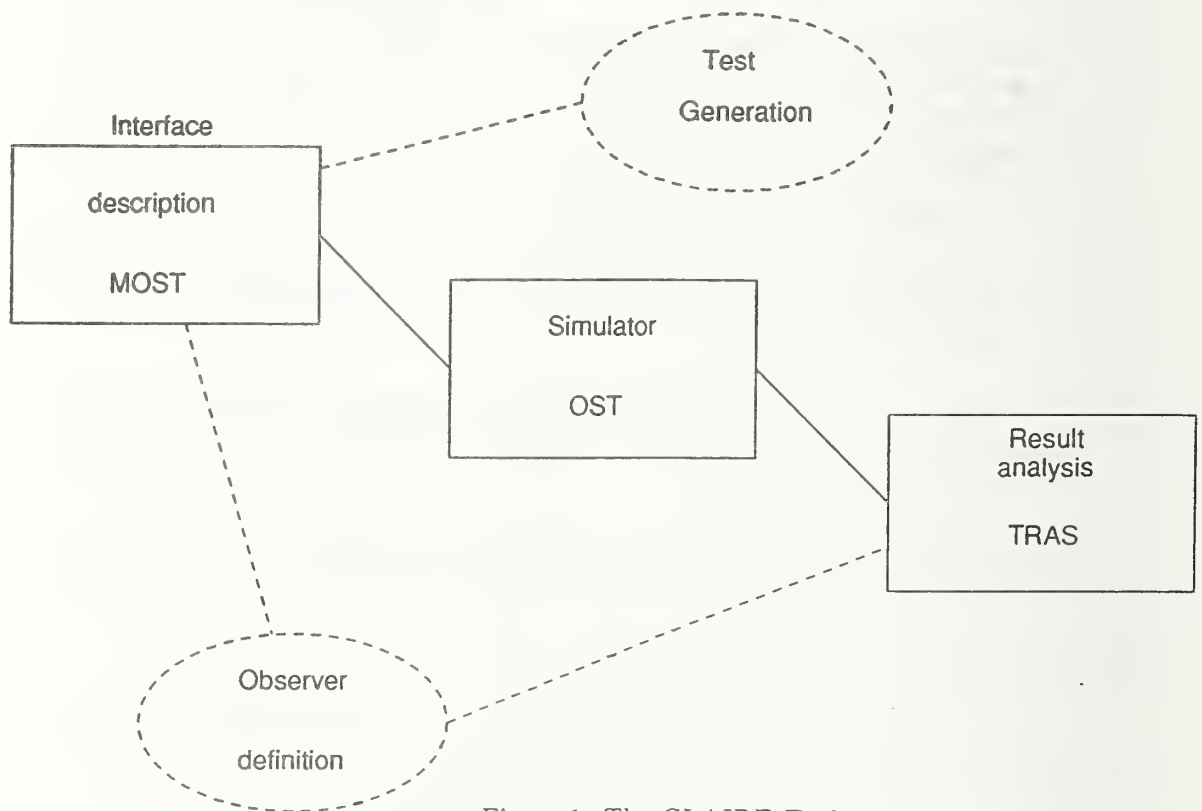


Figure 1: The CLAIRE Toolset

3 Simulation Principles : OST

The structure of the OST simulator is given in fig. 2:

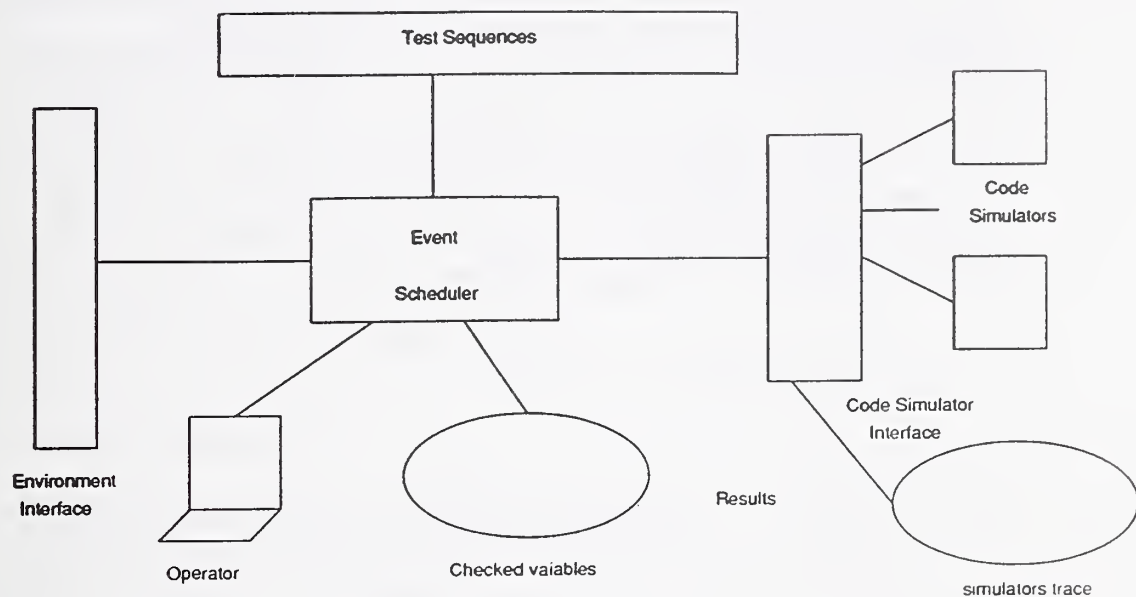


Figure 2: The Structure of the OST Simulator

Each interface (microprocessor emulators, environment interface, test sequences interface) is activated when an event addresses it. When activated, it executes what is required by that current event, eventually creating new events and returning control to the scheduler.

An event is a command to be executed at an explicit date. For example, every modification of a variable of the simulated system will be the result of an event:

- At time 500ms Modify the variable *THRESHOLD* with the value 56.

Execution of the next instruction of the microprocessor code will be emulated at the occurrence of an event:

- At time 501ms Execute current instruction *FU4*.

Time evolves with the *Date of execution* of the current event, an interface started by an event works in *null virtual time*, but may generate *delays*. The date of execution are always explicit and the simulation works in virtual time.

The environment model is structured in modules themselves made of *PROCEDURES* which are the Active Units of the simulation. Each procedure is activated at the occurrence of an event modifying a certain variable, which is a signal, when it starts executing calculations, and creates new events at the current time (execution in null virtual time) or with a delay and returns to the scheduler.

The microprocessor code emulators are commercial that were interfaced with the tool under the following conditions:

- The emulators accept step by step simulation and may save and restore their context (register and memory).
- The emulators allow themselves to consult and modify their context.

When integrated, every emulator is able to work in parallel, this means that an application may be made of two M68000 and two In8051 microprocessors. For example, if you are testing the interaction of calculation codes communicating via network controllers.

With that real time model of simulation, we are able to model the real time behaviour of parallel systems. The tool is of course able to simulate without the microprocessor code and in this case, we apply to verification of the specifications.

4 Graphical description : MOST

The graphical description tool MOST gives the user a mean to develop his test model under the rule described above. It automatically generates the complete simulation application and insures a lot of coherency verifications.

MOST supports a top-down structured data-flow analysis similar to a SADT description. The intermediate nodes may be systems, subsystems or interface modules, the terminal leafs will be microprocessor code, or a test sequence, or an interface procedure. Every data exchange must be described with MOST, the procedural actions are necessarily introduced by the user into each procedure under MOST control.

When the description is ready for simulation, the complete simulation code is generated and linked to run the simulation.

Figures 3 to 6 present different levels of an application description with MOST.

- Figure 3 represents the higher level with the name 'GVA' of the application.
- In Figure 4, we see the first level structure with the component '.GVA1' which is the code under test and the two subsystems: "ENV" is the .GVA1 environment and represents its inputs and outputs, "\$CTRL" is the observer and models the on-line varification of the .GVA1 + ENV behaviour compared to expected results.
- Figure 5 gives the structure of "ENV" which in this case is made of environment modules.
- Figure 6 is the lower development of one of ENV's modules with the procedures and their starter variables defined.

The data-flow links are input, when they enter the left part of a component, output when they start from the right side, input and output on the top.

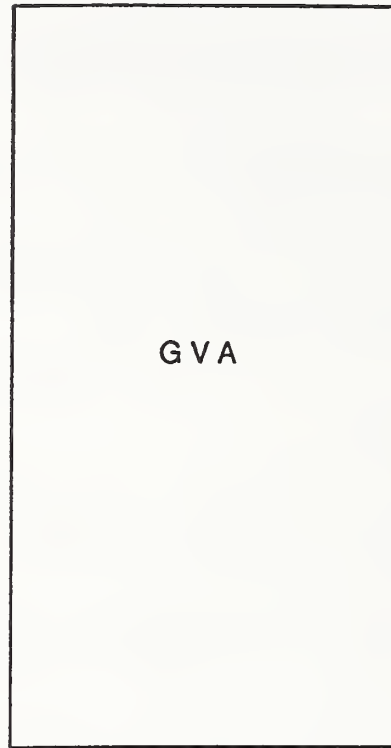


Figure 3: Global structure of the application GVA (Alarm Panel)

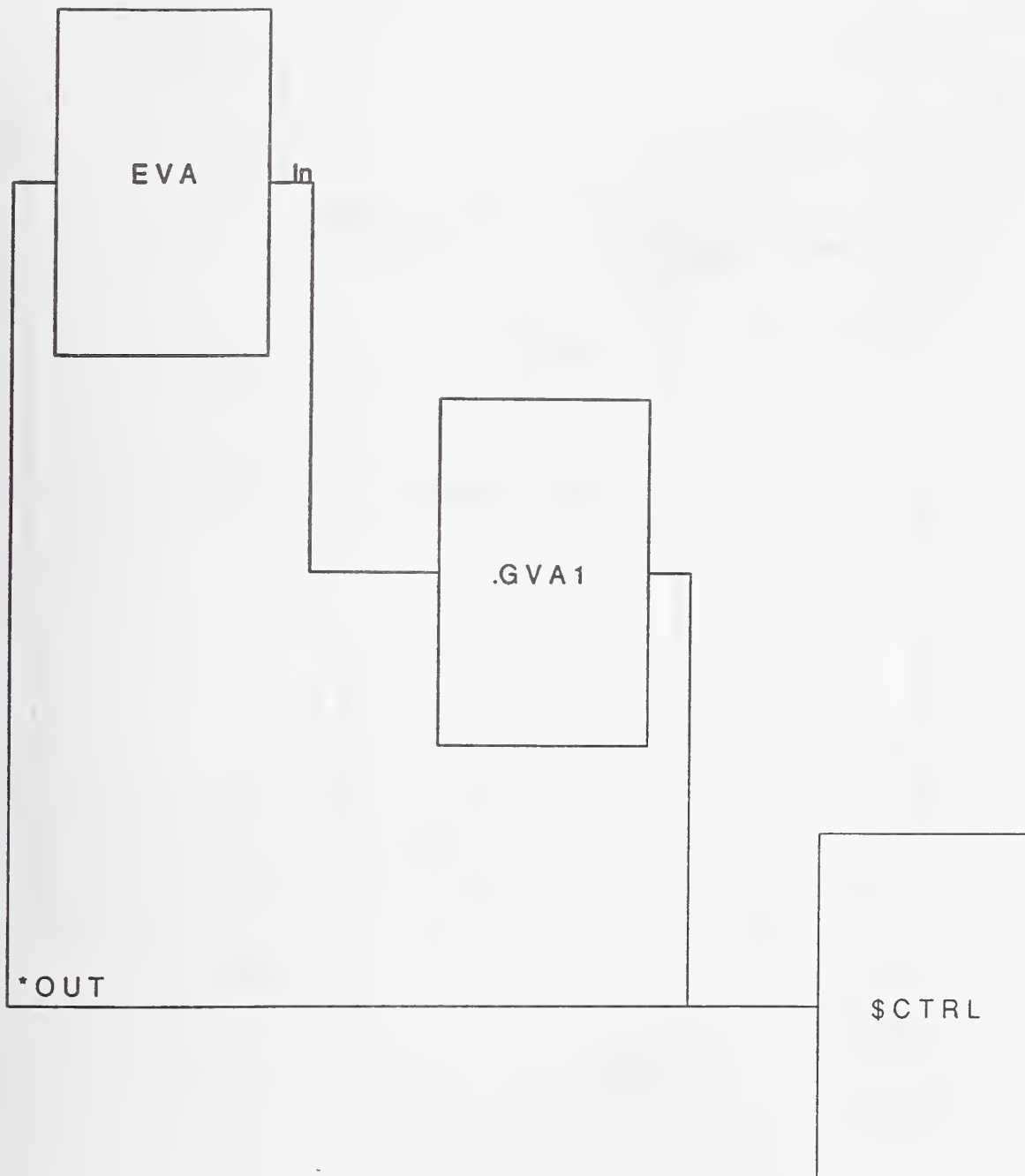


Figure 4: Application GVA is made of a code .GVA1, its environment ENV and the observer \$CTRL.

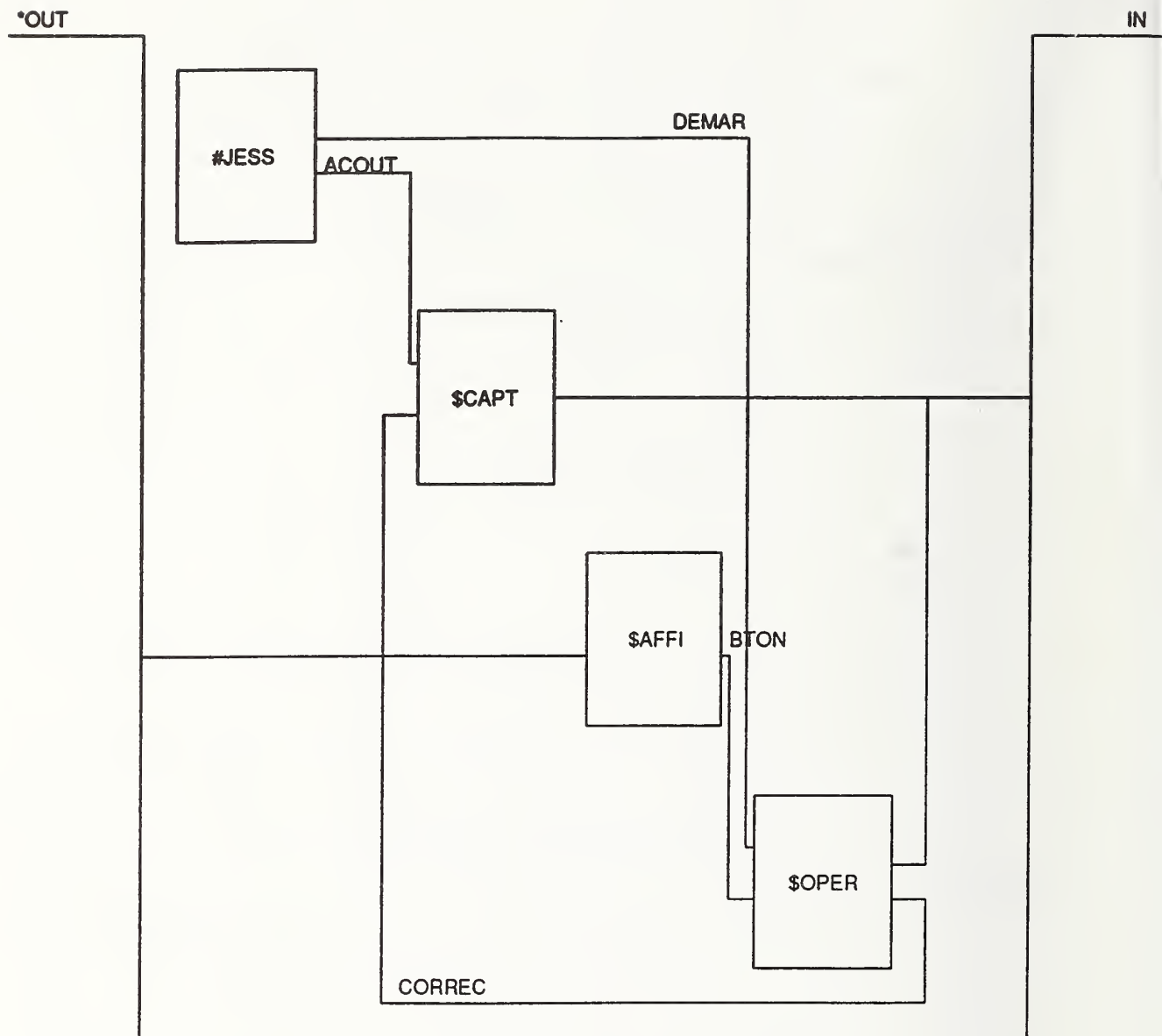


Figure 5: Module structure of ENV

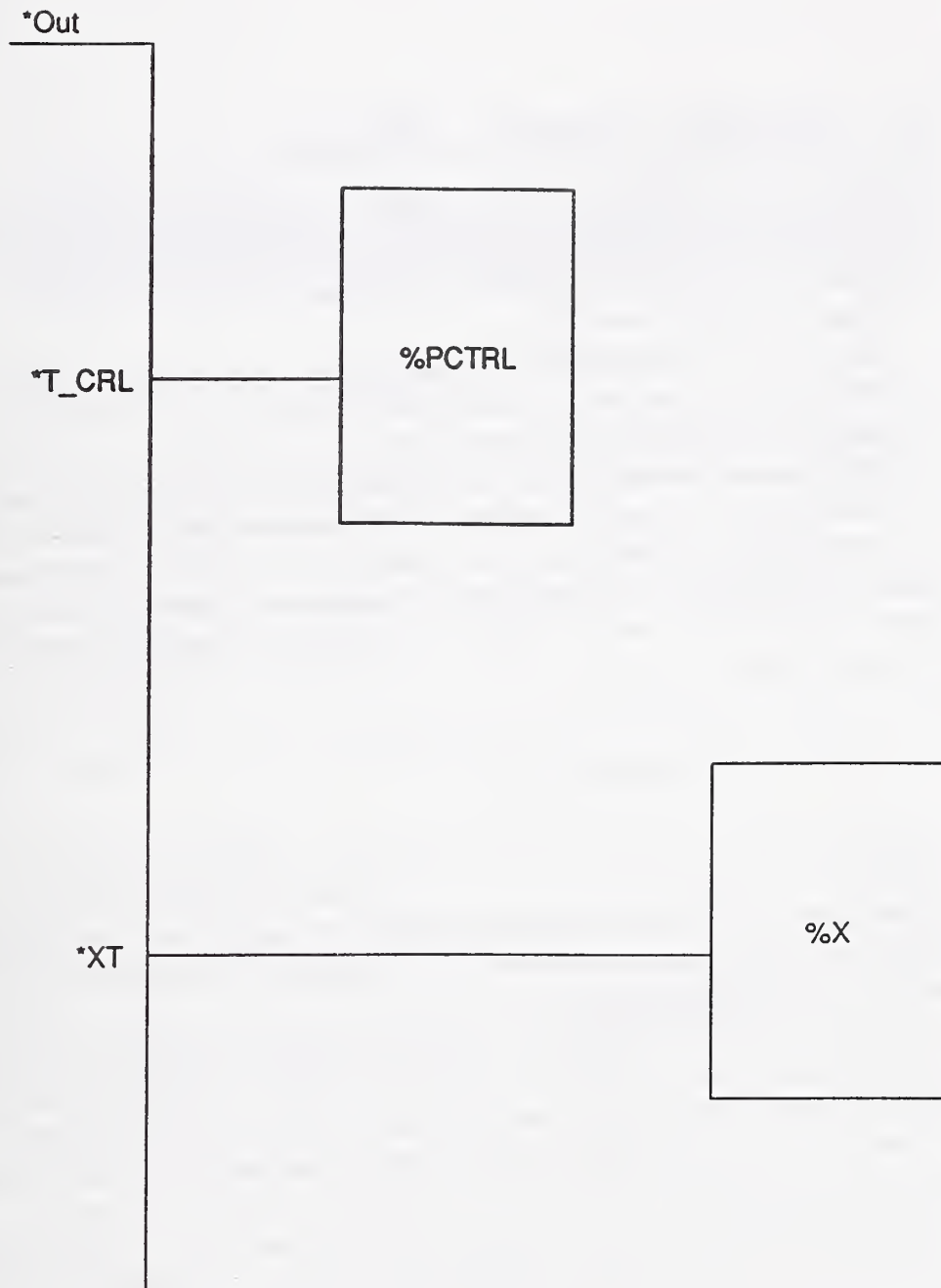


Figure 6: Lower Procedure structure of `$CTRL`

5 Results analysis : TRAS

TRAS tool uses the results of a simulation which is a sequence of temporal evolutions of every variable the user declares to be surveyed. This concerns evolution of values and also traces of passages on surveyed addresses in the emulated micro-processor code.

Temporal windows are determined by conditions on values and dates of evolutions of the variables, and graphical representation of evolution may be interactively requested by the user.

The types of presentations are graphical curbs of values, chronograms, and histograms. In parallel, tables of evolutions may be presented.

From the execution traces of surveyed addresses, TRAS may present the branch coverage of a test campaign, the histogram of number of execution of code of nodes of the control diagram or cumulated execution times on each branch of code.

6 Tests Preparation

CLAIRE toolset does allow the execution of the different types of test. The interface description allows more or less detailed modeling of interactions between the code and its simulated environment and in particular it is easy to verify that the code detects defaults of its interfaces such as wrong input values, wrong synchronizing

The possibility of modifying simply the context of the micro-processor code allows the user to chose parts of the code to be simulated and in an IV&V environment, the user may determine critical software components and apply them a higher level of test. The user may also inject defaults and measure the detection.

The determination and generation of the test sets are not automated in CLAIRE but serveral ways are under study and tools that should assist the user:

- One of them consists of determining by Hoare proof techniques, the logical condition to execute a branch of code. A prototype tool OST-GEN developed at IPSN helps to do it then calculates test sequences fulfilling the conditions;

- A similar method using MALPAS tool allows to express critical conditions, to calculate the logical condition on initial values of variables. Solutions of the logical condition are possible scenarios leading potentially to the critical condition. That method is the theme of a study.

7 Description of the Observer

The observer is modeled by the CLAIRE user according to its needs. It will represent, for example the calculation of variables, theoretical evolutions plus the comparison every n cycles of a calculation. An alarm is sent to the user when the difference becomes greater than the tolerance value. It may also be the expression of a property that must always be fulfilled plus an action when that property becomes false.

The CLAIRE user describes to the observer part of his simulation environment in exactly the same way as the other description environment and it will be generated with the whole application.

8 Conclusion

The first version of CLAIRE toolset was delivered in 1987, it consisted of the OST simulator, in a mono-microprocessor version and with the M6800 emulator. The environment and test sequence description were written in a special Pascal-like language with temporal features. With that first tool, units of the protection system SPIN P4 of PWR 1300MW were analyzed, critical software components were exhaustively tested. The branch coverage of standard test sequences was measured.

Emulators of M68000, In8086 and In8051 were introduced and in 1990 we produced a multi-microprocessor version able to simulate an application with several codes.

The graphical description and Test Environment generation MOST was delivered at the beginning of 1992 and was used to model and simulate SPIN N4 units.

In 1993, we replaced the simulator part with a new one. The environment is now described in the C language and is greatly more powerful. The performances of the simulation are largely increased. The TRAS tool is being

delivered at the end of 1993.

Further development concerning tools to assist the test generation and to automate the verification of properties either during the simulation, by generation of observers or at the phase of result analysis.

6.1.1 Questions: Ms. Anne-Marie Lapassat

QUESTION: HERB HECHT (SoHar, Inc.): What kind of malfunctions or faults have you detected?

MS. LAPASSAT: In fact, on the P4 version of SPIN we detected only very minor faults. We never detected faults in software protection.

QUESTION: DORELLE RAWLINGS (Sorrento Electronics): You talked about virtual time in the simulator. How do you verify that target system timing aspects and order of execution are really adequately represented in the simulator?

MS. LAPASSAT: Yes. I think we are able to model temporal specs with special instructions. In the fourth version of the simulator it was with a special language manipulating the time. We are in a position to model the time actions. With a standard language, we generate C code and we explore every possibility to model any type of interaction of temporal interaction. In fact, we made an hypothesis on temporal interactions and we described that hypothesis. We wanted the code to record that hypothesis, but you can represent any model of time.

6.2 The Risks of Safety-critical Systems: Dr. Winston Royce

The Risks of Safety-critical Systems

Winston Royce
TRW, Inc.

[edited from transcript]

Dr. Royce identified the following six areas of concern with regard to safety-critical systems:

1. Safety-critical systems are implemented in the *wrong languages*.
2. There are *not enough tools* for safety-critical systems development.
3. There is insufficient analysis and distribution of *error measurement* data.
4. There is *no organizational certification*.
5. There is *no people certification*.
6. There is *infrequent graspability* of the full system functionality.

While a newcomer to nuclear safety-critical systems, I have worked on many systems for which, if the system failed, human life would be at some risk.

The first area of concern is languages, which is the starting point for all programmers. All predictions to the contrary language development and language technology are not dead, and C++ and ADA are proof of that. Languages in use today typically require compilers that produce efficient code and that compile reasonably quickly. In addition, current languages and compilers tend to have semantic debugging features and features aimed at reducing error latency rates; however, these languages are the wrong choice for safety-critical software applications.

Choosing a language for safety-critical software development should begin with a different perspective that focuses on the elimination of errors. This perspective forces the developer to compromise efficiency and productivity. There are languages which provide semantic features and inherent self-checking of these semantic features to certify that certain error types are not in the code; however, almost no one uses these languages. The languages we use today do not provide this self-checking.

There are problems with our current language choices. For example, the object-oriented coding world has received little criticism; however, there is a problem with dynamic binding. When a programmer uses it heavily, it makes a program hard to debug, hard to understand and hard to trace. It is even difficult for a programmer to scan his own code and have any chance of tracing through it and determining how the compiler will bind some polymorphic function at run time. For safety-critical applications, we may have to restrict or prohibit the use of dynamic binding.

Another example, is the ADA tasking statement. This is a powerful semantic feature particularly valuable for real-time distributed processing; however, it cannot be turned on in safety-critical applications because it allows time critical errors such as race conditions and deadlocks to develop. Debugging race conditions and deadlock conditions is an extremely difficult job. While ADA has some wonderful error detection features for the development of safety-critical systems, it needs an inherent semantic checker on the use of tasking. While tasking is a desirable semantic, it is also an error-prone one.

These are two examples of very advanced semantics which the C++ and ADA proponents argue for strenuously as virtues of the language which I believe are error prone. Compilers for safety-critical systems should give up some of these features or certify their correctness.

While there are some tools, there are not enough tools because safety-critical software is largely going to have to be tool-based, not human-based. To develop safety-critical software will require the use of tools which restrict the programmer's creativity but prevent and detect errors. Reasonable people should learn to put up with this restriction.

While claiming there are no error measurements is an exaggeration, there is a lack of analysis, dissemination and use of the measured errors. The development process should change based on these errors. Currently, there may be changes on individual projects based on error rates, but not changes on a broader scope. Almost all big software projects measure software errors and require these errors to be formally logged and dispositioned. There is a need for error collection on a wider scope. The Defense Department or the Nuclear Regulatory Commission should collect these errors, and classify them by type so that the industry can know which errors are the most common. The treatment of this type of information should be analogous to the way the federal government keeps records on the cause of death as listed on individual's death certificates. These statistics are used to guide federal research spending, medical school curriculums and insurance company activities. Similarly, government, academia and industry should use knowledge of software error rates to plan which errors to go after in a concerted way through changes in practices and tools. We also need to develop mechanisms for certifying that common errors have been eliminated from software. This is one way to get a step closer to certifiably correct programs.

While I am culturally opposed to certification of people and organizations, I advocate both. If organizations had to keep track of their errors and publish them, I would like to see the Defense Department and the Nuclear Regulatory Commission use these rates when awarding contracts and force the corporate world to worry about this. These rates would be analogous to the SEI methodology evaluation scheme which acquisition commands routinely use. This scheme has brought great attention to methodology issues. I would like to see someone do the same thing with errors.

While certifying people is the hardest part to talk about, there are some programmers, designers and requirements analyzers who are better than others. Not everyone has equal talent and skills

in these areas. If the worry is about errors, then people have to take tests and be qualified just as doctors or lawyers are and the weaker individuals must be weeded out.

The last point is the question of graspability. Errors are higher in systems when the people who use, test, and build the systems cannot fully grasp the system they are building, testing, or using. It's particularly noticeable now with distributed systems. Software designed for one arithmetic register was much easier to understand than software designed for heterogenous networks with upwards of 30 mini-registers requiring parallel or concurrent programming. It is very hard for a very good programmer to grasp the entire system that he is trying to build. This leads to errors. There is a need for system engineering or architectural tools which permit the builder, the program manager, the testers, etc. to understand what is going on in the entire system.

6.2.1 Questions: Dr. Winston Royce

QUESTION: DR. JOHN McHUGH (Portland State University): For over a decade I have been trying to get my hands on the kind of error data that you're talking about. Are you offering to make available to the research and the industrial community TRW's error rate data and error data for their projects? If so, I would like to talk to you about it off-line.

DR. ROYCE: No, I'm not. I'm not empowered to do that, and I hope you understand the sensitivity. I didn't speak to that. To admit you make errors and give the actual statistics, when you're threatened by the fact your error rate may be higher than a competitor, is tough.

DR. McHUGH: This is exactly the reaction that I've run into. I really think that it is unconscionable in safety-critical areas for companies to consider both the process by which they develop software and the results of that process as proprietary. I think the public interest demands that all of those things be made public.

DR. ROYCE: Well, I agree with you totally, John, and while I can't do it, I'll work my level best to provide you the data, or any other group such as you. The question is not that one company do it, it's that all companies do it.

DR. McHUGH: Thank you. It sounds like the regulatory people have an item for their agenda.

DR. ROYCE: They've got the power to make it happen.

QUESTION: GUSTAV DAHLL (OECD Halden Reactor Project): That was a very interesting speech, but I have some comments on this. You're saying that you should go for statistics on errors or faults which were most common. My impression is that the dangerous faults are not the ones which are the most common, the programming faults, but the dangerous faults are the faults inherent from the wrong specification because they repeat throughout the process. These faults don't come out until the product is in use and probably result from a misunderstanding.

DR. ROYCE: I wouldn't disagree with that a bit. I tried to keep my ideas simple and within 25 minutes. But obviously the most common error may be sort of trivial in terms of its consequences. You kind of need a weighing factor that has to do with the importance of the error in terms of the degree of catastrophe as well as its frequency of occurrence. I agree totally with your idea.

QUESTION: DR. JOHN KNIGHT (University of Virginia): Could you expand on your bullet three on people certification, what you think that might involve and how it might be administered?

DR. ROYCE: Well, a potential way is for an acquiring organization--it could be a state, it could be the Federal Government, it could be a body of either the state or the Federal Government--to simply put together some kind of test and ask that programmers pass it and they receive a sort

of certification, and in bidding any of these jobs only certified programmers and designers can be used.

QUESTION: KOFI KORSAH (Oak Ridge National Laboratory): What are the possibilities of making software open to scrutiny, just like you can design a system where you have circuit diagrams in your manuals and they are open to scrutiny; what are the possibilities of making the software also open to scrutiny?

DR. ROYCE: Scrutiny? I thought you meant open as in open systems, which you'll notice I didn't mention, because it's just one more source of error. Open to scrutiny? Of course. I mean every logical step should be open to scrutiny. I don't think eyeballing it is going to help much though, but open to scrutiny in the sense of the acquiring organization, or an independent V&V team, or a user group bringing in their own tools and executing them on the software. Since I do believe it's beyond scrutiny in the sense of eyeballs, code should be entirely open to scrutiny in the sense of test tools.

6.3 Integrated Modeling of Software Cost and Quality: Mr. Kyle Y. Rone and Ms. Kitty M. Olson

INTEGRATED MODELING OF SOFTWARE COST AND QUALITY

Kyle Y. Rone and Kitty M. Olson
IBM Corporation
3700 Bay Area Blvd.
Houston, Tx. 77058-1199

ABSTRACT

In modeling the cost and quality of software systems, the relationship between cost and quality must be considered. This explicit relationship is dictated by the criticality of the software being developed. The balance between cost and quality is a viable software engineering trade-off throughout the life cycle. Therefore, the ability to accurately estimate the cost and quality of software systems is essential to providing reliable software on time and within budget.

Software cost models relate the product error rate to the percent of the project labor that is required for independent verification and validation. The criticality of the software determines which cost model is used to estimate the labor required to develop the software. Software quality models yield an expected error discovery rate based on the software size, criticality, software development environment, and the level of competence of the project and the developers with respect to the processes being employed.

INTRODUCTION

Software cost and quality engineering is a systematic approach to estimating, measuring, and controlling cost and quality. This discipline provides the vital link between the concepts of economic analysis and the methodology of software engineering. The tasks involved in software cost and quality engineering are complex, and individuals with the knowledge and skill required are scarce (DeMarco 1982; Putnam and Myers 1992). The accuracy and consistency of the software cost and quality estimates are often questionable (Kemerer 1987). There is a definite need to improve modeling and simulation of software costs and quality using knowledge-based tools and to improve the accuracy and consistency of the results (Boehm 1987).

Viable software cost and quality modeling depend on a quantitative historical database. Past

experience in managing large software projects, such as the Space Shuttle Primary Avionics Software System (PASS), illustrates that accurate cost and quality estimates based on reliable historical data are essential to software planning and program management. Over the past 17 years the Federal Systems Company (FSC) has collected extensive data from the National Aeronautics and Space Administration (NASA) projects and other software development projects. This historical database of software metrics includes source lines of code, productivity rates, and error rates for more than 250 projects. This data provides the basis for the software cost and quality models (Rone 1990).

SOFTWARE LIFE CYCLE COSTING

Software systems are historically late and over budget. A recent General Accounting Office study pointed out that a majority of the software systems studied sustained substantial cost overruns and serious schedule slippages. Accurate labor estimates based on reliable data for software size, productivity, and error distribution are essential to software planning.

The standard Rayleigh curve models the typical build-up of staff during the requirements and design phases, the peak at implementation, and the tail-off during the testing phase (Putnam and Myers 1992). Figure 1 illustrates the staffing profile for an ideal project. During sustaining engineering, a minimum level of critical skills is required for effective maintenance. This steady-state staffing level forms the support line. It includes critical skills for requirements, design, implementation, testing, and management. The support line is a function of system size, productivity rate, and the unique skill requirements for the software system being maintained. In Figure 1 the area below the support line and above the maintenance tail of the Rayleigh curve represents the capability for new development work.

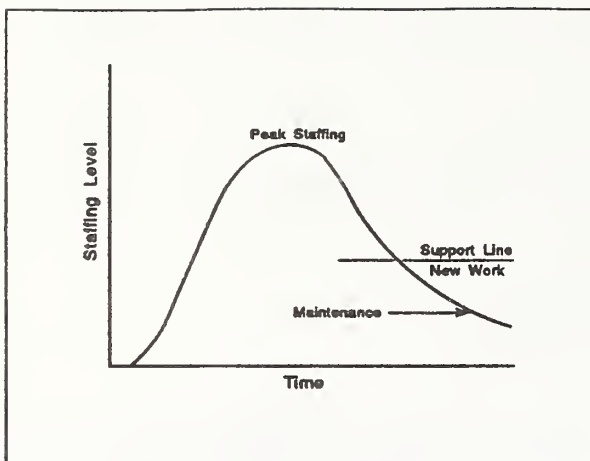


Figure 1. Staffing level modeled as a Rayleigh curve.

As shown in Figure 2 sustaining engineering operational increments also correspond to Rayleigh curves. Each sustaining engineering effort can be modeled as the sum of a sequence of such curves. The sizing and scheduling of new development activities should be planned to provide a stable level of effort as illustrated by the total development line. Software maintenance which handles Problem Reports can continue at a lower support level as illustrated by the total maintenance line. The total development line should not fall below the critical skills required by the project as determined by the initial staffing model.

Figure 3 illustrates the software cost estimation process. Modeling software cost is based on the technique of stepwise refinement. This technique involves decomposing the software system requirements into software functional components. These components are further decomposed into as many independent elements as possible. Decomposition terminates whenever a reused software element or a Commercial-Off-The-Shelf (COTS) software product is identified or whenever the component is decomposed to the lowest level. The software elements and the COTS products are sized and classified according to release, language, complexity, and criticality.

An estimate of software size is critical to obtaining an accurate estimate of the labor required to develop a software system. Size is an important factor that ultimately affects the accuracy of the labor estimate. For example as the size of the software system increases the interdependency among various elements of the software system also increases.

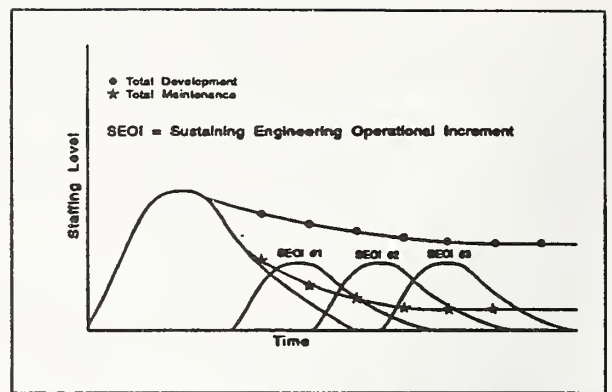


Figure 2. Sustaining engineering effort modeled as a sequence of Rayleigh curves.

Release represents either an incremental product release, a release of the software development environment, or the learning curve associated with the software development process. Language is the programming language in which each software component will be implemented.

Complexity, the relative difficulty of the software function, is an important factor affecting development costs. Some types of software systems are more difficult to develop than others, e.g., development of an operating system compared to the development of utility software. Software complexity is based on three primary factors:

- *Constraint Considerations* - Constraints such as performance, timing, or space are considered a factor in determining complexity if they become "critical." A critical constraint is one which will require functional or logical design trade-offs. This includes user imposed constraints, such as expense and production schedules, as well as installation imposed constraints such as hardware, software, operation scheduling, storage, and standards.
- *Interface Requirements* - Addresses whether intricate interfaces, either hardware, software, or human must be defined.
- *Software Classification* - Addresses software complexity based on the type of software operation being developed: control, computational, device-dependent, or data management.

Criticality is the level of effect of a failure of a software component. Software for certain medical diagnostic or treatment systems, air traffic control, or

the Space Shuttle's PASS must not fail or human lives could be lost. In contrast, an inventory control system should not fail, but the impact of the failure would not result in the loss of human life.

As shown in Figure 3 these inputs--size, release, language, complexity, and criticality--are used by the cost model to generate an estimate of the labor required to develop the software components.

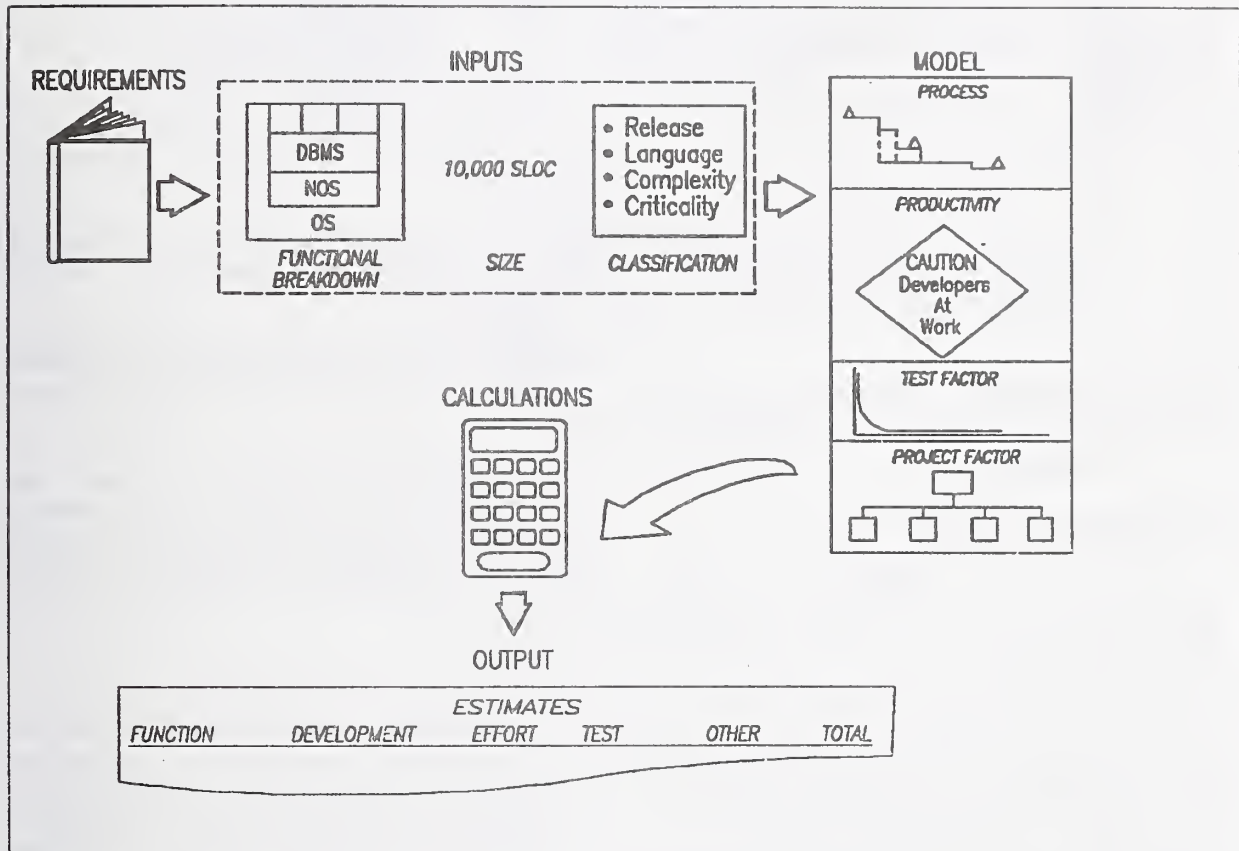


Figure 3. Modeling the software cost estimation process.

COST AND QUALITY RELATIONSHIP

In modeling the cost and quality of software components, cost and quality can be traded against one another. By attempting to minimize development costs many projects simply defer error correction into the product time frame where the cost of error correction is more expensive. To prevent this from occurring a careful balance of product cost versus product quality must be established. The relative trade-off between cost and quality is dictated by the criticality of the software component being developed.

The criticality level determines which cost model is used to estimate the labor required to develop a software system. For example, a software component clas-

sified as low criticality, will incur verification costs and indirect costs which are a relatively low percentage of the overall total development cost. In contrast, a software component which is classified as high criticality, will incur verification and indirect costs which are a relatively high percentage of the overall total development cost. This relationship is illustrated in Figure 4.

Each cost model is associated with a specific product error rate. For example a low criticality cost model may be related to a product error rate of one error per one thousand source lines of code. Similarly, a high criticality cost model may be related to a product error rate of one tenth (0.1) error per one thousand source lines of code.

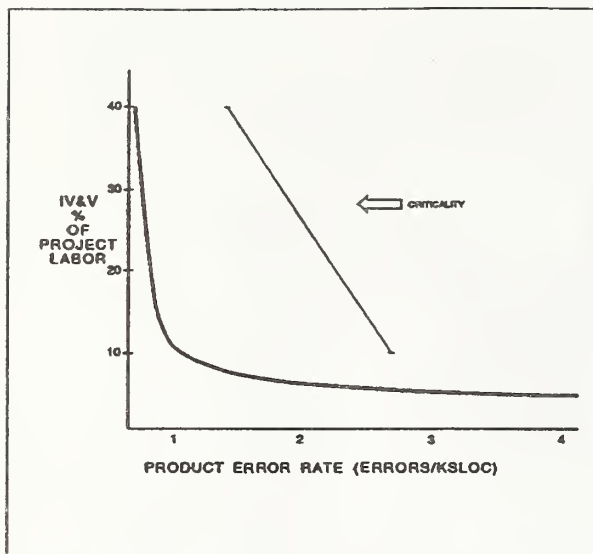


Figure 4. Product Error Rate versus Independent Verification & Validation (IV & V) Percentage of Project Labor

SOFTWARE LIFE CYCLE QUALITY

Since the software development process is complex, ample opportunities exist to make errors. Many of these software errors will be discovered and corrected before delivery. Therefore, it is essential that a project predict an expected error discovery curve. As shown in Figure 5 the standard Rayleigh curve models the error discovery curve of a typical project (Kan 1991). The area under the curve represents the total number of errors inserted into the software system. By comparing the actual error discovery rate to the expected error discovery rate, management can judge whether work is proceeding within expected bounds.

As shown in Figure 6 the system requirements are first decomposed into software functional components. These components are sized and classified according to release, criticality, project proficiency and development proficiency. Project proficiency represents the level of competence of a project with respect to their processes. Project proficiency determines how many total errors will be inserted in the product per one thousand source lines of code. Development proficiency represents the level of competence of the developers with respect to their process. Development proficiency determines the total number of errors that

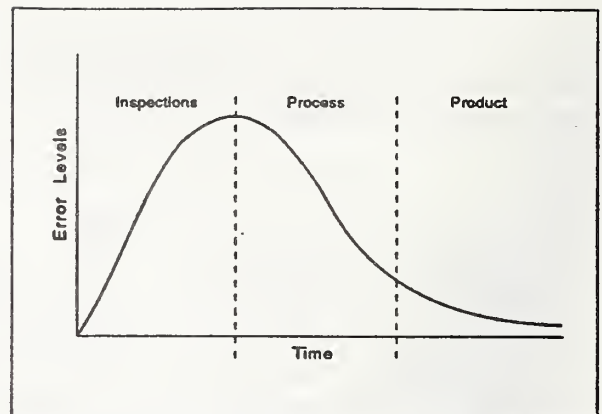


Figure 5. Error discovery modeled using a Rayleigh curve.

will be discovered and corrected early in the development cycle. The criticality level determines the product error rate.

As illustrated in Figure 6 these inputs--size, release, criticality, project proficiency, and development proficiency--are used by the quality model to generate an expected error distribution pattern of early, process, and product errors.

SUMMARY

Software now controls not only the nation's telephone communication system, but everything from sophisticated medical diagnostic equipment to the world's financial systems. As software systems increase in both size and complexity, the cost of developing these systems rises. With ever increasing size and complexity software errors become inevitable. Reliable software on time and within budget is contingent on accurate, timely software cost and quality estimates.

Software cost and quality models provide the capability to quickly generate estimates for diverse types of projects. Changes in assumptions such as size, complexity, or criticality are easily factored into the cost and quality estimates. These estimates can be phased across time to determine if the staffing profile or the error density is too steep at certain points in the process. Integrated software cost and quality models provide the information needed to effectively plan, manage, and control the software development process.

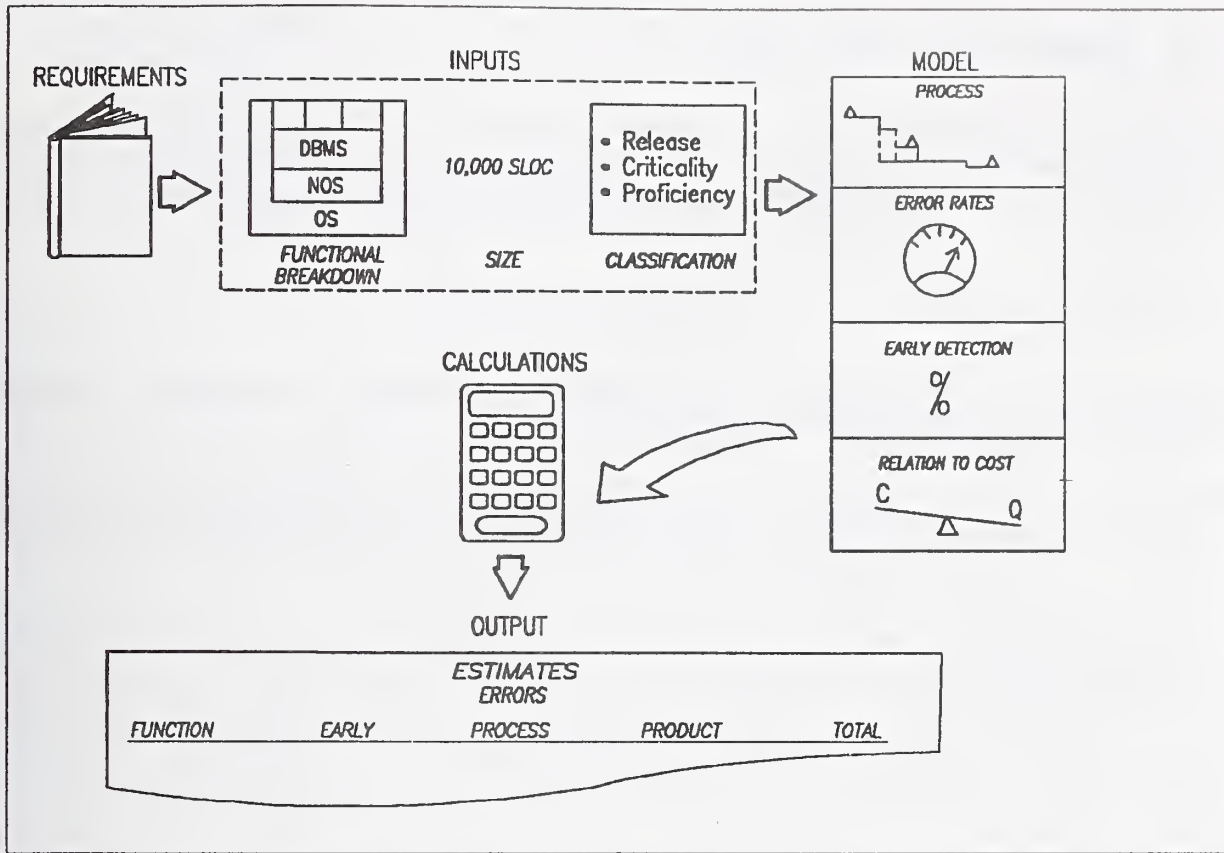


Figure 6. Modeling the software quality estimation process.

REFERENCES

- Boehm, B. W. 1987. "Improving Software Productivity." In *COMPUTER*, vol. 20, no. 9 (Sept.), 43-57.
- DeMarco, T. 1982. *Controlling Software Projects*, Yourdon Press.
- Kemerer, C. F. 1987. "An Empirical Validation of Software Cost Estimation Models." In *COMMUNICATIONS of the ACM*, vol. 30, no.5 (May), 416-429.
- Kan, S. H. 1991. "Modeling and Software Development Quality." In *IBM Systems Journal*, vol. 30, no. 3. 351-362.
- Putman, L. H. and W. Myers. 1992. *Measures For Excellence*. Yourdon Press.
- Rone, K. Y. 1990. "Cost and Quality Planning for Large NASA Programs." In *Proceedings of the Fifteenth Annual Software Engineering Workshop* (Nov. 28-29). NASA, Greenbelt, Maryland.

6.3.1 Questions: Mr. Kyle Y. Rone

QUESTION: MICHAEL NOVAK (Combustion Engineering): I think you talked a little bit about error conditions, error analysis, that we wouldn't want to release software for, but I'd like to know do you use any of these methods to determine when to release your safety-critical software?

MR. RONE: Absolutely.

MR. NOVAK: And, if you do, how do you use them, what levels do you assume, and what confidence do you have in releasing software in this way?

MR. RONE: We collect this data. We collect the cost data and the quality data, schedule data, and once a month, as we go through our processes, we look at that data and make sure that it's consistent. If you look at process models and if you do them correctly, there is a process with an input and an output, and there is also entry and exit criteria for each of those process steps. We use this information. We mark special milestones in there and we call them "control points." When we hit those points we must stop and examine the current entry and/or exit criteria and we use that information to either shut down the process or turn it loose. In particular, we do that at major customer milestones like when the software is turned over to independent verification and then when it's turned over to the customer. The customer is well aware of all of these models and this data. We sit down with the customer go through the analysis of the data and we make a joint decision with them as to whether to proceed with the process or not. So, we very much make use of this data, and it's saved us from disaster many, many times.

MR. NOVAK: Could you state what your confidence is as a result of using these measurements, what confidence you have in these measurements?

MR. RONE: Our confidence level is as high as it absolutely can be. In fact, when we go to the Cape and we sit down with all of the people involved in developing the shuttle, a meeting is held--and it's videotaped by the way--where everybody, every developer of each component, goes through and declares their component ready to fly. And so that's the kind of confidence level we have. We stand up and state our part is ready. We've looked at the data. We've looked at the analysis of errors that we've found, and we're confident that it's ready to fly. Other than that level of confidence, I don't know what you can express.

6.4 Software Reliability for Safety-Critical Applications: Dr. William Everett and Mr. John Musa

Software Reliability for Safety-Critical Applications

Bill Everett and John Musa
AT&T Bell Laboratories

In this talk, we address the question "Can Software Reliability Engineering measurement and modeling techniques be applied to safety-critical applications?" Quantitative techniques have long been applied in engineering hardware components of safety-critical applications. We have seen a growing acceptance and use of quantitative techniques in engineering software systems but a continuing reluctance in using such techniques in safety-critical applications.

The general case posed against using quantitative techniques for software components runs along the following lines: safety-critical applications should be engineered such that catastrophic failures occur less frequently than one in a billion hours of operation; current software measurement/modeling techniques rely on using failure history data collected during testing; we would have to accumulate over a billion operational hours to verify failure rate objectives of about one per billion hours.

However, a case can be made for using quantitative techniques. It proceeds along the following lines: isolate safety-critical functions; establish the level of processing associated with each safety-critical function; use test compression techniques along with measurement/modeling techniques to verify objectives. Cases exist where we can verify very small failure rate objectives in a reasonable period of test time. These notions will be expanded on in the following.

First, for many safety-critical applications, it is not the entire application that is safety-critical but only certain functions that are safety-critical. For such applications, safety-critical functions should be isolated to a few software modules and care taken to prevent interactions with other modules in the application. We can then focus on measuring/modeling the reliability of these few modules to insure objectives are being met.

Second, we can take advantage of differences between "operational time," i.e., the time the application is active; and "processing time," the time spent processing instructions in the software module. It is a module's "processing time" that is a measure of the stress imposed on a software module that may lead to failure.

For some applications; the processing time associated with critical functions may represent a small fraction of the operational time. This opens the door for "test acceleration" where we can simulate the passage of a large amount of operational time in a reasonable test interval. Some

examples are discussed, in particular, the testing of planetary probe navigation/maneuver software.

Although the above example is not intended to be an all-encompassing approach, it does show the possibilities for quantitatively certifying that high reliability levels have been attained for specific critical functions.

Let's contrast the above with the current alternative commonly followed for certifying safety-critical software. This approach stresses the use of particular processes and practices in developing safety-critical software. There are problems with this approach. First, we don't have a quantitative understanding of the relationship between these processes/practices and the resulting software produced. Second, software engineering is in a state of flux with new processes and practices emerging. Again, we do not have quantitative techniques for assessing the affect of new processes/practices on the reliability of the software produced.

So where does this leave us????? We propose that in the future we need to invest more in applying software reliability measurement and modeling to safety-critical software; to supplement the current approach of certifying the quality of the development process. We need to establish a better climate that encourages industry to (1) apply more reliability measurement/modeling techniques, (2) stimulate the emergence of new measurement/modeling techniques. Industrial experience in applying measure/modeling techniques will stimulate their refinement and enhancement. Unfortunately, the current climate of the economy and standards for safety-critical software do not contribute to either (1) or (2) above. The current wording in standards gives no incentive to companies to invest in reliability measurement/modeling methods for software.

6.4.1 Questions: Dr. William Everett

QUESTION: DR. LANCE A. MILLER (SAIC): Yes, Bill. In explaining your cumulative error curves, presumably that's performed on a single version of a program. How do you feel about the assertion by some that once one of those defects is repaired you're dealing with an entirely different computer program, so any historical data now is invalid, so now you must perform this error check over again from the start?

DR. EVERETT: Some of the reliability models and some of the methods we use take into account that error repair is not perfect. In fact, we put it in and you can put it back in. In terms of throwing out all the data which you have, you've got a lot of software there that you've already exercised in a number of different ways, you know, that may not depend on the particular area in which you made that error repair. So, I think throwing all that data away is wrong. I think you're really doing an injustice too because especially in the safety-critical area we rely a lot on using our history data because we have so few failure occurrences.

DR. MILLER: I guess that's the point. It's not my decision especially, but the point is you don't know what areas of the code you've affected by making a change. You can make some assumptions. But the conservative approach is you don't know so you need to start over. Do you disagree with that?

DR. EVERETT: I disagree with it.

QUESTION: GREG MILLER (Idaho National Engineering Laboratory): There seems to be a magic number of 10^{-4} failures per demand brought up as a maximum reliability that one might be able to use by following a very well-defined prescribed process. What I'm getting at is that some have advocated that following your process allows you to assume that you have reliability at least of 10^{-4} .

DR. EVERETT: That's news to me and I don't know how you can do it right off. Maybe the 10^{-4} , I have seen that come around because of this testing figure. How much can you test in terms of processing and how many test hours do you accumulate to verify a particular project? Once you get up to about 10^3 or 10^4 hours you're getting into, what a lot of large projects dedicate for a testing cycle, 3 to 4 or 5 months, so maybe that's where the number came from. But magically saying, "If I follow this process the most I can hope to get is 10^{-4} ," unless you can associate a model that shows you where that measurement came from, I don't know whether you can say it.

QUESTION: WILLIAM D. GHRIST (Westinghouse Electric): Two questions basically. One is that I don't understand where the correlation is between the probability of errors in the software, the probability of detecting them, and just plain ordinary execution time. If you execute a piece of code one time in a particular circumstance and it doesn't fail, it doesn't make any difference whether you execute that for one more time or a million more times it's not going

to fail. The real idea is that the failure is only going to turn up if you execute it in different ways, not for a specific amount of time.

DR. EVERETT: That's true. And one of the assumptions we've made, and especially what we do in a lot of our commercial projects, is we simulate operation, placing telephone calls. We'll select randomly particular calls being made. Now, one particular call may go through the code and execute in exactly the same way, but that call, interacting with another possible call, is where we can possibly begin uncovering failures that result from the interactions between things going on.

MR. GHRIST: But it's not clear to me that that's correlated with execution time though.

DR. EVERETT: Again, if you look at the software and the stress that we can place, we can make assumptions about the way we're executing it in terms of forcing us to move down certain paths and to encounter a particular fault and encounter it under just the right conditions that cause a failure then that's what a lot of this is based on.

MR. GHRIST: Yes. You didn't seem to address at all the idea that experience of testing code is one of the less effective ways of finding errors in it. Inspection and analysis generally find most of the errors in the code, not the testing of it, and where does that fit into this?

DR. EVERETT: That's been our observation too. There has been too much testing to analyze, let's test quality at the end, and in our practice we're trying to move that further back into the process. One of the things we have found is to at least orient our testing toward measurement, making testing more of a measurement process, rather than a bug finding process. This is forcing us to now go back and set objectives and define usage of the product in terms of things we call an operational profile. It is changing behavior. We find our testers now are going out and visiting customers' sites in order to characterize operational behavior and actually participating in setting reliability objectives because they need this information to test the product.

**6.5 Software Configuration Management for Safety-critical Systems:
Mr. H. Ronald Berlack**

**SOFTWARE CONFIGURATION MANAGEMENT FOR
SAFETY-CRITICAL SYSTEMS**

H. Ronald Berlack
Configuration Management International

presented to:
Digital Systems Reliability and Nuclear Safety Workshop
Sept. 13-14, 1994
Rockville Crown Plaza Hotel
Rockville, MD

Abstract

This paper describes how the Software Configuration Management Process (SCM) can contribute to the successful accomplishment of development and maintenance of Safety-critical Software in the Nuclear Reactor System's environment. SCM is the management of the software product as it evolves through development and operates in a designated system. SCM acts as a communicator for the development/maintainer activities through the process of identification of the software product's requirements, design and associated interfaces, establishing baselines, managing changes, reporting status and performing audits.

INTRODUCTION

CURRENT ISSUES:

The evolution of analog processing of signal and sensor data in the nuclear reactor system has caused a valid concern that "...the software and hardware for...computer systems could be vulnerable to design and programming errors that could lead to safety-significant common mode failures" [1]. The basis of this concern is derived from the use of a collection of single parameter configurations for analog Instrument and Control Systems (I&C) versus the use of shared computer systems employed in digital I&C systems. In analog systems, the means to data flow is "hardwired". In digital systems, data flows from many sources of signals and sensor information to a central or shared computer for processing and analysis. The eventual concern for sharing is one of reliability and that the use of this system "can result in a design that has the potential to propagate common mode failure of redundant equipment. The loss would be greater than analog system failures" [1].

The other key concern is that software errors resulting in common mode failures can defeat all of the redundant channels in a protection system even if there is a minimum of shared systems. [1].

The goal, then, is to limit the probability of the occurrence of common mode failures associated with digital computer technology and limit the external loss of functions in the monitoring, control and safety systems. [1]. This is a goal that can be helped by the employment of a Software Configuration Management (SCM) process that is promoted by top management and accepted by the developing and maintaining personnel on a project or at a nuclear site.

The SCM process cannot solve these concerns, but an SCM process with appropriate management authority and acceptance can be a valuable aide in assuring that the goals as stated have been achieved.

NRC IMPLEMENTATION OF THE SCM PROCESS:

The NRC has not ignored the need for SCM nor the fact that it is important to the success of any project. In January 1983, N.P Wilburn, et al published the Guidelines- Software Configuration Management, HEDL-TC-2283 for the Hanford Engineering Development Laboratory [2]. The ASQC, Energy Division is currently preparing a Configuration Management Standard (DRAFT 8 May 1993) in "terminology that can be applied commonly throughout the energy industry" [3]. The ANSI Standard, CRITERIA FOR DIGITAL COMPUTERS IN SAFETY SYSTEMS OF NUCLEAR POWER GENERATING STATIONS, P-7-4.3.2, draft 8, cites the SCM requirement in para 5.3.5. [4] This paragraph also states that SCM shall be performed in accordance with ASME Publication 880, Software for Computers in the Safety Systems of Nuclear Power Stations, 1986, clause 7.3, has a strong statement for configuration control.[11]. The appendices of Publication 880, provide excellent check lists for reference in the performance of SCM. Additional guidance is also cited in NQA -2a-1990, for IEEE Standard 828-1990, Standard for Software Configuration Management Plans (ANSI). [6] Finally NIST

Special Publication 500-204, Sept 1992, High Integrity Software Standards and Guidelines, D.R.Wallace, et al, cites the requirement for SCM in section 2.6.3. [7].

CONCLUSION:

The above citations certainly indicate that the need is recognized and the requirements have been and are currently being established and promulgated to the energy community. The remainder of this paper will be to show how these requirements can and should be implemented.

My thanks to Ms Linda Roy of MAC Technical Service Company for her review of this paper and for keeping it within the bounds of editorial integrity.

II

SCM OBJECTIVES, EXPECTATIONS & ASSESSMENTS

OBJECTIVES:

A fundamental objective of SCM is to COMMUNICATE to ensure that all people on a given development or maintenance team know:

- What is to be built, tested and delivered,
- What is being built, tested and delivered,
- What was built, tested and delivered

In addition SCM ensures that the software product is designed built, tested and delivered/maintained as specified in "customer" requirement specifications; that the design requirements can be traced to the final product, upward and downward; that changes are managed in an efficient cost effective and timely manner; and that the final product is maintained and supported in the user's environment.

SCM is successful if the SCM activities are initiated at the start of a development or maintenance task and that planning for maintenance is started early in the project. In addition, management support is paramount to success through delegation of authority to the SCM Activity and by providing the necessary human and physical resources to do the job required.

Experience indicates that success is also enhanced through a central focal point for SCM at a high management level in order to represent upper management in all matters relating to the SCM process. This includes directing the implementation of the SCM standards and related activities, disseminating information, and providing training.

The model for these activities are illustrated in Fig 1.

EXPECTATIONS:

A recent survey of Software Engineers by the IEEE Working Group developing a Master Plan for Software Engineering Standards [8] resulted in a list of user expectations for the Configuration Management process. These are listed in Fig 2 and should be valuable guidance to those responsible for implementing SCM and for resolving the issues and problems facing the NRC. The ultimate objective and expectation is to reduce the percent of software errors found in the field, illustrated in Fig 3, to zero!

SW ENVIRONMENT MODEL

THE CM FUNCTION

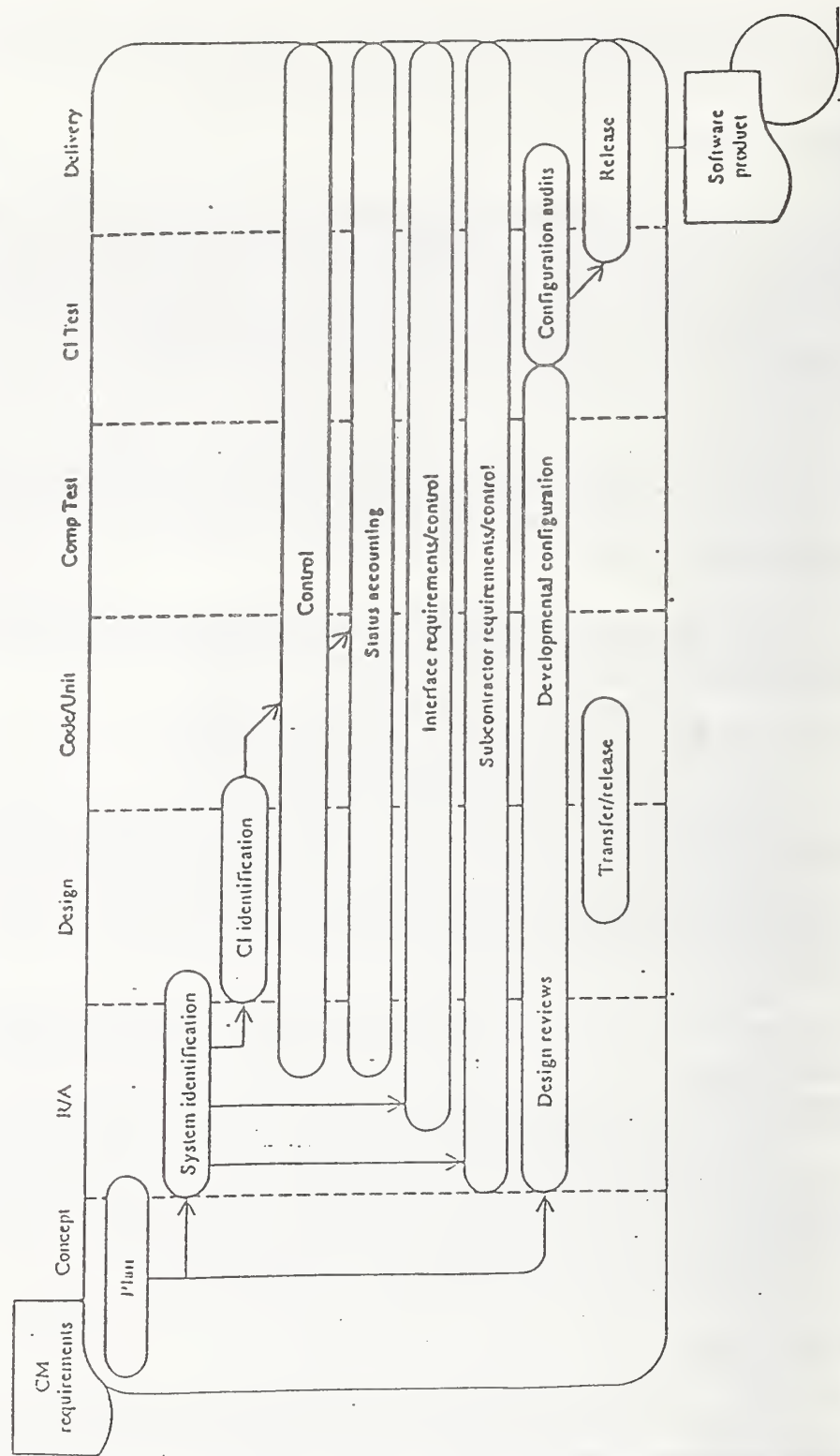


FIGURE 1

USER EXPECTATIONS OF SOFTWARE ENGINEERING STANDARDS CONFIGURATION MANAGEMENT PROCESS OBJECTIVES

THE CONFIGURATION MANAGEMENT PROCESS INSURES THAT:

- a known (identified and documented) and controlled configuration for delivery is provided
- the software in production is consistently replicated or recreated
- consistency and repeatability are achieved during activities such as software builds or testing
- a known point for review, status assessment or application of changes is provided
- change requests receive attention through a specified review process and only authorized changes are implemented and recorded
- evidence is provided for approval of the software before release to a user by appropriate parties
- secure physical control is maintained for all software (including associated data and documentation) regardless of media
- history of changes and modifications is maintained and accessible as required by developers and users
- interfaces are identified, documented and controlled during the life cycle
- subcontractors are monitored and controlled in their performance of CM

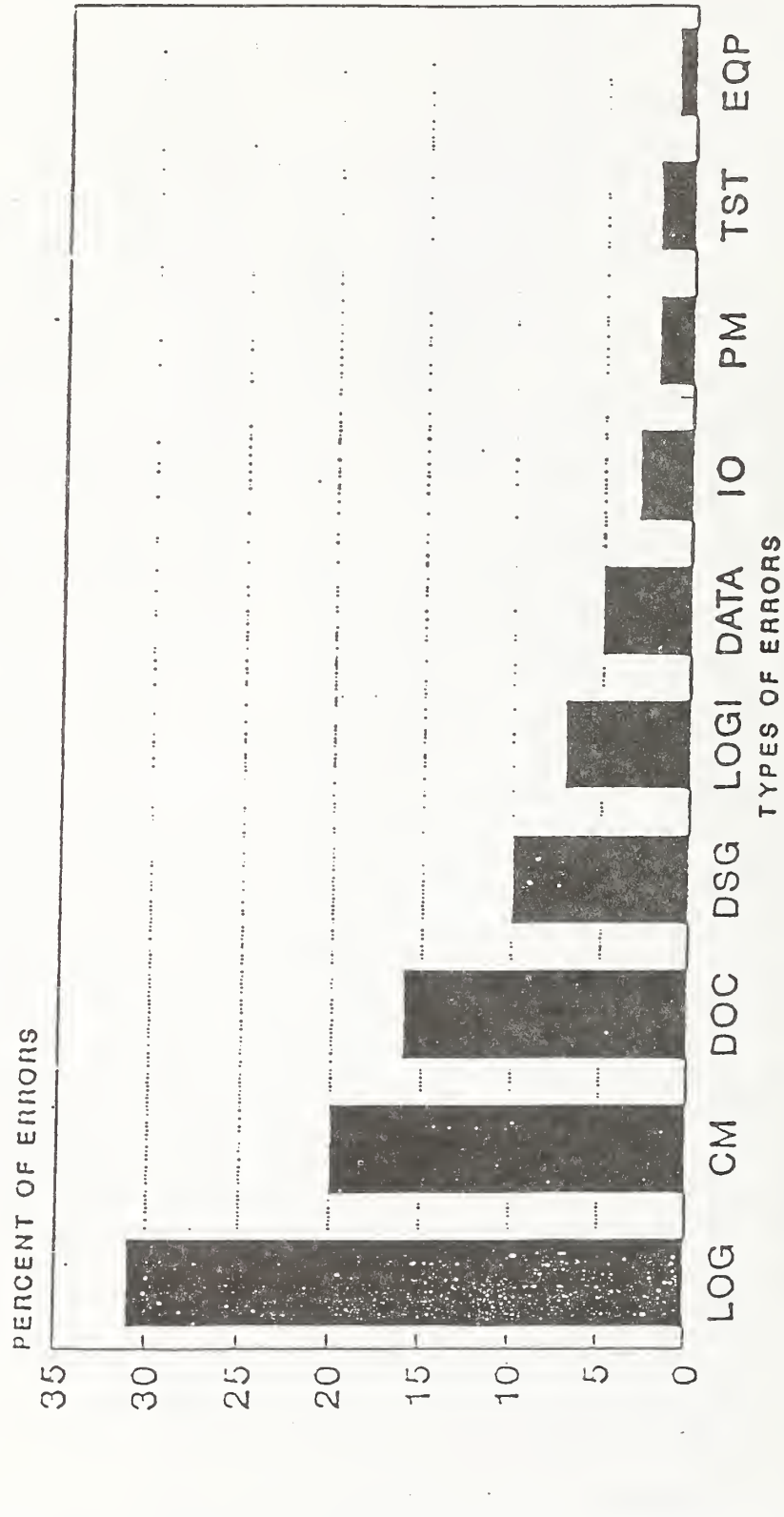
REF: MASTER PLAN FOR SOFTWARE ENGINEERING STANDARDS - IEEE 1993



CONFIGURATION MANAGEMENT
INTERNATIONAL

FIGURE 2

SOFTWARE ERRORS FOUND IN FIELD TYPES OF ERRORS



CMI SWERR2

FIGURE 3

ASSESSMENTS:

During the past several years, the Software Engineering community has responded to the criteria for capability and maturity established by the Software Engineering Institute to develop reliable, quality, fail safe software. This effort was concurrent with criteria established by the ISO/IEC TC176, Quality Committee, as ISO 9000, with ISO 9000-3, a guide for software development.

In order to avoid headlines such as appeared in the news papers on 29 June 1993, "SPACE SHUTTLE COMPUTER SAFETY QUESTIONED - NASA engineers fail to adhere to required standards for software that operates the spacecraft, study says" [9], Studying the assessment criteria and performing an assessment will help those with development responsibility to determine their level of capability and maturity and help them take corrective action to overcome the failure mode issues facing them today.

The High Integrity Software Standards and Guidelines document, referenced earlier, is also an assessment approach. Here, several standards most germane to the nuclear safety problem are reviewed and assessed for their value in answer to the following questions:

"Do the requirements of the document provide assurance of the nuclear safety system software developed, maintained and operated according to these requirements.....?"

"Will the requirements of the document provide NRC auditors with enough information to verify that the vendor product is in compliance with the requirements....."

In order to avoid the above referenced headline, project managers should ensure that the applicable standards they employ, are true, correct, accurate and compliable so that software engineers will not/cannot avoid adhering to them.

III

CURRENT SCM STANDARDS AND GUIDES

There are a number of standards and guides that have been published for use in the government, academia and the commercial world in addition to the ones already mentioned. And although the ASQC writers state that these documents " appear to be prescriptive and too oriented [towards other processes] to be adaptable to the energy industry [3], I submit that they should be reviewed and applied where applicable in order to take advantage of the expertise and experience that went into their development for whatever process.

- IEEE Std 1042, Standard Guide to Software Configuration Management
- ISO/IEC/JTC1/SC&/WG8/CD 12207-1, ISO Standard, Software Configuration Management (DRAFT)
- ISO/IEC/TC176/WG14, DIS ISO 9004-7, Guidelines for Configuration Management

- ISO 9000, Quality Management
- Mil Std 973, Configuration Management
- Mil Std 2167A, Defense Software Development
- Mil Std 498 (DRAFT), Software Development and Documentation
- NASA Std, 2100-91, Software Engineering Documentation Standard
- NASA HBK SEL 84-101, 1990, Managers Handbook for Software Development
- FDA, Good Manufacturing Practices (GMP)
- DOE Order, 5480.CM, Operational Configuration Management Program
- DOE Std, 5489.CMXX (DRAFT) Implementation Guidelines to Operational Configuration Management Program Standard

IV

DEFINITIONS AND ABBREVIATIONS

SAFETY- CRITICAL SOFTWARE - Software that falls in to one or more of the following categories (1) software whose inadvertent response to stimuli, failure to respond when required, response out of sequence, or response in combination with other responses can result in an *accident*. (2) software that is intended to mitigate the result of an *accident* (3) software that is intended to recover from the result of an *accident* [9].

SOFTWARE HAZARD - A software condition that is a prerequisite to an accident [9]

CONFIGURATION MANAGEMENT - A process which ensures that the technical requirements are clearly defined and controlled throughout the development and acquisition process and that the acquired products satisfy the system's technical and operational requirements. The process consists of five principal activities. A sixth can be added for Subcontractor control:

Configuration Identification - selection and identification of items which identify, describe or define the configuration's characteristics for inclusion in the project baselines. This activity includes provision for the unique identification of the items, their interfaces and associated documents.

Configuration Control/Change Management - controlling and managing changes to the configuration to ensure technical and operational requirements are clearly defined and controlled. Includes analyzing the technical, cost, and schedule impacts of a proposed change and affected documents as well as the review and approval of a duly constituted review board (Configuration Control Board - CCB).

Configuration Status Accounting - recording and reporting the implementation of changes to the configuration and its identifying documents.

Configuration Audits and Reviews - verifying that any item selected for CM meets the functional and/or physical characteristics set forth in technical documentation and the conduct of technical reviews to establish points of departure and approved baselines.

Interface Control - defining and controlling the physical and functional interfaces

Subcontractor/Vendor Control - ensuring that subcontractors perform SCM in accordance with customer requirements and that vendors supply sufficient documentation to establish documented identification and provisions for notification of change as agreed on before procurement of the vendor's product.

(Adapted from ASQC Configuration Management Standard sect. 3.1 [3])

V

SAFETY- CRITICAL SCM ACTIVITIES

The SCM process's activities and tasks cannot provide the solution to the design and development problems relating to digital systems and the probability of common failure. Performing the process can, however, provide the ingredients for ensuring that a reliable and safe software product is developed and maintained. The following activities, if well supported and adhered to, can help resolve the vital issues.

PLANNING:

The preparation, agreement and issue of a well defined Software Configuration Management Plan [SCMP] must be accomplished prior to or at the very start of the development/maintenance phase. The SCMP is a means for communicating who is responsible, what activities will be performed, when these activities will be performed, the output from these activities and the expectations that users can look forward to in performing their own tasks. The Project/Maintenance Manager is responsible for the SCMP's implementation.

The SCM activities are Identification, Change Management, Status Accounting, Audit, Interface Control and Subcontractor Control. All activities are initiated at the start of a software life cycle except for Audit, which is held at a point near the end of the cycle. The output of the SCM process are the reviewed and approved specifications and standards, established baselines, with approved changes incorporated, controlled interface and accepted and approved subcontracted products. Status reports and query capability of all actions and transactions provide the status and results of these activities and subsequent products. The final configuration audit will confirm that all of the expectations listed in Fig 2 are realized or indicate that corrective action is needed

The SCMP, using IEEE Std 828-1990 [6] as a guide, must be very specific in how the SCM process will be performed by listing the tasks associated with each activity including tasks, functions, responsible parties, time frames, review authority, release or incorporation authority, IV&V requirements and provisions for maintenance and operation. As an example, the SCMP should describe the minimum requirements for system and software documentation that includes the who, what, when and where for a System Specification, System Operators Manual, Software

Development Files, Software Requirements Specification, Software Development Plan, Software Design Document, Software Test Description and Software Product Specification. These documents will provide the structured and approved baselines throughout the software product's life cycle.

The SCMP should be consistent with the criteria of the definition for safety critical - section IV. Final approval of the plan should be by consensus of the project leadership and authentication of the Project Manager.

CONFIGURATION IDENTIFICATION:

It goes without saying that one cannot make changes to something that has not been identified. Configuration Identification will support safety critical software development by serving as the cornerstone for effective SCM performance. Well documented and uniquely identified documentation and code insures efficient traceability from bottom to top and top to bottom. Performance of this activity insures that all software elements are in place to be able to map out the impact of a change on software documents and related code. It also provides the V&V information necessary to audit the release of code into the system or incorporation of changes into an existing system.

Most important part of Configuration Identification are the specifications or documents that define the requirements - functional, physical, interface, test and qualification; the design criteria in accordance with the requirements; the test plan, test descriptions, test procedures, and subsequent test reports; and the version description document along with the product specification - what was done. SCM insures these documents are concise, contain the required content, are consistent, traceable and are closely reviewed by project experts for correctness prior to their approval or conditional approval for further development of the software.

The review and release documentation is also dependent on use of universally agreed on standards for preparation of these documents. Such standards may or developed by the NRC or the standards are adopted from the IEEE, NASA, NSA, NIST, DOD, DOE, etc.. All of these organizations have grave safety critical concerns. SCM assures that one universal set of documentation is adhered to for ease of communication and commonality of terminology by writer, reviewer and approver. The use of prototyping is also a valuable tool for ensuring a good "try before buy" policy especially for those software elements considered safety critical. Prototyping the design, a peer review by a few good experts, documenting it and placing it under configuration control by SCM will go along way for total assurance of the development of safe software.

Software Configuration Baselines are established from approved specifications and documents as a result of project scheduled technical reviews managed by SCM. The definition, requirements or functional baseline defines "what is required of the software product" The Design or Allocated baseline describes "how to meet the requirements". The Product baseline describes how the product was built, implemented, coded, etc. Baselines are a great benefit to good, efficient requirements and change impact traceability. SCM provides a total picture of the software product through a software hierarchy showing the parent-child relationships of the software elements from top to bottom. Good baselines enable efficient and accurate impact analysis that will insure that safety factors have been accounted for and have or have not been impacted by a given change.

SCM also assures that documents residing in a well controlled baseline have less chance of violating the safety elements in the software product.

SOFTWARE CHANGE MANAGEMENT

Although software change management - change control - is an activity of the SCM process, it is also a process in its own right. A process within a process. Change Management as performed by SCM requires a strong and authoritative baseline, a good efficient status accounting system and a well-defined audit plan that will insure the expedient and validated incorporation of approved changes.

Change Management requires a single point of authority within the SCM process functional entity and an SCM change management/processing team that enables expedient processing of all changes received, via manual or automated means. The authority, normally invested with SCM, is responsible for maintaining the provisions of the SCM Plan and the development of a procedure that will enable the over all change process to proceed in a flawless manner.

Change Management policy and procedure starts with top management to be implemented by SCM and those who submit, review, incorporate and validate approved changes.

SCM insures that initiation of a change request must be easy to understand and to handle. For classification of the changes, a configuration control board (CCB) is required. Analysis for criticality, impact on all elements of the software product and coordination among impacted organizational entities is the responsibility of software engineering and is paramount to good control. Disposition - review- by the CCB must occur. However, approval of disposition is delegated to the CCB Chairperson. The CCB members advise the Chair on how to dispose of the given request. CCB attendance is mandatory and members must be expert in their field of employment. The disposition and instructions to incorporate must be clear and concise.

Incorporation of an approved change cannot exceed the instructions of the approved change documents in order to be validated by V&V. Additional effort requires the preparation of a new change request. Inadequate, or confusing instruction will result in cancellation of incorporation and a referred back to the Chairperson of the CCB for clarification. IV&V and quality inspection is mandatory. Close out (completion) of the incorporation is performed by the SCM activity, but the event should be acknowledged by the CCB Chairperson and IV&V to verify that incorporation did indeed comply with the approved change instructions.

A well performed change management process relies of a well conceived status accounting system to enable almost minute to minute information about a given change or series of changes. All activity, from the time a change request is logged in to the time it is officially closed out, must be recorded in the status accounting system and such information be available by query or report at all times.

STATUS ACCOUNTING:

If SCM is a communicator to all on a software project as well as those who work with it in support of the project, then the mechanism for communication must be planned and the necessary physical and human resources provided.

Status information is derived from the software development library which is established early in the project by SCM and the project Librarian. The library provides the work areas for documenting and coding by the software engineers and the subsequent transfer and release into to the project support library segment for component testing and release to the master library for system test and eventual delivery to the user. This system is maintained by the Librarian, in coordination with SCM with well defined need-to-know rules of access to the library and its segments. The working library segment is controlled by the Programmer. The other segments are controlled by the Librarian.

The platform for status accounting data is the software development file (SDF) that is established for each defined element of software in the software hierarchy - from lowest defined and controlled element such as unit or object or class up to system level - the top element. All data including documented requirements and design and all code, test descriptions and procedures plus reports reside in the SDF. The data that the SDF contains are defined in a data elements dictionary and it is the selected elements that make up a status report in hard copy or screen query format.

SCM insures that a well planned and maintained process enables traceability. This is vital to providing information for impact analysis, change history for determination of failure modes and error patterns and the rationale for approval and incorporation of previous changes. The key to good traceability is an efficient query system that is assessable at all times to all people including those at the nuclear sites and stations!. In addition to the impact analysis requirement, the test team relies on status accounting data for quick questions and above all the IV&V activity for validation of approved documentation and the verification of change incorporation.

The key to good status accounting is in the selection of primary data elements that SCM determines are important to the developers, maintainers and users. These should be selected by consensus, but beware of selecting too many which in turn create cumbersome reports. Another key point is the formatting of the data queried or reported. SCM provides the ability for one to create unique reports as required as well as developing and maintaining standard reports for management, customer and the general user.

Collection of the status accounting data is most important. SCM assures that only valid and accurate data is collected to produce valid and accurate reports. If collected on a manual basis, SCM provides for a quality inspection of such data prior to input by, many times, data entry clerks. As more and more automated techniques and tools are employed, almost all desired data elements can be automatically transferred into the applicable SDF such as creation of a change request via a computer terminal. Once basic information is provided, a change number is assigned and logged into the SDF. Collection techniques should be easy and quickly entered. The more transparent the process is to the developer/maintainer the better the data.

CONFIGURATION AUDIT:

The Configuration Audit is the close-out activity for the SCM process. The audit is managed by SCM to demonstrate that the original requirements in the Software Requirements Specifications/Documents have been satisfied as attested to in the Test Report data obtained during prescribed tests. In addition, the Configuration Audit assures that the software code being

released/delivered is supported by a Software Product Specification, a Version Description Document and a specified user/system maintenance manual that usually goes with the software product.

The Audit activity must be formal and carried out with careful planning that includes the attendance of the all of the principal systems, architectural, design, test, quality and maintenance engineers plus the SCM and Data Management personnel. There are several standards related to reviews and audits and their check lists, as a minimum, should be used for guidance in planning and conducting the audits. The IEEE STd 1028-1988 Software Reviews and Audits and Mil Std 1521B are recommended references. If detailed inspection is carried out during the audits, the chances for failures become far less than without such inspection.

Normally Configuration Audits are managed by the SCM activity and Co-Chaired by the Developer and User Project Managers. Many times, however, the chairs have been delegated to the SCM activity. In either case, the Chairs must recognize that the audit is almost the last chance, short of extensive field tests, to insure that a correct and reliable "As Built to As Design" has taken place and that the software product will meets its assigned mission requirements.

INTERFACE CONTROL:

From the issues and problems stated at the beginning of this paper, the planning for interfaces and their identification, control and status accounting is very important and are important to SCM for traceability of a change request's impact on the software product.

The most common way to initiate an interface activity is to form an Interface Control Working Group (ICWG) to review the System Requirements and identify the interfaces on a system to system basis. The next step is to review the Software Requirements Specifications to identify the process to process interfaces that will be required. Software Engineers can then be tasked to develop Interface Requirements Specifications and subsequent Interface Requirements/Design Documents for Analysis, Review and Approval by the ICWG.

Membership in an ICWG, at the system level, includes the Project Managers of the interfacing systems such as a nuclear power station to a power transfer station or the hardware generator system to the reactor system. Lead Engineers, software and hardware, representing interfacing processes such as sensors to digital signal processors are also members along with the quality, reliability, V&V and SCM personnel. These members are usually from the Prime Designing/Developing activity, although those from other interfacing activities may be called in.

The Prime Designer/Developer activity's SCM is the manager of the ICWG and performs SCM for interface specifications and documents in the same manner as development data.

In this day and time, it is also important that ICWG tasks include identification of those interfaces associated with Local Area/Wide Area Networks and the associated Client/Server software employed with the networks. If such networks are employed the designated Prime/Developer's Network Manager should be a member of the ICWG.

SUBCONTRACTOR CONTROL:

Up to this point the discussion has centered on the role of SCM in the development domain, but now SCM has to be described as a function of the acquirer's/purchaser's/buyer's procurement of software products. The following guidance also applies to "in-house" agreements

for other groups to supply required software products. This could also be the person sitting next to you who has been asked to perform some specified tasks.

There are at least three categories of procured products to be addressed. The first is a software product that is fully developed by another software developer. Such a contract requires, the defining of the acquirer's requirements for the design, development, build, test and delivery of a software product (it may even require subsequent maintenance). The second category is the ordering of a modified software product that is offered by a developer or reseller as modifiable to customer specification or as a turnkey system meeting customer specification. The third category, is the purchase of "off-the-shelf" software products on an "as is" basis. This is known as COTS, computer off the shelf software. It is also known as NDI, a non developed software item, developed by the acquirer, but not a deliverable element.

The first category must be carefully planned and subcontractors that can meet exacting standards for evaluation are selected. This includes the selectees being able to demonstrate their understanding of the SCM process to the satisfaction of acquirer's SCM function.

If acquirer's SCM is to aid in assuring the integrity of safety critical software, then the same rules applies to the subcontractors SCM! The acquirer's SCM must therefore prepare an SCM Statement of Work (SOW) section of the subcontractor Request for Quotation (RFP) that cites exactly what the acquirer's SCM expects. Take nothing for granted and insure that the SCM requirements do not get subsumed in favor of other demands. Once a selection has been made the Contract SOW must also be explicit and require a SCM Plan that will describe how the subcontractor will perform the SCM process. This, in turn, will require periodic, but continuous, monitoring by SCM and the formation of a continuous interface between counterparts throughout.

The second category should be treated the same as the first one, however, SCM must provide for constraints applicable to disclosure of the subcontractor's design and development process under the existing copyright and rights in data laws. Therefore agreements, sometimes with legal assistance, must be made to obtain as much technical data as possible to enable proper identification of the software product and to manage changes that may impact subcontracted software within the acquirer's domain much less changes made by the subcontractor.

The third category is most difficult for SCM identification and control purposes. Hopefully, SCM can obtain technical data sheets in addition to a "fuzzy" users manual to enable adequate documentation for identification and prior notification of pending version changes to the procured software product. This can accomplished best through direct dealing with the software vendor. SCM stands little chance dealing with a re-seller/distributor due to agreement for re-sale made with the vendor.

Provisions for acquiring of software must be included in the SCM plan and be a basis for internal management audits.

SCM SYSTEMS AND TOOLS

There are many new and innovative software development systems on today's market all of which are designed to increase productivity, quality, reliability, lower cost and make the developer's life less tedious. Systems for development of documentation provide templates that can reduce writing time by half; and also provide the means for traceability and development of a document tree that in turn aids in the establishment of the software hierarchy. Source code control during development such as CMS and SCCS is now very popular and most major developers are using a variety of versions for a large number of developers - large and small projects. Such systems are good and should be evaluated in order to maintain quality and reliability of the software and provide transparent controls to the developer.

Good productivity can be achieved if the software development system selected fits the job at hand and does not require "stuffing" the job into the system! To prevent this, an evaluation plan should be prepared to include the methods for selecting vendors, the criteria for evaluation and demonstration of their product to prescribed benchmarks and the ultimate check list for selection of one, two or three trial runs before the final selection.

There are four basic SCM System models to select from. No one model may meet all requirements such as an applications program type development vs an embedded computer type development. These models are: the check-out/check-in (co/ci), the composition, the long transaction and the change set models.

The co/ci and composition models are programmer driven. The co/ci provides for complete control and access to the programmer's work and allows for work to be performed on the module by checking it out, making revisions and checking it back in with an indication of new version or just a revision to the current version. The composition model allows one or more programmers to move a version into a work area of the model and create a new version or enhance a current version. The software product then consists of versions of elements selected by the programmer(s) for release. No two programmers, however, can work on the same element at the same time.

The long transaction model is "system oriented". A team of programmers, sometimes at several different locations, select a version of software residing in the repository to work on and move that selected version into a work area. Once there, the programmers may work on several elements at the same time. By consensus, the programmers will/may declare a new version and commit it to the repository for eventual release. All versions are stored. Any version can be selected to create the next version. This is compatible with the principles of concurrent engineering.

The change set model is similar to the co/ci, but involves retaining the deltas created against a fixed version of software. Several programmers can work on individual elements and log in the deltas - adds, deletes or changes - and can then select the deltas required to make up a desired version. This model also provides detail status of the controlled code for reports including the information needed for good traceability and impact analysis.

Some of the more familiar SCM systems are SCCS, RCS, DSEE, CCC, Rational Rose and Aide-de-Camp. all are adaptable to most software languages although Rational is known for Ada

language support and SCCS is UNIX oriented. In any case it is recommended that evaluations be conducted and appropriate tools selected that will enhance the SCM process.

VII

TRAINING

In order to insure efficient and effective SCM, all of those closely connected to the SCM process should receive at least awareness training and those who actually perform tasks should become proficient through seminars and on the job training programs. In addition, management personnel and others in the user environment indirectly impacted should also understand what SCM is and how it works to their advantage.

A number of training plans are available. Local colleges and universities have one or more extension type courses available and there are a number of training companies and consultants that provide public and in-house seminars and on the job training assistance.

Training is important and once provided should be followed up on at least a yearly basis. In addition there are number of special interest groups or association groups that have been and are being formed for CM, SCM and Data Management. There is the Configuration Management group in the Washington DC area that meets every couple of months. (Contact D. Murphy at 703 406 8787) The Electronics Industries Association (EIA) Data and Configuration Management Committee meets quarterly - twice in the DC area and includes an annual workshop that also provides a tutorial on the CM process. (Contact C. Denham, 202 457 4965). Some of the groups stress software entrepreneur, technical issues, assessment and certification requirements. All are very good and those with an interest should attend.

VIII

CONCLUSION

When Software Configuration Management has been employed on large, medium or small projects, the processes has proven to be a vital and integral part of the software product's development.

The SCM process should be performed on all NRC projects, especially those of a safety critical nature to provide the means to create and control the software product's documented identification, perform change management, status accounting, interface and subcontractor control during development and maintenance.

Selection of an SCM system to fit the needs of a project or like projects will assist in alleviating the software critical issues facing the NRC today.

REFERENCES

1. NRC Policy Issue - Digital computer Systems for Advanced Light Water Reactors, James M Taylor, 16 Sept 1991
2. Guidelines Software Configuration Management, HEDL-TC-2263, N.P. Wilburn et al, January 1983.
3. Configuration Management Standard (DRAFT), ASQC 8 May 1993
4. ANSI Std. Criteria for Digital Computers in Safety Systems of Nuclear Power generating Stations, P-7-4.3.2, [EXPECTED] Sept 1993.
5. ASME NQA 2a 1990, part 2.7, Quality Assurance Requirements of Computer Software for Nuclear Facility Applications.
6. IEEE Std. Software Configuration Management Plans 828-1990
7. High Integrity Software Standards and Guidelines, NIST Special Publication, 500-204, D.R.Wallace, et al, Sept 1992.
8. IEEE Master Plan for Software Engineering Standards (DRAFT), L. Tripp, et al, 1993.
9. Space Shuttle Computer Safety Questioned, Associated Press 29 June 1993.
10. Standard for Software Safety Plans (IEEE WG DRAFT, 1993).
11. IEC Standard, Publication 880, Software for computers in the safety systems of nuclear power stations, 1986.

FIGURES

1. Configuration Management Model, H.R. Berlack, Software Configuration Management, John Wiley and Sons, New York, 1992
2. User Expectations - Configuration Management Process Objectives - Master Plan for Software Engineering Standards (DRAFT), L. Tripp, et al. 1993
3. Software Errors Found in the Field.

6.6 How Much Software Verification and Validation is Adequate for Nuclear Safety?: Mr. Roger U. Fujii

How Much Software Verification and Validation is Adequate for Nuclear Safety?

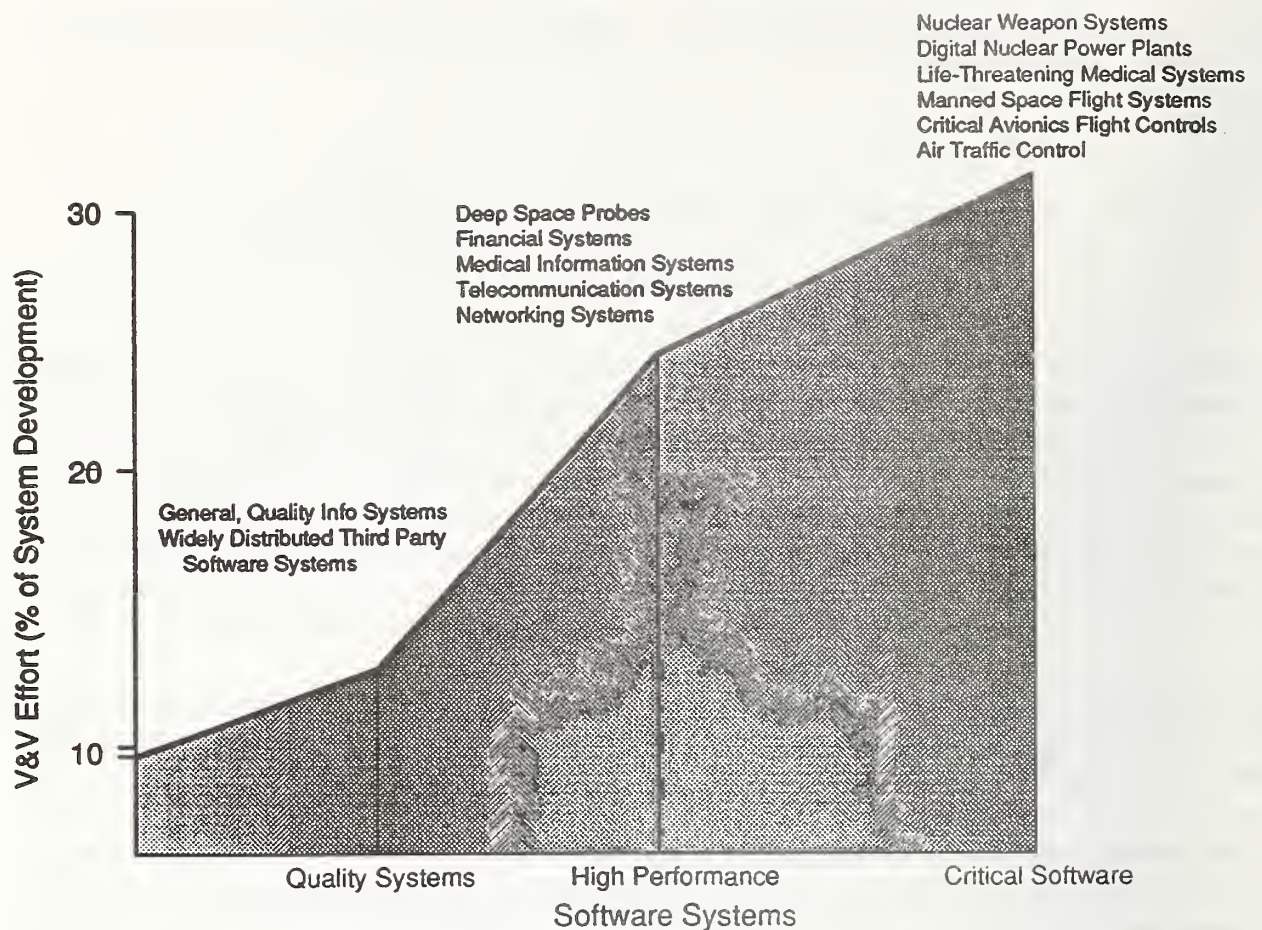
Roger U. Fujii

Logicon, Inc.

Introduction. For over 25 years, software verification and validation (V&V) has been applied to major DOD weapon systems, especially nuclear weapon systems, to ensure that the software is free of catastrophic software errors. Software V&V is a systems engineering discipline that evaluates the software as part of the entire system including hardware, human operators, and other interfacing software. When applied from a systems perspective, software V&V has been proven to be an effective technique for the early detection and correction of errors. Several V&V cost/benefit case studies for the Rome Air Development Center have shown that the dollar savings of the early detection of errors clearly offsets the cost of the software V&V.

Rule of Thumb for Estimating a Software V&V Effort. The question facing systems budget planners and program managers is to estimate how much software V&V is adequate for nuclear safety. The V&V effort is generally expressed as a percentage of the development effort. The "rule of thumb" for estimating the V&V effort is that software V&V ranges from 10% to 33% of the development effort depending on the criticality of the system. For the highly critical software such as nuclear weapons delivery, life-threatening medical, avionics flight controls, manned space flight, and air traffic control systems, a comprehensive and thorough software V&V effort is required because of the obvious impact of a software error to the loss of life. Highly critical software requires approximately 33% of the development effort to be devoted to software V&V. For high performance systems such as the unmanned deep space probes, financial software, telecommunications, and networking systems, where a software error may significantly degrade the system performance or cause the system to function permanently in the degraded mode, the "rule of thumb" for the V&V effort is approximately 20-25% of the development effort. The criticality of high performance systems is dependent on the user's tolerance to loss of system performance and the recovery time to fix or bridge the erroneous portion of the system. Many other programs and systems demand a level of quality above the standard quality assurance effort provided by the development team. When software V&V is applied to these quality systems, approximately 10% of the development effort is used. Figure 1 illustrates the range of software requiring V&V such as quality systems requiring 10% of the development effort to highly critical software requiring 33% of the development effort.

However, estimating software criticality is more than categorizing the type of software and the consequences of software errors. Estimating software criticality involves a rigorous analysis of the system functions, each function's effect on criticality criteria (e.g., safety, security, performance), and key parameters affecting the development environment. Figure 2 illustrates an example of these application features and development parameters which affect the criticality



93-08-101a

Figure 1: Rule of Thumb for Estimating Software V&V



93-08-100a

Figure 2: Software Criticality Parameters

of software. First, it is important to define criticality as it relates to specific system functions. For example, criticality for a nuclear power plant could involve all reactor control functions, interfaces to the pumps/values and sensor involved in monitoring and controlling critical plant coolant, control switches and console receiving operator control of critical functions, displays functions which process and format data of critical plant parameters for display to the operator, security functions to prevent unauthorized access to the plant software system from maintenance personnel or other parties who may inadvertently or deliberately interfere with plant operations, and database functions and data containing key plant control parameters. Other critical features may include system security, easy of useability, maintainability, or specific performance features. Next, the development environment is assessed by examining: 1) maturity of the system architecture (i.e., is it a copy of a functioning system or is it new with no prior history of use); 2) is the processor new and invokes new technologies or is it a proven, widely used processor; 3) are there many aids and tools for the development reflecting a robust environment or are the development aids and tools new and thus possibly error prone; 4) is the development staff experienced in this type of system or is the technology and functionality new to the development team; 5) is the development following a rigorous methodology or is it following an ad hoc methodology that has unknown performance; 6) is the development schedule tight such that any schedule or development perturbations causing more error potential in the software turning normal functions into critical functions; and 7) is the software functionality new thus increasing the likelihood of errors in the software or is the software based upon previous developed work thus lessening the expected error potential.

Aggregating the detailed analysis of the features contributing to the critical of a software using the method described above gives a clear cut method of assessing the criticality of any software. For example, high criticality software could require less than the normal "rule of thumb" 33% if the critical functions are well compartmentalized into small areas of the software, uses a well defined software development environment, is not a new development but is one that follows previously built efforts, and has a well trained staff who have developed similar programs in the past. Therefore, part of the answer to the question of how much software V&V is adequate for nuclear safety is determined by the scope of functions involved in the criticality of the software, how the software is designed and architected, and how the development environment supports or deters from the software V&V effort. Most software V&V efforts fit within the 10-33% rule of thumb identified in Figure 1.

System Safety Framework. In the following sections, a framework will be described that will show how to estimate how much of the software should be analyzed and tested. This detailed decomposition approach first starts with the system framework and proceeds through definition of safety critical requirements.

Figure 3 illustrates the overall system/software development cycle starting with the system concept definition. The purpose of this figure is to illustrate that the software must always be analyzed and tested from a systems perspective. The interaction of software with the hardware, human operators, and other software elements is more complex and interwoven into the total system solution in modern systems than existed on previous systems. Therefore, software V&V

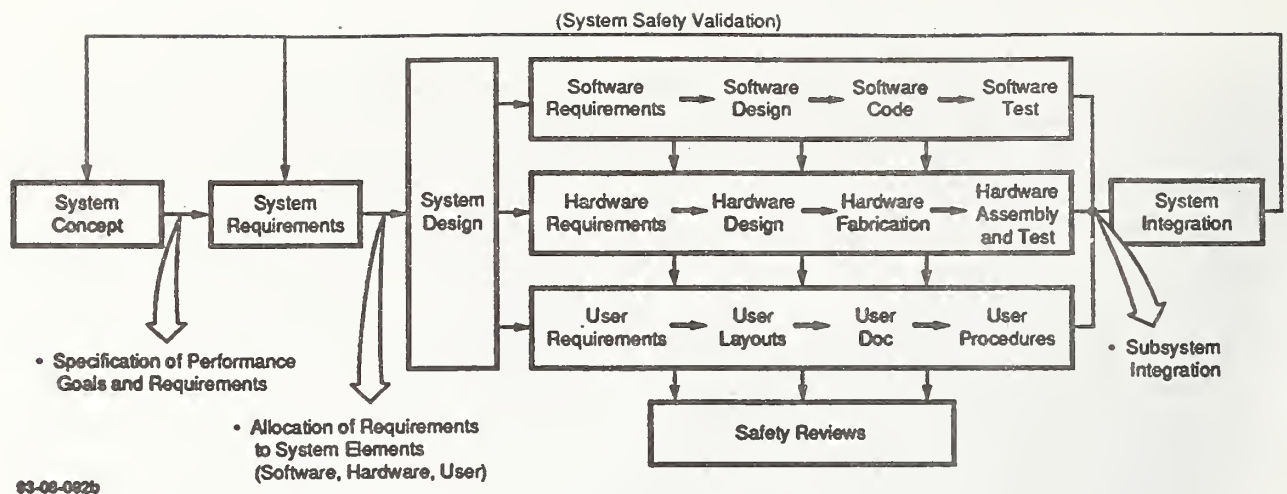


Figure 3: System Safety Framework

must have a systems engineering orientation for an effective criticality analysis to be performed. Within the system development cycle, there are key stages in the metamorphose of the software. The two important steps in a system development is the specification of the system performance requirements, including any safety, security, and other special requirements. Second, the allocation of the system requirements to hardware, software, and human operators is an architecting choice which is a crucial determining factor to defining how the system functionality is achieved and to identifying how much effort is needed to demonstrate that the requirements are satisfied. In most modern, complex systems, software is behaving as the glue which ties all elements of the system together. The allocation of requirements is therefore one of the most important steps that define the ease or difficulty of integrating and testing the system components at later development stages. A poor allocation will tend to spread critical requirements throughout the system architecture making the later analysis, test, and integration effort very difficult and time-consuming. One method to avoid last minute surprises is to conduct periodic reviews of the product at key development milestones. At distinct review points in the software development phase such as at the Software Requirements Review (SRR), preliminary and critical design reviews (PDR, CDR). Test Readiness Review (TRR), and Formal Qualification Testing (FQT), safety reviews are normally conducted to assess the safety of the system and software as developed to that stage. Safety problems are identified at the reviews and corrective actions are taken to create a more error safe system architecture. At each stage of the software development, the software is verified for compliance with the previous phase's specifications and also being validated constantly against the overall system requirements and objectives. The early feedback loop of the software V&V analysis and test results assures the early correction of detected errors. Latent errors or weak designs leading to the greater error potential are identified and corrected as time permits to build a more robust system solution.

Information Criticality Analysis. Once a system architecture and critical requirements are defined, a flow analysis can be performed to determine those portions of the system which are impacted by each critical requirement. This flow analysis traces each requirement through the architecture and identifies all software routines, hardware interface components, and human

actions which are involved in processing the critical requirement. This trail identifies the portion of the system which must be examined by the software V&V effort. Utility functions and data required by the critical functions are identified as part of the critical path requiring detailed analysis and test. The criticality analysis is conducted for each defined critical function/requirement. By tabulating all areas of the system affected by the critical functions, one can obtain a picture of how much of the system needs the software V&V effort.

Instrumentation & Control Software Engineering Issues. New technologies and functionality being considered for instrumentation and control software for nuclear power plants present unique critical issues that require close examination during software V&V. Figure 4 lists some I&C software engineering issues which will present some unique issues for software. If features are designed into the software and hardware which allow plant performance (e.g., turbine efficiency) to be modified during operations, there is concern that important safety and control features could be altered or indirectly influenced to cause plant error recovery or detection operations to perform less than as expected. For safety and security features, there is a constant debate as to how much protection and security is enough and do the safety and security features interfere with optimal plant efficiency. For example, password controls have been used in many systems to control authorized access. However, as evidence has shown, when the use of passwords is overdone and required for the most simple of operations, operators begin to circumvent the protection by pasting the password on the terminal and ignoring any warning messages associated with the protection because they have seen similar messages too often, making the real message indistinguishable from the normal warning messages. Many system architectures are considering a distributed architecture that places the processing and controls close to the sensors and plant control features (pumps/vents/values). Today, most systems utilize a centralized control system where all the information is funneled into a main processor for decision-making and security control. With a more distributed architecture, more security features are required to control access. Centralized decision-making requires more communication links between the distributed nodes to evolve an overall picture of total plant operations. More decision-making is also being assigned to the software since it can react faster to detected error conditions. However, there are certain conditions where human operator decision-making surpasses any automated decision making especially in those situations where a complex, process flow analysis of many different system functionality coupled with past operating history is required. A new plant architecture must have clear cut points where the decision-making is left to the human operator. Many new power plants are using artificial intelligence and expert systems to detect developing error conditions and to aid the human operator in interpreting and responding to plant operations, especially to emergencies and error conditions. Plant designers are also trying to create system architectures where new technologies can easily be inserted into the system without having to redesign and develop the system a second time. All of these types of new performance issues will place a stronger need to perform software V&V on the new power plant software, with special concern and assistance required in the early system architecting phases.

Staff/Skill Mix. The software V&V teams' experience base and skill are equally important in determining how much software V&V is adequate. With the proper allocation of staff experience and skills, the software V&V can be optimized to assist the development team in the early

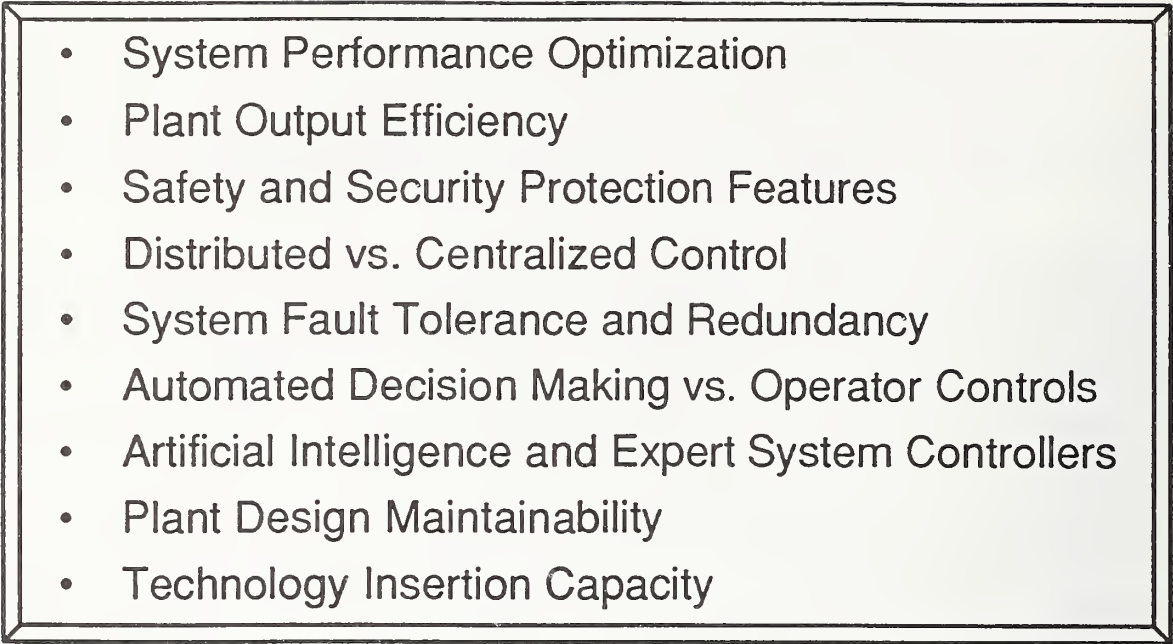
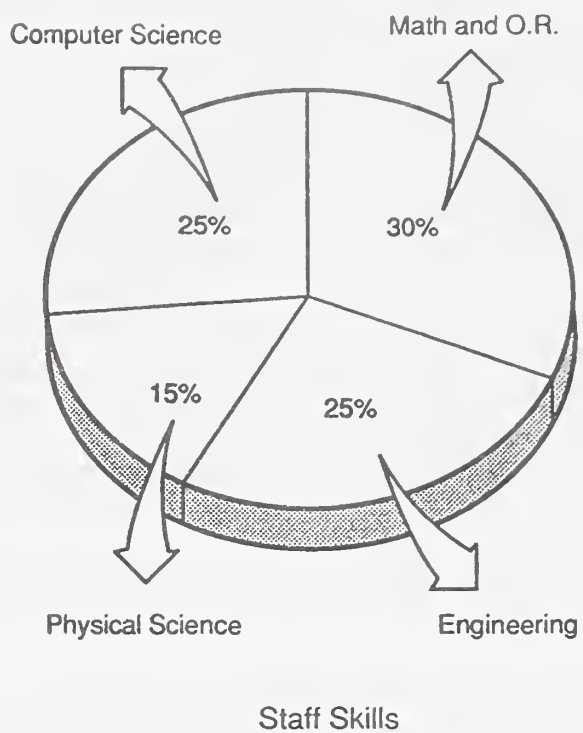
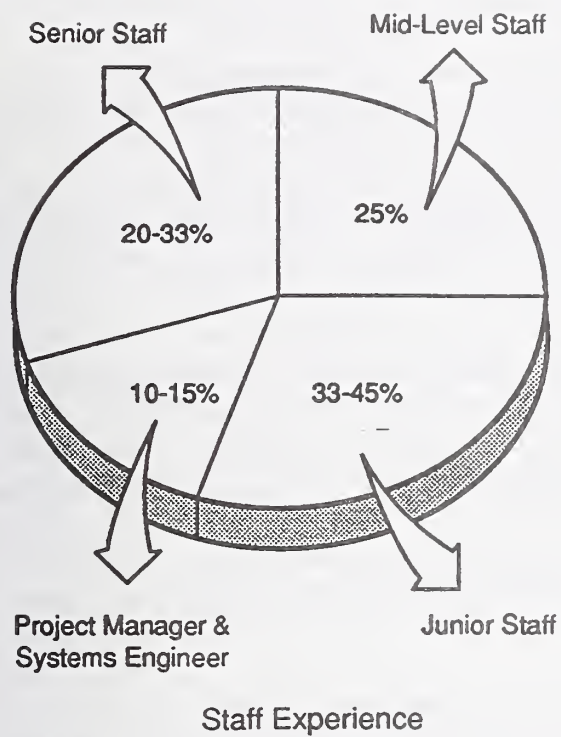
- 
- System Performance Optimization
 - Plant Output Efficiency
 - Safety and Security Protection Features
 - Distributed vs. Centralized Control
 - System Fault Tolerance and Redundancy
 - Automated Decision Making vs. Operator Controls
 - Artificial Intelligence and Expert System Controllers
 - Plant Design Maintainability
 - Technology Insertion Capacity

Figure 4: Software Engineering Issues

detection and correction of errors and to identify the subtle design flaws which turn into difficult to find errors. The author's experience has shown that the optimal staff experience is a mix using the "one-third" rule. A software V&V consisting of one-third senior staff, one-third mid-level staff, and one-third junior staff along with a program manager functioning as the systems engineer produces the optimal staff experience base. Equally important is the educational background of the staff. Because software V&V is more of a systems engineering discipline, personnel with strong engineering and hard sciences background are better able to evaluate whether the software solution is satisfying the real systems application. On a typical software V&V, the staff skill mix is as shown in Figure 5 with 70% of the staff having a strong science or engineering background. Knowledge of computer science is very beneficial but with the use of modern high level languages and more reliable compilers and support tools, the software V&V problem is not a coding problem but more an "engineering type" problem of determining whether the software solution corresponds to the systems engineering solution.

Summary. Software V&V is a comprehensive analysis and test of the system and its software. The software V&V must be performed in a system framework, especially when considering safety where portions of safety has been allocated to the hardware, human operator, and other interfacing software. In estimating software V&V, a rigorous, top-down approach of assessing the criticality flow within the system architecture defines how much software V&V is needed. For any criticality, especially safety, a detailed definition of the critical safety requirements is essential to ensure that the criticality flow analysis does not overlook any key requirements or portions of the design/code. As a general "rule of thumb", for high criticality and high performance software systems, software V&V requires 20-33% of the development effort.



93-08-115a

Figure 5: Software V&V Staff/Skill Mix

6.6.1 Questions: Mr. Roger U. Fujii

QUESTION: JOHN GALLAGHER (NRC): This isn't so much a question as maybe I can give you a data point with respect to safety-critical software in nuclear applications. There was a meeting last year at the Royal Academy of Engineers in England talking about the Sizewell-B software. The verification and validation effort for this project was between 300 and 350 man-years out of a total of 500 man-years, or about 70 percent of the effort in the development process was spent on the verification and validation activities, which is a little larger than yours.

MR. FUJII: Well, I have achieved 100 percent. But that was an unusual case. You know, there was a 2 man development team, and I had one and a half persons on it, something like that. But let me suggest to you that there might be something else at work because your V&V team gets to be so smart about the system that they become an alternative for your development team. So, when development gets into trouble, oftentimes V&V is pulled in to help out development. All I can say is that's a fairly high number. Perhaps the development team was in trouble, and sometimes the V&V team kind of mushrooms to that size because they're trying to solve a problem. Wherever you classify them could be a debatable issue, but that seems to be a pretty high number to me.

QUESTION: ALI HEKMATI (General Electric): In our particular case the safety-critical requirements happen to be very simple. It's just simply a comparison within the process variable and the set points and, as a result, you get a tripping. However, I was wondering, in your experience how have you handled the verification of the operating system in the compiler? I think as far as we're concerned that would be the most important part of our V&V.

MR. FUJII: The operating system in the compilers turns out to be, like I said earlier when I was talking about compilers, to be one of the most critical elements of critical systems. This is why DoD builds their own operating system for all of the nuclear weapons systems. They literally do not acquire it off the shelf for the reason that if you're acquiring it off the shelf generally there is no documentation that goes with it. It's also very hard to track down all of those Trojan Horse paths that might exist there for other applications but not for your critical application. This is an area we see as a great, great concern, that when you're starting to pick up third party software, like an operating system, which has a lot more features in it than you want for your application, you are exposing your system to a great deal of risk. You can perhaps take mitigation strategies to limit that risk.

Winston Royce was talking about how the compiler was being limited. In the case of Peacekeeper they were using ADA, and we did exactly what he said. After some early analysis it was determined that tasking would create such a huge complication to it that tasking was forbidden to be used as a language feature on the system. So, you can do that in order to limit your exposure, but once you have it there you have a large effort to prove it.

MR. McCREA: Sitting in this workshop that's exactly where I see you're lacking--that the NRC's mission, I think, is lacking--the objective, or the direction, of designing an operating system for microprocessor software based systems. At least in our case I think that's the most

important part of the safety-critical systems, because the algorithms for the safety functions are very simple and the operating system, as a component, is the most important.

MR. FUJII: Yes. I agree with you. We can take that up later as a further discussion.

QUESTION: John Knight said that formal verification is not panning out as well as we'd like, and Winston Royce said that provable correction is 56 years away. Do you agree? And if a mathematical-based discipline is not used in your approach to verification, how would you characterize the discipline used? Is it just a matter of it looks good to some experts?

MR. FUJII: Well, we think that formal proofs have a lot of value because they have some rigor that we're all trying to achieve in software analysis. In the process of applying a formal proof, or a formal analysis, you find out a lot about the assumptions that are in the specification or weren't in the specification. The process that one goes through in a rigorous methodical way proves to be a lot of value.

Now, I think a lot of the speakers were talking to what the practical aspects are of applying formal proofs of correctness. For one, since you don't have your formal specifications written in a form to be used for formal proofs of correctness, you have to first take your specification and essentially rewrite it in a normal proof format, and then secondly, apply this very rigorous mathematical approach. Now, that approach is very, very complicated, and we have used it on several projects. It becomes a highly tasking intellectual activity, taking some of your best talent, and it does take a while to do. So, until we can solve this time problem, it takes that much time to do it and it does take some high-powered talent. I think that's what a lot of the speakers were talking to in terms of the practicality, but you can use the notion that formal proofs are trying to give you the ability to specify in a somewhat rigorous way, all of the conditions that really describe your software and the problem you're trying to solve. So, in many respects, a lot of the techniques that people in our business on the V&V side use, we use to follow up that formal proof of correctness without going to the actual proving side itself. So, I believe that it has a lot of value but we have to overcome some of those obstacles before it'll become common place in our work.

6.7 Fault-Specific Verification (FSV)--An Alternative VV&T Strategy for High Reliability Nuclear Software Systems: Dr. Lance A. Miller

FAULT-SPECIFIC VERIFICATION

(FSV)

**An Alternative VV&T Strategy for
High Reliability Nuclear Software Systems**

Lance A. Miller, Ph.D

Science Applications International Corporation

PO Box 1303

McLean, VA 22102

(703) 556-7079

lamiller@mcl.saic.com

BASIC POSITION

- o Digital I & C can be safe for nuclear plants

- o BUT, software notably devious, need some changes:
 - Revised VV&T Strategy, (with changes in devel.)

 - Operator Decision / Transducer Functions must be specifically allocated and evaluated

 - Need better way to evaluate level of assurance (than IEEE 1012): e.g., a joint function of of System Complexity and Required Integrity

WHAT IS FAULT-SPECIFIC VERIFICATION ?

- o Over time, Enumerate and Classify *ALL Software Faults*
 - > *For all Lifecycle Artifacts (Reqs., Design, ...)*
 - > *Between any Artifact Pairs*
- o Over time, Develop and Empirically Validate
 - a large set of VV&T Methods -- *Specifically designed to detect and classify EACH software fault*
- o Migrate all methods towards automated (formal)
 - Static Analysis (vs. Manual & Dynamic Testing)
- o Develop, Share, and Update:
 - > *Data on Fault Occurrences and Classifications*
 - > *Success rates of each Method*
 - > *Ratings of Methods on Power, Ease-of-Use, Cost-Benefits, etc.*

EXAMPLES OF FSV APPLICATION

<u>Artifacts</u>	<u>Fault Description</u>	<u>V&V Level</u>	<u>Method Recommended</u>	Manual Semi- Automatd	
				Static	Dynam
				 v	 v
Reqs. Doc.	Reqs. Unclear	Class 3	Reqs. Analysis	S	M
		Class 1	Formal Spec'n Lang. (CaMera ReFine, Z, VDM, ...)	S	M
Reqs. and Design Docs.	Unintended Functions	Class 2	Reqs. Tracing Tool (Supertrace)	S	S
Design Doc.	<u>Com. Mode Fail.</u> Interrupts	Class 2	Soft. Arch. Review	S	M
	Non-cont. Loop	Class 2	Program-Graph Analysis	S	S
Implemented Source Code	Data - Schema Violation	Class 3	Data-Model Eval'n	S	M
		Class 1	Data-Schema Checker (New)	S	A
	Incorrect Data Referencing (pointers, sub-s)	Class 2	Cleanroom Inspctn	S	M
		Class 1	Runtime Reference Checking (New)	D	A

THE COMPLEAT PICTURE

Fault Specific Verification

+

Incremental Full-System Build Life-Cycle

+

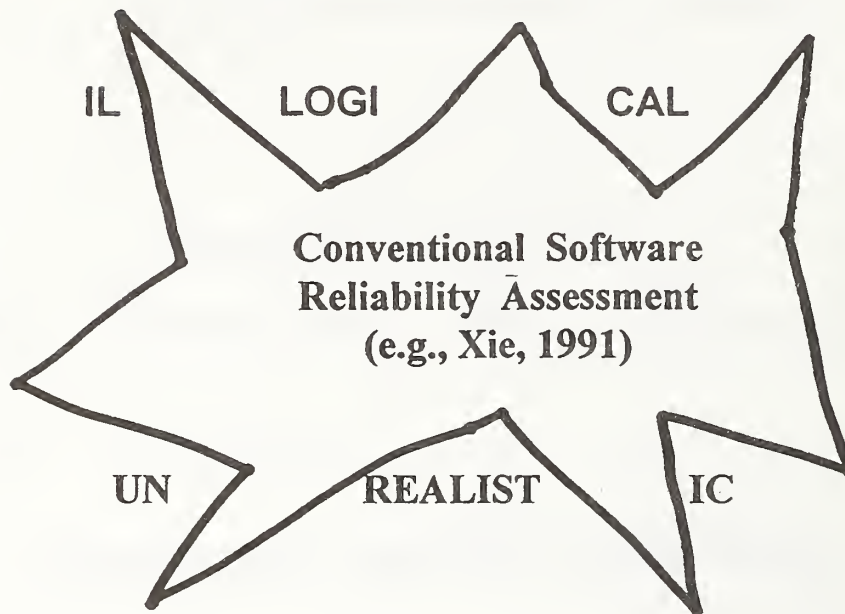
TQM (Continuous Process Improvement)

+

Re-Use: *Trusted Code*

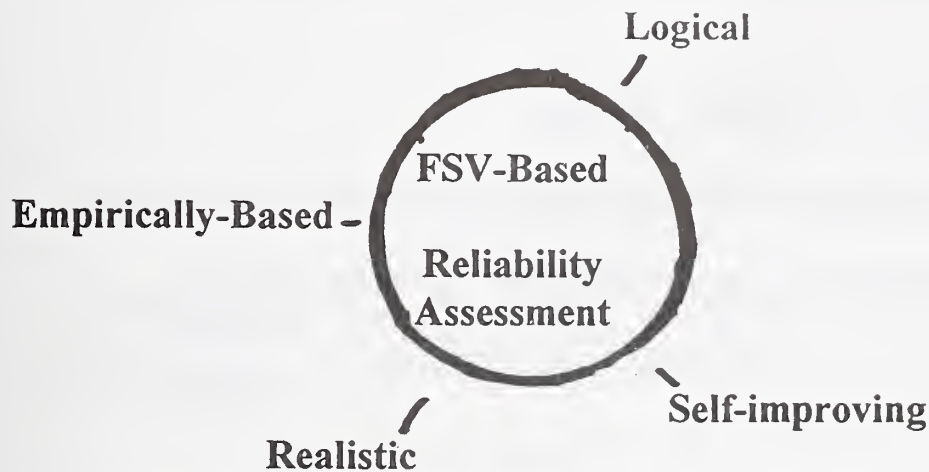
= *Ultra-High Reliability Software*

A Polarized View of Reliability Theory:



- o Based on Hardware Models (assumes long histories)
- o Assumes Fewer Defects-Found = Fewer Defects Remain!
(Failure Rate = $g[\text{No. Faults Remaining}]$)
- o Assumes All Faults Equal Size, Detectability, Significance
- o For complex systems, needed # test-cases is astronomical
- o Doesn't take into account fault "testability", method Power
- o Assumes more time = proportional info (calendar OR exec'n)
- o Deals only w/code artifact, only for dynamic testing (random)
- o Doesn't account for Intelligent testing strategy (order effects)
- o Doesn't account for Dynamic Development-Process modification
- o Doesn't take into account System Complexity re Defect Type

FSV RELIABILITY MODEL



- o Failures are due to faults in artifacts in ALL phases
- o Faults differ in significance, detectability, correctability
- o Methods differ in power, ease-of-use, cost-benefits
- o IF all faults are classified, F_i , have Method M_i to detect them

$$R_{fi}(t) = g[\text{Power}(M_i, F_i)]$$

estimated from \nearrow Empirical test data
 \searrow Mutation Tests

- o IF unknown faults remain, UF_i ,

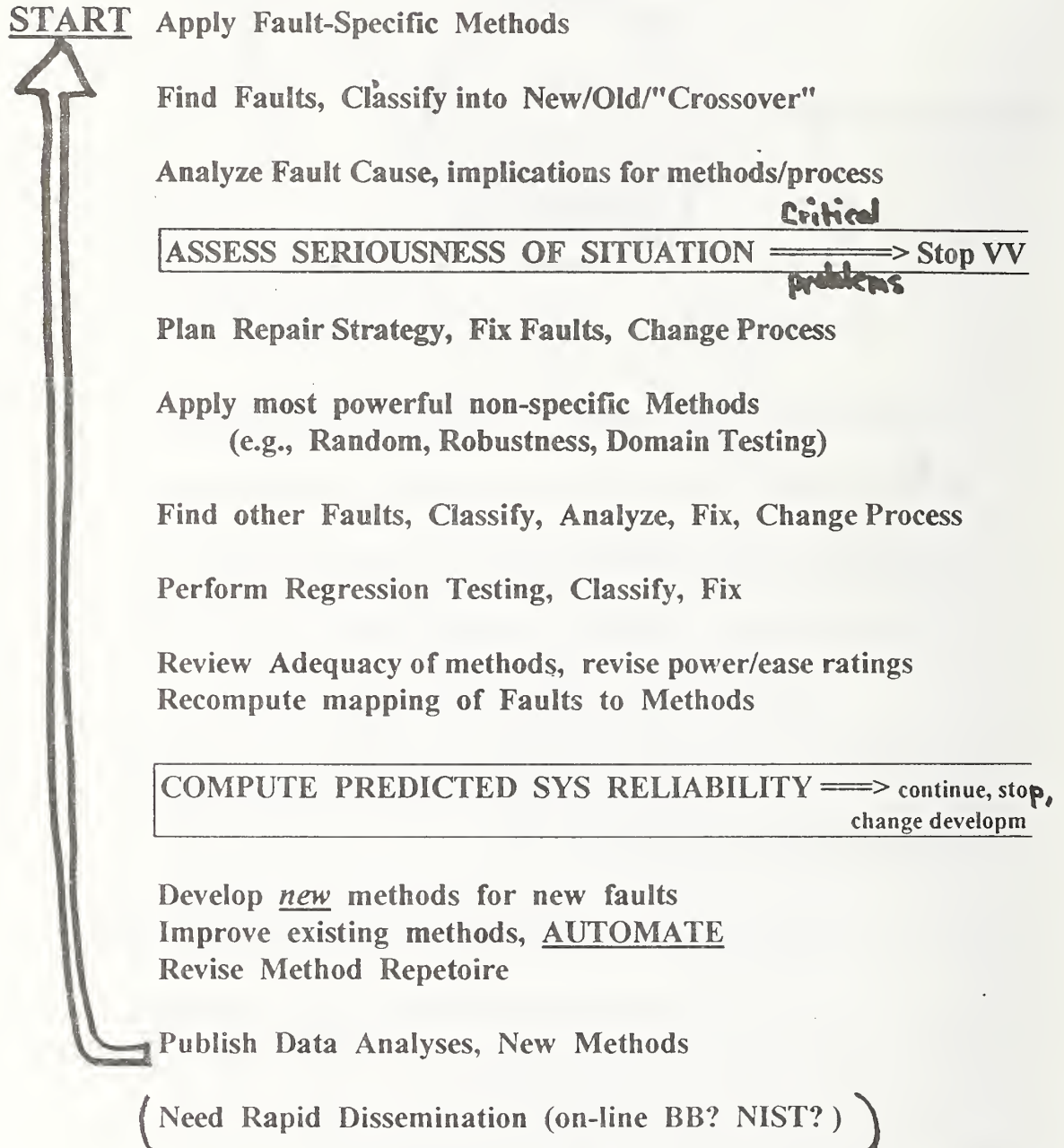
$$R_{F_i}(t) = g[\text{Power}(M_i, F_i), \text{Power}(M_j, UF_j)]$$

estimated from: system complexity,
 empirical tests, mutation

- o With constant improvement in FSV data, methods, ratings, etc.

$$Pr(UF_i) \rightarrow \epsilon \quad Pr(\text{Undetected known faults}) \rightarrow 0$$

FAULT-SPECIFIC VERIFICATION LIFECYCLE



Present Status for FSV

- o Classification of Need for V&V as Function of Complexity and Required Integrity (NUREG/CR-6018)**
- o Rating method for VV&T Techniques in terms of 4 POWER and 4 EASE-OF-USE Factors, Weighting for 3 Classes of V&V (Ibid.)**
- o Detailed Enumeration of Many Artifact Faults (NRC/EPRI Contract Report, forthcoming)**
- o Mapping of Recommended Method to Faults (Ibid.)**

6.8 Certification of Software for Reuse in Safety-Critical Applications: Ms. Charlotte O. Scheper

CERTIFICATION OF SOFTWARE FOR REUSE IN SAFETY-CRITICAL APPLICATIONS

Charlotte O. Scheper

ABSTRACT

This paper describes a process for certifying software for reuse in safety-critical applications. This process is based on a two-tiered hierarchy of certification levels. The first, or Primary, level specifies evaluation objectives commensurate with assurance demands of increasing levels of criticality; i.e., it provides graded requirements for assurance evaluation based on criticality. The second, or Secondary, level specifies increasing levels of confidence that the certification process has met the evaluation objectives of the Primary level. It provides graded requirements for the rigor and breadth of the certification techniques needed to achieve the evaluation objectives given particular component and system characteristics.

The structure of this certification process was designed to adapt to the certification requirements of multiple domains. It is based on a framework of evaluation techniques and tools that are common across all domains (and all certification levels). The selection and use of the techniques and tools to accomplish the evaluation objectives specific to each of the certification levels are managed by certification policies that are specific to a domain. The levels to which reusable components are certified are common across all domains, but the target certification level for a component is selected based on the derivation of domain-specific certification requirements.

INTRODUCTION

Certification provides assurance that a process or product meets a fixed set of requirements. When viewed as an aggregate, certification policies based on current standards (e.g., DoD-Std-2168, MIL-Std-882B, RTCA/DO-178A, ISO-9000, ANSI/IEEE-ANS-7-4.3.2-1982, and IEC 880) constitute different levels of certification differentiated by the rigor and thoroughness of their requirements. The differing levels of certification provided by existing standards reflect differences in the intended applications or uses of system components. Generally, the more critical the correct operation of the component to the system and the more harmful the effect of system loss or malfunction, the more stringent the certification policy and the more extensively the component must be tested and analyzed. On the other hand, certain design attributes and operating characteristics of a system, such as the way in which components are implemented and the processing constraints under which they function, may make it more difficult to correctly implement components and may lessen the suitability of testing and analysis methods used for their verification and validation. Therefore, the evaluation procedures used in the certification process should vary with respect to both evaluation objectives (how extensively a component will be tested and analyzed) and acceptable level of confidence of the evaluation techniques (what methods can be used to attain the required level of completeness in the testing and analysis).

When the certification process addresses reusable components within a safety-critical application domain, it has to be able to accommodate multiple levels of evaluation objectives. It is recognized that there is a need for a graded classification of safety functions for safety-critical applications and varying levels of assurance associated with each of the criticality levels [SECY91, IEC45A, ANS7432, and Wallace92]. It is also recognized that system attributes and the required level of assurance affect the applicability and overall confidence in evaluation techniques [SECY91, Wallace92]. Since the reuse of software components brings together components from diverse development projects and disperses them to be used in new projects, a component may play a different role in the new system at a new level of criticality. Ultimately, the reused component will have to be verified and validated with respect to the safety requirements of the new system. The certification of reusable components for safety-critical applications requires, then, the establishment of a multi-level certification process that (1) takes into

account different criticality levels and the applicability of particular assurance techniques at different levels and for different system characteristics and (2) defines the validation procedures at different levels in such a way that the potential reuser of a component can judge the distance between the current level of certification and the required new level.

THE MULTI-LEVEL CERTIFICATION PROCESS

A multi-level certification process for reusable components under development at Rome Laboratory [Scheper93] addresses the two aspects of certification through a two-tiered hierarchy of certification levels and the derivation of certification requirements for a component to determine its appropriate level of certification for a particular application. The Primary level of the hierarchy specifies evaluation objectives commensurate with assurance demands of increasing levels of criticality; i.e., It provides graded requirements for assurance evaluation based on criticality. A Secondary level specifies increasing levels of confidence that the certification process has met the evaluation objectives of the Primary level. It provides graded requirements for the rigor and breadth of the certification techniques needed to achieve the evaluation objectives given particular component and system characteristics.

An overview of the multi-level certification process is shown in Figure 1. The certification process uses the library structures and databases that implement a model of the domain to determine a target certification level for a candidate component. The selection is based on the certification requirements that have been determined by an analysis of the domain. These requirements consist of two parts: an assignment of criticality level and a specification of required level of confidence (RLOC) values for a prescribed set of component and system attributes. The certification requirements are used to select the appropriate certification level as follows: the Primary certification level is selected based on the assigned criticality level and the Secondary certification level is selected by computing an acceptable level of confidence (ALOC) based on the specified RLOC values. Based on the target certification level and the relevant application domain, the appropriate certification policy is determined. The certification policy defines the procedures for evaluating the component. It associates certification levels with evaluation techniques based on the evaluation objectives for each of the levels and the level of confidence ascribed to each evaluation technique relative to each evaluation objective to define evaluation procedures. The evaluation procedures stipulated by the policy are enacted using the available tools and methods. If the component passes the evaluations based on the criteria established by the policy, then it becomes a certified component.

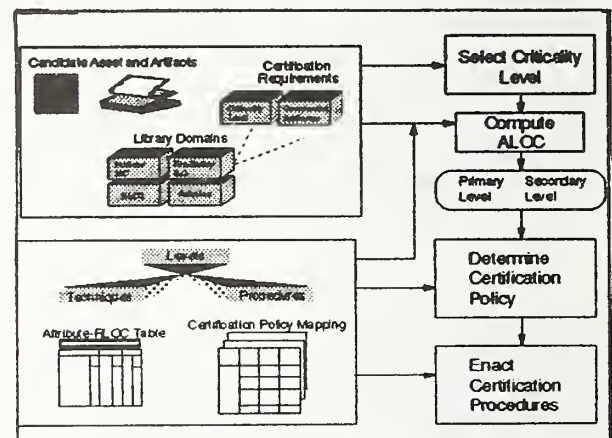


Figure 1. Process Overview

Certification Levels. The Primary certification levels are summarized in Table I. There are three levels based on criticality and one unevaluated level. They specify increasing assurance demands for increasing levels of criticality; each level assumes that the assurance demands of the preceding level have been met. The focus of the evaluations associated with the levels range from the internal correctness of a component to more global measures of correctness such as how the component interacts with and interfaces to other components to achieve system requirements within a particular operational environment.

Table I: Primary Certification Levels

Primary Level	Level of Assurance	Error Probability	Level of Criticality
0	None	NA	NA
1	Implementation Correctness	Probable	Low
2	Design Correctness	Improbable	Medium
3	Operational Correctness	Highly Improbable	High

Within each Primary level, a Secondary certification level specifies a level of confidence that the evaluation objectives of the Primary level have been met. These levels are distinguished by the increasingly global requirements against which the component is verified and by the increasingly sophisticated evaluation procedures used. The Secondary levels are summarized in Table II. Although the Secondary levels are the same for each of the Primary levels and specify the application of the same class of procedures, the particular evaluation objectives of the Primary level determine the overall scope of the certification evaluation. Thus, although components certified at the same Secondary level for different Primary levels have been evaluated using similar techniques, they will have been evaluated with respect to different objectives and thus, ultimately, for different levels of completeness with respect to associated error classes.

Table II: Secondary Certification Levels

Secondary Level 0	Evaluation Techniques	Requirements Level	Level of Confidence
0	Static Quality and Functional Analysis	Local to Component and/or Quality Model	0
1	Dynamic Quality and Functional Analysis	Local to Component	1
2	Dynamic Functional and Static Behavioral Analysis	System	2
3	Dynamic Behavioral and/or Formal Analysis	Mission or Operational Environment	3

Certification Requirements. Certification requirements are derived for components by a domain analysis that classifies components according to criticality and establishes values for a prescribed set of component and system attributes. The criticality classification is currently based on the effects of system failure on the operational environment and on the effect of component failure on system operational capability. It is accomplished by examining the correlation between the possible system failure modes and their effects on the operational environment to determine the hazards that can result from system failure, and by examining the correlation between the failure of the component to function

correctly and deliver correct results and the system failure modes. As a result of this analysis, one of three criticality classifications is assigned to the component: low, medium, or high. These assignments are made as follows: low if component failure produces minor effects on system operation or if component is used in systems whose operation poses at worst minor hazards to its operational environment; medium if component failure severely limits the operational capability of the system or if component is used in systems whose operation poses hazards to its operational environment that could result in an inability to accomplish the goals or complete the activities of the intended mission; high if component failure results in catastrophic loss of system operational capabilities or if component is used in systems whose operations pose life-threatening hazards to its operational environment. The target Primary certification level for a component is selected according to its criticality classification as follows: Primary Level 1 if low, Primary Level 2 if medium, and Primary Level 3 if high. It is not certain that there is a definitive criticality classification for safety-critical applications, although two current efforts are noted in [Wallace92]. The approach taken by this certification process should be compared to developing approaches.

There are currently four classes of component and system attributes specified for determining certification requirements: requirements, implementation, verification, and usage. Each of these classes contains a variable number of individual attributes that specify types within the class. For each attribute there are four possible characteristics. As a result of the domain analysis, the appropriate characteristic is determined for each attribute and a value is assigned to the attribute based on four distinct rankings of characteristics. The value ranking of a characteristic is based on the effect it has on the difficulty of verifying correctness for a component. Each of the four ranks corresponds to a required confidence level (RLOC) which indicates an increasing need for confidence in the rigor and applicability of the certification evaluation techniques. The current attribute-RLOC specification targets a broad range of systems. It can be tailored specifically for safety-critical systems or for a particular safety-critical application domain. The set of attributes and their associated characteristics can be selected commensurate with the current systems in the domain, such as those used in nuclear power systems. Also, as the state-of-the-art changes, the set of attributes can be adapted. The use of the individual RLOCs to compute an overall acceptable level of confidence allows the difference between old and new certification levels for a component to be objectively assessed as system requirements and capabilities change. This ability to incorporate changes in the assignment of certification levels and the selection of evaluation techniques provides a degree of flexibility that is desirable for regulatory processes that have to accommodate evolving technology [SECY91].

Evaluation Framework. The evaluation framework for the multi-level certification process is the set of techniques (and the tools and methods that support them) that are used to perform the certification evaluations. The current framework categorizes evaluation techniques into three classes (quality, functional, and behavioral) according to the aspect of correctness they address. Quality evaluation addresses processes that are used in developing the software and with internal structures and characteristics of code that are deemed to be either predictors of quality or detectors of anomalies that may indicate deficiencies in quality. Quality evaluation techniques included in the framework rely on the computation of quantitative measures based on the hierarchical quality model described in [Bowen85], which associates user-oriented quality factors with specific software characteristics and metrics for those characteristics.

Functional evaluation addresses the correctness of the component with respect to its functional requirements. Three categories of techniques are included in the framework: static, dynamic, and formal analysis. The static techniques are based on human and automated inspection. The dynamic techniques are based on executing the component using test cases representing usage scenarios that will exercise the code to reveal potential faults. Formal analysis techniques are based on the existence of a formal specification and involve reviewing each level of development and proving it to be a correct implementation of the specification at that level. The techniques to be included for functional evaluation

were selected based on those specified as acceptable techniques in the Rome Software Test Handbook [Presson84].

Behavioral evaluation addresses how a component interacts with other components in the system, both software and hardware, to meet the functional, performance, and dependability requirements for that system. It is comprised of three levels of analysis: requirements (resource requirements and allocation concerns), design (decomposition issues such as functional allocation, granularity, partitioning and isolation concerns), and operational (performance and dependability characteristics under target conditions and load levels). The framework bases the behavioral evaluation on analytical and simulation models that are developed hierarchically. The behavioral evaluation techniques were selected based on the model-based design and assessment methodology developed in [Scheper91]. These models capture the analytical, simulation, and measurement information produced during the design, development, and use of the component. Within a fully model-based reuse environment, such models would be the basis for selecting components and determining what transformations were necessary and feasible to reuse the component. For systems at the upper range of performance and dependability requirements, particularly those with complex hardware and software mechanisms to achieve those requirement, such models are critical to the assessment process.

Certification Policy. The certification policy determines what is evaluated for each certification level and how by specifying the detailed evaluation objectives for each Primary level and the techniques that can be used for each of the Secondary levels. It also defines procedures for conducting the evaluations that specify input, output, process, and completion criteria for each technique used. Multiple certification policies can be defined in cases where more than one application domain needs to be considered. One certification policy was defined initially for this certification process. The evaluation objectives listed in Table III were selected based on Rome Laboratory's work in software testing, particularly the Rome Test Handbook [Presson84]. Particular requirements of a standard, or a consensus of the requirements of several standards, for a specific safety-critical application can also be incorporated into the evaluation objectives. The evaluation techniques specified by the current policy are listed in Table IV. The confidence ratings for the quality and functional evaluation techniques used to determine applicability to the Secondary certification levels are based on an integration of the three evaluation paths in [Presson84] (software category or domain, test plan or objective, types of errors apt to be encountered) to "choose the right technique at the right time to find the right errors". The confidence ratings for the behavioral evaluation techniques are based on the work done in [Scheper91].

Table III: Evaluation Objectives

Primary Level 1	Primary Level 2	Primary Level 3
Detect coding errors	Detect data deficiencies	Validate algorithms/equations
Detect program logic errors	Detect processing accuracy and precision errors	Validate program data requirements
Detect output format errors	Evaluate mission performance capabilities	Validate functional and performance requirements
Detect output content errors	Detect design errors	Validate design requirements
Detect data errors (content, format)	Detect design deficiencies	Validate software interface compatibility
Detect data I/O errors (transfer, communication)	Detect performance requirements errors	Validate hardware interface compatibility
Evaluate software performance capabilities	Detect functional requirements errors	Validate system operational capabilities
		Evaluate system interface compatibility

CONCLUSIONS

The multi-level certification process described in this paper specifies a method for certifying software for reuse in safety-critical applications such as those of the nuclear power industry. It provides a range of certification levels and associated evaluations varying from minimal inspections to rigorous testing and behavioral analysis, based on levels of criticality and component/system attributes. The criticality classifications established for this certification process address concerns of the nuclear safety industry for varying levels of assurance based on criticality assessment, while addressing a more generic view of criticality than that related specifically to safety. The use of secondary certification levels to determine a required level of confidence from the evaluation techniques addresses the concern that the applicability and completeness of evaluation techniques are limited by both the required level of assurance and specific characteristics of design attributes and operating characteristics of the system. The multi-level process provides effective management of resources (time and money) by directing more costly procedures to more critical components. Its assignment of evaluation level based on objective requirements facilitates the selection of reusable components compatible with requirements of

Table IV. Evaluation Techniques

Secondary Level 0	Secondary Level 1	Secondary Level 2	Secondary Level 3
Program Quality Analysis	Debuggers	Data-Flow Guided Testing	Assertion Checking
Reviews Walkthroughs	Structure Analysis	Performance Measurement	Symbolic Testing
Error/Anomaly Detection	Path Analysis	Assertion Checking	Formal Analysis
Structure Analysis	Domain Testing	Random Testing	Requirements Analysis
Path Analysis	Partition Testing	Mutation Testing	Design Analysis
	Path/Structure Analysis	Real-Time Testing	Operational Analysis
	Performance Measurement	Requirements Analysis	
	Random Testing	Design Analysis	
	Functional Testing	Operational Analysis	
	Real-Time Testing		

a new system in which it will be reused. Finally, the adaptability of the process allows future changes in technology and regulatory requirements to be accommodated.

ACKNOWLEDGEMENTS

This work was sponsored by Rome Laboratory/C3CB and the Office of the Assistant Secretary of Defense through the DoD Data Analysis Center for Software. The author would like to acknowledge the significant contributions of Deborah Cerino of Rome Laboratory to this work and the cooperation of DISA/CIM and DoD Defense Software Repository System staff.

REFERENCES

- [SECY91] Policy Issue Memorandum SECY-91-292. Nuclear Regulatory Commission, September 1991.
- [IEC45A] 45A/WG-A3(Secretary)42. "(DRAFT) Software for Computers Important to Safety for Nuclear Power Plants as a Supplement to IEC Publication 880." International Electrotechnical Commission Technical Committee: Nuclear Instrumentation, Sub-Committee 45A: Reactor Instrumentation, Working Group A3: Data Transmission and Processing Systems, May 1991.
- [ANS7432] ANS/IEEE-ANS-7-4.3.2-1982. "Application Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear Power Generating Stations." American Nuclear Society, 1982.
- [Wallace92] D. R. Wallace, L.M. Ippolito, and D. R. Kuhn. "High Integrity Software Standards and Guidelines." Technical Report NUREG/CR-5930, NIST SP 500-204, National Institute of Standards and Technology, September 1992.
- [Scheper93] C. O. Scheper. "Certification of Reusable Software Components." Technical Report, Rome Laboratory, March 1993.
- [Bowen85] T. P. Bowen, G. B. Wigle, and J. T. Tsai. "Specification of Software Quality Attributes." Technical Report RADC-TR-85-37, Rome Laboratory, February 1985.
- [Presson84] E. Presson. "Software Test Handbook: Software Test Guidebook." Technical Report RADC-TR-84-53, Volume II, Rome Laboratory, March 1984.
- [Scheper91] C. O. Scheper, R. L. Baker, and H. L. Waters. "Integration of Tools for the Design and Assessment of High-Performance, Highly Reliable Computing Systems (DAHPHRS)." Technical Report RL-TR-91-397, Rome Laboratory, December 1991.

6.8.1 Questions: Ms. Charlotte O. Scheper

QUESTION: MEL BARNES (ADA Technology Consulting): You described a certification system based upon functionality. We have also derived a certification scheme under the EPRI protocol. Have you considered other non-functional requirements? For example, the software has to be maintained, so maintainability is very important. Also, ease of use is important, so useability is a factor that needs to be taken into consideration. And the software will likely be moved to a different hardware platform throughout its life and so portability is an issue to be considered in any certification. Have you any comments on that?

MS. SCHEPER: I think you're absolutely right.

The framework addresses those other things through the component attribute table that in the paper. Well, that's not quite true. I was going to say the paper actually spelled out what all the attributes are. But, for example, there is a dependability attribute and the values for that are availability, reliability, fault tolerance, safety and security. And the notion is that those sorts of questions are more germane to selecting the actual technique and procedures to be applied than to the overall evaluation objectives. So, yes, we're addressing those, but in a way that's a little bit different from EPRI.

7 PANEL: APPLICATION OF WORKSHOP TO NRC ACTIVITIES

The final workshop session was a panel presentation followed by an open question and discussion period. This panel contained a mix of academic and industry experts discussing the risks of safety-critical digital technology, especially the use of safety-critical software within these systems. The experts were asked to address these four questions:

1. Are the proper issues being addressed?
2. What other issues need to be addressed?
3. Are proposed NRC regulatory positions complete and correct?
4. What are the considerations for further research?

The panel members were

Dr. John Knight (University of Virginia)
Dr. John McHugh (Portland State University)
Dr. Winston Royce (TRW, Inc.)
Dr. Joseph Naser (Electric Power Research Institute)

The moderators for this discussion were Mr. Franklin Coffman (Office of Nuclear Regulatory Research, NRC) and Mr. John Gallagher (Office of Nuclear Reactor Regulation, NRC). The panel addressed the questions by examining the overall context in which the questions were asked. This context includes but is not limited to the following:

1. The state of the nuclear industry generally and with respect to digital systems
2. The potential consequences of a software failure
3. The state of the art in software engineering
4. The state of the practice in software engineering
5. The reasons for the lack of good practice in software development
6. The current use of digital systems in related industries

Each of the speakers presented some of this information and proposed investigating other areas listed as a means of finding the answers to the questions. The speakers agreed that Answering the questions requires understanding this overall context, and understanding the context may, in turn, provoke new questions.

Following the panel, the audience had an opportunity to ask questions, challenge the panelists positions, and provide their own opinions. Many participants contributed to a dialogue on the future use of digital technology within the nuclear industry, and the open regulatory and technical issues.

7.1 Presentations

This section contains the edited transcript of the panelists presentations. This editing consisted of minimal editing to correct grammar and remove extraneous references to microphone volume, etc.

7.1.2 Presentation by Dr. John Knight

I will suggest a framework for discussing the questions. For example, on the question, are the proper issues being addressed? To a large extent, the proper issues are being addressed. The very fact that this meeting is taking place is impressive, but there is an orthogonal way of addressing some of these issues.

Are the proper issues being addressed? It depends on what the consequences of failure are. I do not understand what your applications engineers are trying to do or appreciate the nuances of nuclear control. My perspective is that of a lay person who has typical fears of failures at nuclear power plants. It is reasonable to ask what are the likely consequences of failure for the digital systems that you want to build, and ask are the proper issues being addressed?

Secondly, assuming that we know the consequences of failure and that they are significant, how can we avoid failures to as large an extent as possible? You cannot avoid all failures. How can you avoid as many as possible? It depends on how the application engineers use digital systems.

What other issues need to be addressed? Well, if there are gaps in areas with the potential for failure and we could address those gaps, then we should.

How can we be sure that digital systems are implemented correctly? The NRC's document intends that the engineers who build these systems proceed along a particular path. This path may not be appropriate, but how can we be sure that these engineers are following the right path? They are very competent and may have essentially no malicious intent, but there is often ambiguity in the directions they receive. There are plenty of opportunities for misunderstanding if the requirements are not well defined.

Another approach to the questions is to ask how can we use knowledge of what is important and what we wish to do to avoid the consequences of failure to make sure that development decisions are not left to chance?

My observations lead to two general points in this area. One is that there is a great deal known about software engineering which is missing in some of these discussions. There is a huge body of literature, for example, on testing which is a special verification technique, an experimental one. This knowledge should be brought to your problems. Although the existence of this meeting and the work at NIST and the NRC is very impressive, I would encourage you to seek out the existing body of knowledge in fields like software processes, software testing, and software design methods. For example, an analysis was presented of a popular programming

language, C++, in which various advantages and disadvantages were systematically stated. You gained, I hope, some understanding of the benefits and the disadvantages of C++. There are numerous other potential discussions that could be had in the programming language arena. It is a very large, complicated field.

So, I do not want to waste too much time here. Given this meeting and interaction, I think that things are very positive. I would suggest that there is a lot to be learned from the software engineering literature and that in order to answer the questions about issues and so on, perhaps these kinds of questions could act as an orthogonal driver on the discussion.

7.1.2 Presentation by Dr. John McHugh

In IEEE Standard 729, the Software Engineering Glossary, the definition for software engineering contains a rather imprecise set of words about software life cycle and related areas. In any reasonable standard dictionary, for example, *The Random House College Dictionary* or one of the *Webster's College Dictionaries*, the definition of engineering contains words to the effect that this is the application of science and mathematics to the construction of useful artifacts.

The striking difference between these two definitions is that IEEE forgot all about science and mathematics, which are key to any engineering discipline. I notice in the audience, also, some tendency to abandon science and mathematics when it comes to software engineering. I heard in various talks, discussions of expert systems, neural networks and so on, as solutions to problems. I do not know how to specify or accurately evaluate those systems in complex disciplines. It seems that when people who are not software engineers look at software engineering they grasp at the latest buzzwords and the latest catch phrases, rather than applying the same kind of engineering discipline that they would apply in their own engineering area to the problem. This is one of the reasons why we have a lot more problems with software than we seem to have with the products of traditional engineering disciplines.

As digital systems with software go into reactor instrumentation and control systems, we are going to have to apply the same kind of rigor to those systems that we have traditionally applied to the mechanical, electromechanical and hydraulic systems that we are replacing.

I see some abdication of leadership in that the people who operate power plants make it very clear that they are not in the business of doing research into ways to build them. For the most part, I do not see the people who have built these plants in the past doing research into the best, safest and most appropriate ways to control them. I see people saying, "Hey, we have these controller lines that you can certainly put into your nuclear plant," but there does not seem to be any specific analysis that looks at the kinds of risks that might be different in those cases. I see a need for research not only in the safety and operating control areas. I suspect that there is a potential for controls based on more elaborate models than we have had in the past to provide higher degrees of efficiency and operation; however, better control models are realistic only if they result in as or more reliable controls than those used in the past.

What I do not see is anybody taking the lead in those areas, and it may be time for somebody to take some prescriptive, rather than advisory, steps. The NRC contends that it is not their role to tell the industry what to do, but it does not seem to be the industry's role to find out what is appropriate. Somebody needs to take the initiative if this area is going to get off dead center.

7.1.3 Presentation by Dr. Winston Royce

I want to stress one point and provide a few examples. There is a remarkable tolerance for errors in software by all parties who use them, build them, buy them, and sell them. Stringing together logical elements line by line is error prone work. One of the previous speakers made a remark about retaining mathematics and science within the definition of software engineering, but stringing together the right, best, or acceptable lines of logic basically owes nothing to mathematics, physics or science. Logic is man-made. It is not based on the laws of nature or mathematics. People who are good at mathematics and good at science are very often good at stringing lines of logic together, code in other words. This means that we software people have no higher standard to adhere to. An antenna cannot violate Maxwell's equations. Airplane designs cannot violate LaPlatha's equation. There is no comparable test for most software. It is built by men and women, and men and women have to judge its correctness.

I have tried to decide why there is this remarkable tolerance for error. One partial explanation, is that it is very hard to sell software at a high enough price. There is a tendency for the buyer to accept a lower price and the seller to sell at a lower price because of the nature of competition, DAFOs and Federal procurement. The result is that the price is too low. Tom DeMarco wrote a remarkably accurate, and kind of humorous, article on this in *IEEE Software* two issues back that I recommend you read.

The legacy of this nationwide long-term low-balling process is that there is not enough time or dollars to get the errors out of the software or to go into research. The universities generally do not do research in error creation, detection or correction. This area is under-funded. The software builders have been low-balled down so that they cannot afford to use advanced error finding through techniques like better requirements analysis, traceability, using tools to find errors, or configuration management.

There is plenty of investment in software design and coding techniques. Those are the only two essential features of software development. The way we manage software development in this country, at least for the DoD/military world has been to minimize the investment, but not in those two areas. Testing and other error finding methods have really suffered. The legacy we are facing today is a partial legacy of that sort of tolerance for errors that exists in those who build, buy and use software. One of the most important things we can do is stamp out that tolerance, and move the industry and research community along to do better.

The error problem has a negative consequence in that you have got to judge human beings in ways that they do not like to be judged and that we do not like to judge them. We basically have a tolerant, democratic view that people ought to be free to create and build things, but

building software requires intolerance. We need to move to some method of licensing people to prove they are qualified to build software in an error-free way, and culling out those who cannot work that way. Some of the most creative software builders I have seen have also been some of the biggest blunderers I have ever dealt with. There is not a correlation there. Creative people generally are pretty good at being error-free, but there are a few that do both, and that has got to be fixed.

7.1.4 Presentation by Dr. Joseph Naser

We have heard at this meeting that there is universal agreement on the advantages of using digital systems. We see these advantages and the reliability improvements over analog systems. We certainly see that as analog systems are getting older they take more effort to keep going, and we have questions about their reliability. I think there is a general agreement that we should move to digital systems because they offer something good.

At the same time, there is a universal acceptance that it is important to implement digital systems safely in our plants. There is an additional factor; there needs to be a cost-effective assurance of safety of the implementation of these digital systems. If we take a look at the reality of the situation in the utility industry in nuclear power plants, competition is of prime importance. To be able to implement systems you have to do it in a cost-effective manner so that you can continue to produce electricity competitively; otherwise you are going to shut the plant down. If we agree that we want to put these systems that improve safety and increase reliability in the plants, we have to find a way to do it cost-effectively.

There are a number of issues that need addressing when we talk about the cost-effective assurance of safety. The first one is licensing stabilization. There was a discussion this morning on the importance of well defined requirements when you develop software systems. The same is true for well defined requirements if you are going to qualify those systems. We have to have well defined requirements for licensing stabilization, or for the licensing of systems, and the technical basis and processes for determining and using the requirements.

We need to remove the uncertainty of the effort required. If you say, "I want to put such a system into a plant," you have to have a good idea of the requirements for accepting it. This is the same as a software engineer who needs to be given specific requirements for a piece of software to quote you a price and guarantee you a deliverable. When you keep coming back to him and saying, "I need this, and this," then, as we heard this morning, the quality goes down and the price probably goes up. Changes in licensing requirements are costly. If the implementation is going to be cost-effective you have to have a good basis for your starting point.

A second area to look at is the emphasis on system behavior rather than on component behavior. We have to make sure that we are looking at meaningful questions and not trying to assure perfection, because we know we are not going to get there. We have to look at the consequences of a problem and not just the existence of that problem, to identify what really needs to be done.

For example, a control system for a valve may have a very complex controller requiring tens of thousands of lines of code to push this to an extreme; however, the controller may only be able to open, close or make the valve flutter. From a safety point of view, we need to assure that the plant is safe, no matter which one of those conditions occurs. We need to use techniques like defense-in-depth, and our industry has considerable experience with defense-in-depth in analog systems, while other industries have experience with defense-in-depth in digital systems.

We never expected perfect hardware, and we designed for that imperfection. We also need to do that with software. We do not want software that is no good; we are concerned with software reliability. We want that system to be good. We want the plants to be safe, and we also want them to be very reliable from an operational point of view. Even if unreliable software does not attack the plant's safety but only causes the plant to trip, this is very costly.

We also need a good technical basis for deciding how good is good enough. We need to know when we are increasing the safety of the plant and when we are just creating more paper. There is a feeling that we do not understand that border at the moment. What we really want is safe plants, and we want to make sure that what we do increases safety. We also have to do it cost-effectively, so we do not want to be just creating paper that does not help.

We also want to learn from other industries. We can look at process industries where there are similar requirements. Fossil plants, chemical plants, oil refineries, off-shore oil rigs, and so on have similar capabilities required and perform similar functions. These industries have considerable digital experience that we should be able to use. The safety and control systems in these industries are more complex than those in nuclear power plants, yet have demonstrated high reliability with digital systems. We have considerable experience that shows that digital systems can perform similar functions with high reliability.

When we look at the cost-effectiveness of safety systems in these other industries, we find that they are doing more complicated jobs with extremely high reliability for at least an order of magnitude less cost. We need to evaluate why and determine how we can work smarter in our industry without jeopardizing safety.

We also need a technical basis for commercial grade dedication of hardware and software. In other industries, there is digital equipment performing the functions we need with millions of operating hours of experience. If I want to have a really reliable system and I have a choice between a system with hundreds of thousands, or millions, of hours of operational experience or a system built from scratch using formal V&V, I think I have more confidence in the one that has proven its reliability. Choosing a system in use provides viable suppliers and people who will support you. Customized or nuclear industry supported suppliers may not stay in business. If you can take advantage of equipment that has been used by many industries, then you will have that supplier independent of the demand by the nuclear industry. You also have a large user base for feedback on the equipment.

We need a technical basis for deciding how to use compensating factors, such as operating experience, software development techniques used in the development of equipment, error reporting, et cetera, to compensate for not starting from scratch and not using a formal process.

We should also look at experience. The Department of Defense and NASA are going to commercial grade equipment for both reliability and cost reasons. We could learn from them how they justify it and how they are doing it.

An associated issue is the need for reusable software. We need to look for cost-effective acceptance criteria and a technical basis for reusable software.

There are always special technologies that you want to put in. For example, Commissioner Rogers and Director Beckjord mentioned opportunities for AI technologies. We also should be developing a technical basis for cost-effective acceptance criteria for these AI technologies. If we see opportunities for them, that means we want to use them, but it also means that we have to know the acceptance criteria for them. We need to look at expert systems, neural networks, fuzzy logic, all the buzzwords. Associated with these technologies are the operational aids which will influence how the operator runs the plant.

I want to reconfirm that I am not suggesting that we compromise safety, but that we find ways to operate safely in a more cost-effective manner and have a technical basis to justify it.

7.2 Questions and Discussion

This section contains the edited transcript of the question and answer session. This editing consisted of minimal editing to correct grammar and remove extraneous references to microphone volume, etc.

QUESTION: DR. LANCE A. MILLER (SAIC): You mentioned expert systems and neural nets. We just completed a bibliography on V&V of expert systems, and there are about 400 articles in there. There have been conferences every year since '86 on how to validate them. The exciting thing about that work is that the knowledge base is the primary application-dependent component of those systems and it's declarative in nature; therefore, it's open to, as someone said, not inspection but scrutiny. In fact, there were some very formal and very exciting programs, or formal proving programs, DEVA, CRSV, COVER, VERITES, just to name four of them, that can identify the non-functional defects. I think it actually points to directions that conventional programming might use, particularly the problem, for example, of validating data bases. So, I think this is actually very scientific and mathematical.

DR. McHUGH: I need to become more familiar with that literature because my information in that area is probably out of date but contains some real horror stories.

QUESTION: H. RONALD BERLACK (Configuration Management International): Dr. McHugh, 729 has been superseded by IEEE Standard 610.12 in 1990. I think that you would find that there was a revision to the definition of software engineering among a lot of other definitions in there that were added and modernized. I think also the revision to IEEE Standard 830 on software requirements also has a definition on software engineering.

DR. McHUGH: Okay. My understanding was that 729 is still operative. My version was never recalled.

MR. BERLACK: We'd be very happy, for a phenomenal fee, to send you the revised one.

DR. McHUGH: That's one of the problems with the standards is the fees are phenomenal for a person who buys them out of their own pocket.

QUESTION: MR. FRANKLIN COFFMAN (NRC): You said that we should be intolerant of errors in software. I think the question before the NRC might be to what degree should we be intolerant? How intolerant should we be of errors?

QUESTION: WAYNE JOUSE (University of Arizona): While sitting in attendance I've noticed a trend, especially from the software engineers, that the standard being applied to V&V, in this case, is as low as reasonably possible. Is this an appropriate standard?

DR. ROYCE: I don't have an answer for that. I can pose that question also pretty well, but I really don't have an answer. But, in lieu of no answer, let me tell you a current trend that's going on which reinforces the intolerance and maybe partially answers the question.

It has become common for many DoD software contractors to estimate the number of errors that will be in the software at delivery, and fairly extensive models are now being built. They're very like the cost models that emerged 10 and 12 years ago, in which contractors, by looking back through hind-sight at programs they'd previously done can estimate, or actually measure, the errors and project what the number of errors were inside delivered systems or at any phase in the life cycle. Contractors use these projections to deal with current systems they are on the verge of delivering.

And generally speaking, today an error rate of about 3 per 1,000 lines of code is tolerated. And 3 per 1,000 lines of code is going to kill people in safety-critical systems almost with certainty. So, it's something at least an order of magnitude under today's sort of consensus standard.

Kyle, as I remember, this morning quoted .06 errors per 1,000 lines of code as the standard for flight-critical NASA spacecraft operations. So, I think those are sort of the current standards.

DR. KNIGHT: In dealing with these numbers, the way that Win was extrapolating, you also have to keep in mind the area in which the resulting damage is going to take place. One of the differences between military aircraft, for example, and commercial aircraft is that the military aircraft is being flown by a pilot who understands that he's in a high-performance, possibly experimental, aircraft and he has an ejection seat. In commercial aircraft the passengers are often unaware that they are dependent on a digital system, and they certainly don't have ejection seats.

Now, on military aircraft, therefore, one would be prepared to take a certain risk with the software, perhaps .6, or .5, or whatever the number turned out to be, but when it comes to other systems it is often the case that we really couldn't even stand that level of defect and consequent failure rate, which is why I put that bullet on the slide earlier about the consequences of failure. The people involved, the aerospace people, or the nuclear people, or whatever, I think really have to try to understand exactly what the consequences of failure are. In a military aircraft, if something goes wrong, you can push a button and get out of there. You can't do that in a commercial aircraft. The differences in the consequences of failure are dramatic, and that dictates considerably to the way in which commercial aircraft software has to be built.

QUESTION: JAMES HILL (Northern States Power): Dr. Naser spoke somewhat from an industry perspective and I felt a certain disconnect for the past two days from a lot of people in this room because I am in an operating nuclear plant and a lot of people talking about software engineering without a lot of insight as to the application. But I'd just like to give you a little insight as to our perspective from an operating nuclear plant.

There can certainly be some very unique risks that may be appropriate to consider in using software, but I regard this somewhat as analogous to nuclear power's role. People regard nuclear power as a risk without considering the risks of alternatives, other forms of generation. Now, to illustrate that, we came on line back in '72 and '73, and our reactor protection system is initiated by a set of DC relays. Those relays are about 20 years old, and we just replaced

them. Now, what would be an obvious alternative for replacing relays? A programmable logic controller. Well, we chose to replace relays with relays because the implementation of microprocessor-based gear is very difficult to deal with.

Now, can there be a common-mode failure with hardware? Sure there can. With the relays being installed there turned out to be an epoxy mixing problem that was a potential common mode problem with those relays. That was examined and was found acceptable after it was carefully examined to make sure that the nuclear plant was operating safely. The point is that it doesn't matter if its software, or hardware, or firmware, whatever you put in, anywhere, there's always a potential for some sort of common-mode failure. We talked a lot about software risks without looking at some of these other hardware risks.

Another example is our analog gear which would sense reactor process signals and then send the trip signals to these relays. We have Foxboro H-line equipment and, as I said, we started up in '72 and '73, and the Foxboro H-line equipment became obsolete in 1968. That's when they switched product lines. We still have that equipment in our plant, and now they're beginning to replace it with Westinghouse equipment. These analog product lines are going to be much more expensive than microprocessor-based gear if we look at commercial grade products, but that's the route we're choosing to go at this time until things settle themselves out in the industry. The picture just isn't firm enough to make any decisions on which way to go yet, and so we're looking at that as an interim approach.

Lastly, we have a microprocessor-based alarm system. We recently did have a failure on that. The problem was a fuse blew in the power supply. Well, there's another hardware problem. My personal perspective of problems with microprocessor-based equipment is that many of the problems that I see in the industry are hardware-based, just like the one I mentioned, or they're user problems. There has been a lot of discussion about configuration management. When this equipment is incorporated in the plant it's very, very important to understand how it's going to be used and to maintain it properly. But again, that's not a software problem; that's a problem of training and qualification of the users of that software.

After hearing Dr. Naser's comments and other opinions from a lot of very smart people over the last two days, I have a hard time seeing when that's going to come out as a product for the nuclear industry use. Even if it does come out as a product and it is a product we can use, it may still be too expensive for us to use. If we could afford to run up the Federal deficit like was done for portable weapons systems, for instance, I'm sure we could do anything we want, but we don't have that option.

I don't know what the answers are to the questions that are in the program, but I'm just trying to give you more of a perspective from a user who is going to be the eventual customer of the things that we're talking about producing in this workshop. Thank you.

MR. GALLAGHER: I might just add to Mr. Hill by pointing out that his plant was one of the first ones to undertake the installation of a modern digital feedwater control system, so he speaks with a lot of experience in dealing with this technology. It was a very successful program.

QUESTION: HERB HECHT (SoHar, Inc.): I found the workshop very stimulating and I would like to mention two or three items that I found particularly pertinent and then follow up on what the previous speaker said about cost.

First of all, we found there may be faulty software around but you can't tell it from Suzy's software. The point is not that we develop fault-free software, but how do you certify or how do you identify that software as fault-free. Kyle Rone, this morning, showed a very interesting graph of how as the cost goes up as you try to certify to lower and lower levels of reliability.

Secondly, we know about the requirements. In answer to John Knight's first question, the consequences are unacceptable. I don't want to quantify unacceptable. The answer is not that nothing can be done; far from it. The answer is diversity. You can produce two diverse systems at much, much lower cost than you can certify one to the extremely high demands of whatever standards you want to apply.

Third, if you apply these diverse systems maybe you learn something. It's not the statistics that we need to learn, it's the nature of the faults that are in there that we need to address.

Now, diversity was mentioned in Mr. Russell's talk, and diversity, by itself, isn't terribly meaningful. You've got to define what is diversity, how do you certify diversity, and what kind of diversity to require, for a given application. It's a big problem, but it's easier than to produce fault-free software.

There is one other article that goes with diversity and that is ultimately you have to have a selector or a combiner, or something, that distinguishes between these two or three chronometers that we may have. Again, it's not a trivial issue, but one that should be addressed, and I hope that these two subjects, the diversity and the combining mechanism, would be something that the next workshop can devote itself to.

QUESTION: GEORGE C. RUDY (NUS): I have several questions and the panel can stop and say it's too long a list, but I'm encouraged by this seminar. You brought to the nuclear industry good insight in software. But let me ask you a couple of questions.

We now have a three-headed horse, I guess, because where is the integration of these things? You brought to the party very, very good insights and a good admission that software has errors, and so does engineering, or hardware engineering. You bring to the table the extreme experience you've gained through DoD programs, as an example the NASA programs. The thing that's really missing in all of this so far is the systems integration, bringing people together, bringing the nuclear experts, the software experts, and so on together. That has been the big problem even with traditional nuclear programs. There has been a tradition in the nuclear industry--I've been in it for 30 years--that a program is ragged out due to a safety issue and something is delivered on the dock. It may be managed or configuration control is lost. We have examples of that in the industry right now on traditional issues, not anything to do with computers.

The other side of the issue is it's been a tradition in the industry to meet schedule regardless of consequences. We drop it on the dock and we fix it in the field. It isn't going to work with these things. The mission is different. If you had the airplane out at Edwards and it fails, you lost your crew or you got out. You can't take that risk. We can't tolerate any upsets.

So, the issue that I ask, and it relates to several of the presentations, is a concern I have because there was a message delivered that there are errors and you can expect all these errors.

Is the error on a subsystem level? What about fault tolerance system configurations? And I guess the question on error rates, is it a system error rate or a subsystem error rate?

DR. McHUGH: Well, I'll give a stab at some of these things. The error rates in software that we're talking about are what I guess I'll call "wholesale" error rates. I mean they're basically measured in terms of the number of errors that are found, or believed to exist, in a certain number of lines of code, largely independent of how that code is structured into a system. What that means is that those errors could be absolutely deadly or benign. Unfortunately, until you figure out where they are with respect to the functionality that you have to deliver, you don't know. Now, that's one of the things that makes it particularly hard.

Some of the kinds of testing that you can do will tend to focus on demonstrating that the functionality that you have to have is more thoroughly tested than the places that are seldom entered, and that may or may not be good. It depends on when you enter those places that haven't been tested very well.

Basically, given that any of the components in the system are prone to failure, you have to develop some kind of an architecture that will tolerate those failures. One of the things that we have seen with the advent of digital computers is a tendency to put more and more eggs in one basket. For example, if we have a single processor that is supporting a number of diverse processes, errors that appear in those individual processes may very well be non-interfering in the sense that a process can produce wrong answers while the other processes continue to produce right answers. On the other hand, an error in the underlying operating system that is managing resources for all of those processes is a common mode error potentially to everything that is running on that physical processor. When you look at building a system so that a single point of failure cannot cause a system failure, there are a lot more things to look at because the nature of the particular failures occurring in the software can either be such as to take down many things and be a real single point of failure, or it could be relatively benign. We don't know completely how to deal with these things.

We do know work that John has done in attempting to get design diversity in replicates of software and that the cases where software developed independently to perform the same function fails under the same inputs are more than random occurrences would lead you to believe. The statistical independence that says that if I have two identical circuits sitting here on the table with nothing connecting them, that the chance that they are both going to stop at the same moment in the absence of something like an EMP that is going to nail them both is very, very small. We find in software, where the errors really have to be looked at as distributed through the input space rather than distributed through time, that we get more common failures on inputs.

Now, intuitively it's reasonable for this to be the case. There are portions of every problem that are harder than other portions, and I believe when you look at the data that we have, the hard places tend to collect more failures. It doesn't mean that the same mistakes were made in both cases; it just means that mistakes were made in the same cases. And those are the things that are particularly difficult to deal with.

I suspect that diversity is going to mean dealing with separate phenomenologies for the ultimate source of the input data and for making certain that the ways in which the data are processed are truly different. Given that most of all our software engineers seem to be trained

out of the same bag of tricks, there is a possibility of a common-mode failure due to a widespread way of thinking about things. You can't really look at two pieces of software and say that they're diverse.

MR. GALLAGHER: Certainly I would like to add that as Mr. Russell and Mr. Beckjord said and as is in your handout, the framework for the NRC Branch technical position emphasizes both quality and diversity. It contains 4 steps at the end which are directed towards one way of achieving diversity. You also heard Mr. Beckjord say that he views this as an area where a lot more work needs to be done to establish whether or not functional diversity, using the same program structures, is adequate, or do you have to go to different types of program structures. Those are questions that have to be asked. This is one of the efforts that's underway in the regulatory position.

FRED PAULITZ (NRC): I have a few comments there. I suppose if it wasn't for the emphasis on advanced reactors, all this digital equipment going into reactor protection systems may not come into being as such. There has been a lot of applications in feedwater control, feeder-drain control, and other non-safety portions of the plant. So, it looks like the horse got in front of the cart here a little bit. These systems are designed, and people want to stick them in there. Now, we're going back and saying what do we really need to prove that they are reliable.

In the seminar, the heading says, "Digital Systems Reliability," and nobody has said anything about hardware. It's all software-oriented, and those are definitely related in that hardware does sometimes effect software.

The speaker yesterday said that if you put things in the right little compartments that we're less apt to get in trouble, and that if you run parallel who knows what's going to happen.

And then you mentioned the experience of a transfer of things that are already out there, which I suspect NASA and DoD and the rest of them have, that we, in the nuclear industry, ought to start transferring it over here somewhere and being as smart as they are, I guess.

Now, in the aircraft you mentioned you're saying there's two guys in the aircraft and you bail out and then in commercial you've got 300 people. Just recently, for the last year or so, every time I get in an aircraft the first thing they tell everybody is, "Shut down those portable computers, radios, et cetera," because one day that cracked up a plane because of those. But the same methodology that you talked about for developing software is the same thing that has been, and should have been done, in designing a whole new compartment in the first place. I noticed a lot of precepts were good, and that's what we need. Thank you.

SID BHATT (Electric Power Research Institute): I have several comments I want to share with you, and I think we have many experts who shared their wealth of information and some honest admissions of guilt, et cetera. But they are missing some points. If I were to set up the workshop and I had some experts, I would try to relate to the problems of the end users that I'm trying to reach.

Point number one. Diversity, language, and programming. Very popular in the software area. The utility industry and the Electric Power Research Institute, they had a program, a data emulation programming experiment, in 1978. In the utility industry in Europe they also have

followed up a program with DoD and we have voluntarily shared some of this information. In this industry we have been sensitive to public safety and we have been sensitive in knowing about what the technology is and what is good or is bad, or where we have to watch out.

In airplanes you have to keep it flying, or you've got a problem, so-called graceful degradation, but you're still flying, and your control system takes care of it. All people who contributed to the nuclear power plant program, they said, "Hey, we've got a control system and we can institute something else, separate from the control system, called a protection system." But they need a safety net under the control system also, and we need to distinguish the safety net which exists. So, the protection system is, and has been, a safety net activity for a long time. By nature they are simple. Protection systems are able to implement and do some simple relay logics with analog equipment. What I'm trying to convey is that the complexity of that program, including the analog logic, is simple, and there have been some checks put into those kind of design-in-depth in nuclear power plants. So, don't simplify and dramatize this by giving us something comparable with airplanes that DoD has safety systems on and commercial doesn't.

My fourth comment. Go around and look around at what's happening in the aircraft industry also. There is Standard 178. They actually share information internationally, and many parties are investing in this including the commercial enterprises. The international airline body worked with Lockheed and others to come up with 178. They understand the limitations.

We understand that there are limitations in software, and we are not asking for perfection. What we are looking for is not to discount new technology, but we will be using it with care, utmost care. We don't want to simplify it, and say that there is a common mode demon running around. We have lived with those kind of things, handled different technologies through diversity. Both industry and regulatory people understand that we have to live with those kind of issues for a long time.

My fifth comment is that this is a complex decision making process, and we cannot really dilute the emergency, timeliness, and criticality of being able to utilize this technology and handle it in such a way that it is in the U.S. interest, but the protection system has been implemented in digital systems in other countries. For example, France and India have very similar operations, and they have plants operating with digital safety systems. They have learned how to handle technology and manage the critical safety, technical and management issues.

And these are some things which I've observed and I hope I haven't hurt anybody's feelings, but we will progress if we try to focus on the technical basis that you've mentioned instead of simply trying to find a solution within this workshop. So, I say, pay attention to Al Sudduth who said, "You guys are talking, and we're already dealing with the process." The industries know how to handle this kind of thing. Don't kid yourself. We are not born yesterday that this is magic to us. The people at the utilities know about the technology. They can handle it. People do come down and manage things in the nuclear area also. The question is how do we manage these three circles, the utility industry who needs to do things right so they can have a peaceful sleep, and the regulatory people who have to manage and certify so they can also feel comfortable. We would like to get that technical comfort level so that it is reasonable. We are not asking for error-free code. There is going to be some errors. The question is how do we manage and where do we go from here?

MR. GALLAGHER: Thank you, Sid.

You raise the issue of inversion programming. You brought that subject up. And I was wondering if Gustav Dahl is still here? The reason I looked at him is he certainly has done a fair amount of work in this area, and it's a subject that keeps coming up, and I don't know whether he's willing to give his view on the status of that or not.

GUSTAV DAHLL (OECD Haldon Reactor Project): We have pursued several research problems in inversion programming, or actually diversity based on the same specification, which is not exactly the same as diversity in the wider sense, which includes functional diversity. The findings we have don't differ from what has been explained earlier. You get some gain in the reliability of the system, and our research shows that for some types of failure, but not all, we get again what you would expect if it were completely independent. First of all, we have, obviously, common-mode failure caused by, for example, common specification. Another problem is error masking in which several different faults are grouped together so that the effect of the diversity is lowered. This problem doesn't mean that you don't gain anything by diversity. Roughly you get, for example, a 10-fold increase, but not 10^3 . So, there is a gain, but it is not a sufficient way of gaining extremely high reliability.

MR. GALLAGHER: Okay. Thank you. Yes. I think that was a point that Herb Hecht also made that you have to consider who is going to be the arbitrator in how you make a decision. I think it's very important, and Dr. Naser brought this out, that the cost-benefits certainly go far beyond just the design process. They go through the whole software life cycle. So, if by doing something you're adding an extra burden to the latter part of the life cycle, then the cost-benefits are very doubtful.

DR. KNIGHT: Well, based on what a couple of the questioners from the microphone were saying, it would appear that I've upset some peoples' feelings, so let me start by saying that was not my intention at all, and let me go through the five points that were raised by the last speaker from that microphone.

Both speakers seemed to be concerned that I had pointed out the volume of existing literature. That wasn't meant to be an insult to this community. It was merely to point out that in fact outside of application domains, but just within the mainstream software engineering literature, there really is a great deal that you might be unaware of. I know little about nuclear engineering, and I would not expect to have access to the literature. If there are nuclear engineers here I just wanted to point out that, in fact, a great deal might be available to you by looking in the mainstream computer science literature rather than in the application domain.

The second comment was about diversity and the record at the EPRI. Well, I know that literature fairly well and I wasn't aware of work done at EPRI in '78, but all I would say about design diversity is it's an extremely complicated topic. It is by no means anything which one can adopt as a simple, or even a moderately complicated, translation of the way in which it's used in hardware. I would caution people that the use of design diversity is something that has to be looked at very carefully.

The third problem was with continuous operation and the analogy that I used. As I explained yesterday, the only application domain that I even know a little bit about is flight

control systems, and so I tend to use examples from there. All I was trying to point out through the example that I cited earlier was that there are differences in the consequences of failure in flight control systems and that, in fact, different degrees of risk are undertaken. I wasn't implying that the fact that those systems have to operate continuously is something that you should take into account. It was merely a small example to try to elucidate the point that in the domain that I understand anyway there are different consequences of failure.

The fourth point about DO-178A and DO-178B, those are existing standards. I'm aware of the existence of both of them, and they play a role in the aerospace industry. I don't think they have a role especially to play in your industry, but certainly I'm aware of them.

On the fifth point, about the existence of foreign nuclear plants, as I understood it, running successfully, I don't have any doubt that that's the case. I'm perfectly confident that you can cite plenty examples of that form and that there is valuable experience there. And again, I don't know what I said that caused at least some subset of people to have their feelings hurt. That wasn't my intention.

I guess a lesson which I would like to bring out here is from my perspective as a humble software engineer. Software engineers don't know much, and building software is hard. We don't know very much about how to do it right. Perhaps, the real lesson in all of this is you shouldn't listen to anything that I've been saying.

QUESTION: WAYNE GLIDDEN (Nuclear Utilities Software Management Group): I'd kind of like to pick up on a comment that Mr. Hill made and extending that to another comment. I love hearing a disconnect in this conference. I felt that since we got the brochure that we have a lot of experts and a lot of people at the leading edge of software development and testing, and that's great, but the problem is that the workshop is supposed to be on the leading edge as it impacts the nuclear industry. I don't hear too much of that.

If the nuclear industry has got to put the effort into software that I've been hearing over these past couple of days for what I really see is a minimal increase in the safety of that plant that's not cost-effective. If we won't do it where our analog systems break down and we can't replace them, then we will be shutting plants down, and it won't be cost effective. As was said, they've already canned a couple of digital replacements. There has got to be a realistic approach. We've got to take the information we hear from academia where most progress starts and get somebody to translate that into what it means to the nuclear industry. I think that we've missed that in these two days. We only had two speakers from a utility in these two days. I believe that this conference is great because we've heard what's going on, but there needs to be one more step before we can really answer the questions that have been raised for this panel. For example, are the regulations going to be good? In light of what we've heard, the regulations are going to become more restrictive and will not be beneficial to the utilities and the nuclear industry. I'll be out of a job, a lot of you from the NRC will be out of a job too because we will have nothing to regulate.

MR. HILL: Yes. Diversity was discussed, and I didn't touch on that. Again that's a consideration once it filters down to the user. Defense-in-depth, I feel, is very important for diversity. If we talk about a diversity of product lines, for example, the feedwater control with our Westinghouse WDPF, for an engineer to be qualified to properly maintain the configuration

of that system, he or she has to go to a 6-week school, and the technicians have to go to a 6-week school. In addition, there's warehousing of parts to support that system. When you add another diverse product on a site, that more than doubles the problem and I feel, again, that one of the major roles of the utility is to properly manage the configuration and taking care of that software/hardware system. It becomes very difficult to do when you start adding diverse product lines. This problem should be considered.

MR. GALLAGHER: Yes. That was my comment that you have to evaluate anything you do with respect to the impact on the whole software life cycle, and you pointed out that the maintenance in the field is a big item there.

QUESTION: RAYMOND J. RETTBERG (GPU Nuclear Corp.): I get the impression through this whole conversation and this last two days about all the difficulty with keeping errors out of software. I wonder if we're talking about a different sophistication or complexity of software? Specifically I'll pose the question, if we replace a simple, purely hardware system, such as a 2-channel system with a microprocessor with software in it to do the same function, do we inherently have new problems, or more serious problems, or something that we really never had before?

DR. McHUGH: I guess I would have to ask the question of what were those channels doing?

MR. RETTBERG: Let's say it's a purely logical function, or say it's a combination of logical function and measuring some analog inputs to decide the trip functions for example.

DR. McHUGH: Just simply a comparator? No integrators, no filters, no nothing like that? I guess the question I would ask would be why would you want to replace something like that with a computer? It seems like the system that you have is already very simple. You're going from parameters that are measured in a continuous domain and effectively a continuous decision process to one in which you have to sample and sequence them through a set of repetitive instructions that it is going to be substantially more complex because of the structure of slicing it up and turning it into a discrete domain and processing it in a large sequence of steps, rather than a continuous flow.

Now, if that's all that it is doing, my intuition, if I were building a system like that, would be to build something really simple on bare-bones processors with no operating system and no nothing, and it wouldn't be much more complex, but it would still, in terms of parts count and potential for hardware failures in the system, would probably be a more complex piece of equipment, just simply because of the shift of paradigm from continuous measurements to discrete measurements and time stepped chunks.

On the other hand, if you say, "Well, I'm going to buy a general purpose computer, with an operating system on it and so on," you've probably added several orders of magnitude in complexity in order to provide the infrastructure to do the simple job that you had to do. But it almost seems to make sense to do this with custom designed logic that isn't really a processor, or to stay over in the analog domain, if that's possible. I gather the real problem that we're

talking about is that nobody is making the analog equipment anymore, even though it was perfectly adequate for the job.

MR. RETTBERG: Right. Or there may be a model, but there may be additional functions beyond the critical functions that are the core of this thing that truly affect the safety of it. The question was what is it fundamentally that we bring into this thing that somehow puts us into a greater risk than we were before? Because to me that seems to be underlying a lot of what I'm hearing and I'm not sure that in a simple system that that's a valid premise.

DR. McHUGH: I think one of the things--and John wants to say something else here--but one of the things that I will say is that there seems to be a temptation to say, "Look, we've got all this computing capacity that we're not really using to do that simple job. Let's use it for something." That may be the wrong answer, because as soon as you start having that processor doing 37 dozen other things that aren't part of that mission on the grounds that the processor is there and it has the capacity to do it, then the complexity goes sky-high.

In the automotive world we have seen some problems when spaghetti code is in a processor because it not only runs the fuel injection but it runs the transmission and the windshield wipers. Separating out those functions and keeping them straight, some of which are safety-critical at all times and others are not, has turned out to be a major mess.

We haven't had a recall in the automotive world, I think, based purely on software yet, but if we do it'll start at about \$20 million dollars or so and go up from there in terms of the cost to the automotive company.

But, if all you want to do is replace that comparator with a computer, and the chip is logically a \$1.98 part now (a nuclear qualified one may be \$198,000) and, if you dedicate it to one simple function and forget the excess capacity it probably isn't that much more complex, but it isn't off-the-shelf either.

DR. KNIGHT: I wanted to start answering that a little bit by asking a question of this community here. You've heard a couple of numbers quoted of defects per 1,000 lines of code. Now, when you build the kinds of plants that you build, I imagine you put very large amounts of wiring, very large amounts of piping and very large amounts of other things in them. Does anybody ask you how many defects per foot of pipe you have?

SPEAKER: Yes.

DR. KNIGHT: They do?

DR. NASER: Yes.

DR. KNIGHT: What's acceptable? Well, this is revealing yet again how little I understand of your business, but the thing about software is that the defects are design defects that are there from the beginning. Software doesn't break. Pipes, as I understand it, do break. You have to worry about that as a maintenance issue over time. You have to have defense-in-depth against pipe breakages, but are there design defects in the piping that you have to worry about and in the cabling associated with power distribution?

(No response.)

DR. KNIGHT: I really didn't think there was or I wouldn't have asked the question.
(laughter)

DR. McHUGH: You lose, John.

DR. KNIGHT: Sort of. You can then begin to deal with this software question immediately. If you have so many thousand feet of pipe and you're prepared to accept .6 of a defect per foot, or per 1,000 feet, or something, then the software has to be probably comparable. In other industries the requirement when replacing an analog or a mechanical system is that the digital system meet the demonstrated performance. So, for example, in replacing hydraulic and electromechanical systems, the FAA's requirement is that you make the digital replacement as good as the history has demonstrated the mechanical systems to be.

Now, because of the many thousands of operational years of exposure, the numbers are in on the mechanical systems, and that's where that 10^{-9} figure comes from. I think in dealing with that kind of replacement, that might be the kind of strategy to take. What's the demonstrated performance of the old system? Can we meet that in a way that is reasonably scientifically convincing?

DR. NASER: Actually this is a comment for John, and this is sort of a clarification. One of the reasons why you hear people saying that they want to put in excess functionality really isn't because you've got the computer there and, as he says, "I've got it there, why doesn't it do things?" There may be some of that, but I think that if we look at the real basis, we find real advantages. For example, if we look at problems that have occurred in the plants which have caused plant trips, frequently that's been caused by the human, and it's the human doing things like checking the system. So we'd like to put in additional functionality which does things like self-testing and knowing when we need to do calibration, instead of doing calibration in a periodic manner and doing it when we don't need it and perhaps causing more problems. I think there are some good reasons why we want the additional functionality as well.

DR. McHUGH: Let me just reply to that. That's perfectly reasonable. The question that I was asked there was if I want to just simply replicate the functionality, and I would argue that probably is a very straightforward thing. I think you're right. You have very good reasons for going further. The question is, where on the reliability versus complexity curve do we come when we do that and do we get ourselves up into the area where we start really worrying about whether that extra functionality is safe especially if it has been put in to make the plant run more smoothly or for economic reasons as opposed to making it safer. I think there are some trade-offs in there. Some of that software may very well serve to make it safer, but if it was safe enough before, do we get unsafetys from one introduction of additional functionality and more unsafetys in another area? I don't know. This is something that I want to look at more.

DR. NASER: Actually just one last comment. I think you already gave away what I was going to say anyway. I think some of the things we're looking at, we think, are ways to guarantee or try to assure safety. For example, there is a dynamic test system that the gentleman from AEA made some reference to earlier. This test system makes sure the system still works by putting signals through to actually test its capability so that you know the trip will work when you want it.

MR. SUDDUTH (Duke Power): I guess one of the problems I have, since I seem to be one of the few people who actually builds digital systems that go in power plants, is fitting the paradigm under which I develop software into the paradigms that I've heard over the last day or so. Let me real quickly run through what I do.

I go to a vendor like Westinghouse and I say, "Sell me a system." What they sell me is hardware, processors and I/O networks and things like that, but they also sell me software. They sold me an operating system. In the case of Westinghouse it's actually RMX, if you all know what that is. That is an Intel-created real-time operating system. They also sell me a whole library of functionality that says, you know, "These are our little modular blocks and all you've got to do is string these things together and you've got a control system." And that's exactly what I do. I make a drawing which represents the manner in which I'm going to string together those Westinghouse library blocks in order to create a control system. That drawing can then be shown to any number of control engineers in my plant, and they can all tell me exactly what that control system is going to do. It's a completely unambiguous description of what the system has to do.

I then create the software that implements that using all these little Westinghouse blocks. I can actually parse that software back into a drawing and make sure I did it correctly. Then I put the system in and test it until I'm satisfied that it works.

I'm sitting here saying, "Well, gosh, I must be doing something right because the systems work," but I don't do any of the things that I've been hearing for the last day or so in order to do that. So, help me figure out, is the thing that I do so far out of the mainstream of what you all are talking about, or am I doing what you're doing, except I'm doing it differently?

DR. KNIGHT: Well, first of all, you're doing exactly what I think John and I were saying yesterday; your diagrams, in fact, are a formal specification that you can communicate between yourself as a computer engineer and the controls engineers, and you have a very advanced notation, in fact, compared with what most people use. That provides you with a great deal of security against ambiguity and misunderstanding. Then you are depending upon something which is being provided to you by a manufacturer who presumably has put a lot of effort into various forms of verification of those items and has a great deal of field experience with them and so you can put a great deal of trust in them. Then you're going ahead and doing testing of those items yourself anyway.

There are nuances to this which we could discuss, but I think what's happening is we're using one kind of terminology and perhaps it isn't quite the one that you use, but there is a great deal of commonality between what's going on.

DR. McHUGH: I think the thing that is important is that Westinghouse has done all the testing of what's inside those blocks so that you can trust them without looking inside them. It looks like there's a lot of evidence that they have. If the testing has been done properly, then what they have done is they have taken you at least one, or two, or three, steps away from the bare bones of the machine in giving you an abstraction on which to base your stuff which is much closer to your problem, and that's exactly the right way to divide the labor on that.

MR. GALLAGHER: I may add something to that. I used to work in the place that he's talking about before I joined the NRC, and our general manager never understood why a fossil plant took so few engineering hours because mainly the engineering was to assemble the system down on the basement floor in accordance with the specifications, and then the person who was going to use the system would come in and put the software in and check it out.

Well, for a nuclear plant there is a few things like safety analysis and lots of pre-justifications one has to do that would prove to some reasonable level of assurance that what you were going to build will meet all the safety requirements and regulations, and that takes an awful lot of extra engineering that you don't ordinarily find. I know, because every Monday I used to be confronted with this difference and it wasn't easy.

DR. NASER: Actually I'd like to make one comment. I think what you've said is certainly true of the environment. I think probably the thing we need to question ourselves is how much extra do we need. I think there is no question that we need some extra, but the question we have to address is to determine how much more we need to do. I think this is one of the issues that perhaps should be looked into at the NRC. For example, if we look at the WDPF systems that are going into the fossil plants, going into petrochemical plants, going everywhere else, they've really proven themselves to be very reliable. Why are they no good for nuclear plants?

MR. GALLAGHER: I think maybe there is somebody here from Westinghouse that can answer this, but I think if you look at the Sizewell that a lot of the balance of the plant was done exactly in that way. So, when one speaks of nuclear you have to speak of the safety system.

DR. NASER: Right. Thank you.

QUESTION: DAVID HOLCOMB (Oak Ridge National Labs): One of the comments I'd like to raise here is that I felt a rather strong disconnect with most of the topics that the speakers have brought up. It seems that the workshop has overlooked the basic fact that the control and the safety functions are physically separate in nuclear power plants. The nuclear power plant safety functions are much simpler than the safety functions of the systems in spacecraft, in flight controls, and in medicine. It seems unlikely that higher level computer language, things like C, C++, ADA, and all the other object-oriented and advanced computing languages, as well as the advanced microprocessor architectures, are going to be used in the safety-critical protection systems for nuclear power plants. What using them would require is for us to start qualifying the circuit layouts of these advanced microprocessors, the operating systems, and the compilers, which seems to be well beyond the scope of anybody's financial resources. It's more likely that we'll do something with bare-bones, basic hardware and very simple direct ribbon watching with no operating system. It just seems that advanced software and hardware architectures are going to be restricted to the control systems to get the maximum efficiency out of our plants and stick with as simple as possible in the protection system.

DR. McHUGH: Let me take a shot at this because I think what I've been hearing is a substantial disagreement between parties on the nuclear industry side of this. Some of the things that I have seen proposed and under construction for safety shutdowns on reactors are far from

as simple as you've described. I would suggest that taking that approach is going to lead you to exactly the same situation that you are in with the analog systems now. If you go and put a particular small bare-bones processor and some dedicated controllers--ADA-D, et cetera--in, in 15 years that processor isn't going to be available. You will have programmed it at a level where it will be necessary to, in effect, rebuild that entire board up from scratch the way you would be doing now. This is the way you would probably build a new analog board if you could get new analog components.

But at the level of picking something which is specific to a particular processor and particular converters and controllers and so on, my guess is that it's going to become obsolete even more rapidly than the analog equipment has become in the current plants.

Now, it may be easier to re-implement it every 5 years than it is to re-implement the analog systems, but I think you're going to have the same sort of problem. It's only if you get away from a processor-specific system that you have some chance of having the system that you build outlast the platform on which you put it, and it sounds like that's what has gotten the industry into the situation of needing to convert from analog to digital for the safety shutdown systems at the current time.

MR. HOLCOMB: One response that immediately comes to mind to this is that coding for advanced microprocessors is just as processor-specific. It is often more so since much of the coding takes advantage of very selective, often hidden, features which much of the rest of the community won't know about. Importing the code on the next generation of advanced microprocessors, which certainly have a very rapid turnover, is going to require another of these huge efforts. People have mentioned here that they have 1968 hardware that they're updating 30 years later. Let's stick with the bare-bones processor, and maybe in 15 years then you can go to the next generation of very simple bare-bones digital systems with the added advantage of almost desktop manufacturing of many of the processors. Some of the digital processors aren't that difficult to make, and it may be possible to get a sole-source of supply of some of the simple digital processors for many years to come.

DR. McHUGH: It's possible. I honestly don't know. I think if you build your system at a fairly high level of abstraction that you can probably count on the underlying libraries then you depend on to communicate with the processor architecture itself being available. Whether they will qualify for these kinds of applications, I honestly don't know. I'm not sure that there is a panacea in any of this. As of a few years ago, I think DEC would still build you a PDP-1 out of their Traditional Products Group, for example, if you had to have one, but it would have been an astronomical price. I suspect that 10 years from now the simple processors that are being used as controllers would have to be custom fabricated. I know ARPA has been looking at issues like this because the military has this problem in spades. They've got equipment deployed that has outlived the technologies with which the equipment was built, and they are faced with the problems of either complete re-implementations of systems, which cost billions of dollars, or building with obsolete technologies, which costs billions of dollars. They've been looking at ways to be able to have automated factories that will manufacture any old piece of stuff that they need to keep systems going, but that may or may not be an answer in this area. I have a feeling

that technology turns over too fast for plants that last 20, 30, 40, 50 years without planning on major refits of some kind every 10 or 15 years.

MR. HOLCOMB: I also then need to add that you should take a look at what is being proposed by GE and Westinghouse and some of what's going on, and what people are still proposing. I haven't seen any real proposals for very complicated protection systems. You see control systems. But the GE system and the NUMACs have very simple comparator logic. That's all that's sitting in there that they want to digitize. It's not that fancy.

DR. KNIGHT: Look, it's often the case this way, but I'm confused, because on the one hand the French plants were cited as being an example of digital control systems. I'm aware of one paper being published by that group in which they cited their development process, and the total size of that program is in the tens of thousands of lines. The Ontario Hydro people have been developing various kinds of control systems, and while I don't understand the details, given the presentation they made yesterday, they're not talking about simple logic. They're talking about something that must be tens of thousands of lines to warrant the attention they've put into it.

MR. GALLAGHER: One has to be very careful how you count lines of code, because some people follow IEC-880, which says you have to build it without any operating system, and then when you look at the total lines of code, it's much larger than a system built on an operating system. That's one of the reasons why you hear such large differences in numbers. Also, as Dr. Naser pointed out, people put into these systems code to try to get rid of problems that they have in the present system, for instance to aid in automatic testing requires bypass and other features, that add additional lines of code. There is a lot of interest in developing a safety kernel that would be a very straightforward simplified system and then add on the top of that these other features, and that's one of the things that people are looking at.

QUESTION: BRUCE MOORE (Data Refining Technology): Most of my experience is in the chemical processing industry with digital control systems today. On the subject that's just being discussed, one thing about a protection system is that even when you have that safety net, once that's in place, from a practical point of view you become very concerned about challenges to that safety net. A great deal of importance is placed on events that cause your control system to exercise that safety net. It then becomes serious to make sure that second layer works adequately, and I can tell you it's very uncomfortable to be a control designer and be the cause of a safety net being exercised. You always remember that experience the rest of your life.

The second thing is I wanted to talk about is on the issue of functionality. There was a question raised earlier: if I've got an analog box and I'm replacing it with a digital box with the same functionality, what's the big deal? Well, I think one of the big deals that's been talked about in software has to do with the difference between external functionality of a box, the way it behaves to the outside, and internal functionality of a box, what happens on the inside. External functionality is usually the domain of the systems engineer or the applications person; internal functionality is often the domain of the software person. One of the very difficult problems with building software and having it useful and cost-effective over a long period of time is to design and implement it so that a small change in its external behavior is

accommodated by a small change in its internal behavior. Much of the effort that's going into software research is in understanding how to do that and how to make software so that 5 years after the system has been designed, if someone comes up and wants to change the way it behaves externally, that internally people can do it without a major huge effort.

If you get back to that simpler question and you say, "I'm replacing an analog box with a digital box," well, you're putting a whole new world of internal functionality there and that does create a lot of complexity. Thanks.

DR. NASER: I think there are two things we need to concern ourselves with when we're talking about this additional complexity and functionality when we're looking at functionality internal to the box and external to the box. If we're looking at a safety system, I think, where we've already looked at the need for defense-in-depth, so we've looked at that, depending really on what the external response to that box is, and then I'm not sure how much difference we'd have as far as the safety of the plant. There is certainly complexity in the box, and if you've said for operational purposes you don't want to challenge those safety systems, it's a major difference.

MR. MOORE: Yes. I think one of the issues there is how that box interacts with you. How do you know that that box is functioning properly? So, you know, you can get into issues of silent players with a system that alters the internal functionality. You know, that can be a problem.

QUESTION: DR. WILLIAM EVERETT (AT&T Bell Labs): I'm a software engineer, but a lot of what I'm hearing is that maybe this workshop has been structured a little bit in the wrong way. What you've heard in these two days was about software engineering technology, and now those of you who are out there running the plants have the challenge of molding your problems around a solution. Maybe next year what we should have is the software engineers in the audience and those who are in the job of running these plants up explaining and characterizing what your problems are, what are the kinds of failures you have, and so forth, so that we can maybe mold some solutions around your problems.

I've heard some other things too, and I think there is a language barrier here. There is some communication problems. I hear those who are in the plants talking about failures, reliability and so forth, and I hear the software engineers talking about errors and faults, and I think maybe the software engineers who are going to work in the nuclear power area should be focusing their language to the users, and that is "failures." In running the plants you're concerned when do failures occur, how often do they occur, what's the propensity of occurring, and what are the consequences of the failure. That's got to tie into the equation, if you want to do anything about it you've got to measure it. That came out of Kyle's talk. You have costs and you have schedules. If you can't measure reliability then you can't do anything about it. You've got to tie those measures back to a lot of these technologies you've been talking about. How will an object-oriented technology quantitatively affect reliability of the product?

MR. BHATT: What I wanted to say is the issue in the nuclear industry is how we make our decisions on reliability and cost. We have a terminology which is something called SCRAM, which means something happened which means we are going to temporarily shut down because

the safety system gave us the picture and shut down. SCRAM is already a key indicator of importance of safety parameters and the operator and the utility people also get lost time and money. Feedwater controller was the first one which we tried to adapt because they were the first indicator on the SCRAM basis. Second within the line is the reactor protection system, and that's the reason we are talking about the protection system.

QUESTION: JERRY VOSS (Tenera Operating Company): I have a couple of issues. I hear a lot of people saying these are simple systems. These are simple systems, and they're not anything we are going to work with. I kind of thought back to the time, for the instrumentation people here, when we went from pneumatics into electronics, or to the electrical systems, and everybody said, "Yes, no big deal. I take the air hose out, put a wire in here, and nothing happens." It seemed like they were faster. It's not just the simple system. There are other things that are involved in here, and we, as instrumentation people, have to understand that and what that box being put in there means to us and what digital systems means to us. We can't sit here and say it's easy, anybody can do it, and we don't have to do anything extra for it. When we look at errors and what errors are acceptable to us, we as instrumentation people in the design of these activities have to look at what are the functions we want done, what are the systems in there, and we specify that function and we specify whether an error is acceptable or not, and how we would accommodate that error or if there has to be error trapping, or whatever we have to do in there. So, it's really based on going back to the systems people and the experts in this area to define whether errors are acceptable and to present those definitions right up front. "Yes, I can take an error in this area," or, "I can't take an error in this area," and give us good specifications at the beginning and then we verify everything based on that specification.

QUESTION: WILLIAM GHRIST (Westinghouse Electric): I was trying to sit back there and control myself, but some of the comments and questions that came up were sort of addressed to me without people knowing it. I was intimately involved with the development of the digital protection system that's being used on Sizewell, and so I think I can maybe clarify things a little bit. I'd also like to comment on a couple of statements that I think could be a little bit misleading.

A digital-based protection system is not a real simple system. Just because the protection function, from a high-level standpoint, is relatively simple and straightforward doesn't necessarily mean that the way you implement it ends up being completely simple. There are several reasons for that. If you think about what's in an analog system down to the level of individual transistors, resistors, and capacitors, maybe it's not quite as simple as you think it is. When you think down to the level of the software code that's going into the digital system you're really thinking down to that same level.

Another aspect is a lot of the complexity that goes in there is put in there for a specific reason to accomplish specific things. I think it is dangerous to say that more complex is necessarily less reliable, because most of the complexity that goes into these systems is put in there specifically to address the reliability and the integrity of the software design.

When we started on this, years back, some of the principles we tried to use in developing our software we later learned people were calling, at least to some extent, an object-oriented design approach. I didn't know it was an object-oriented design approach then, but we did things

like trying to isolate independent functions into different modules so they don't interact with each other in peculiar ways and so you can change something here which has to do with a protection function without having to worry about whether or not your analog input processing is going to work with that new protection function you're doing, because the analog input processing was isolated off into its own software module.

Doing that sort of thing adds complexity, but what it does, and one of our key goals, was to take the complexity that was necessary for those sort of aspects, and for things like putting in self-diagnosis, which improves the reliability and improves the fault tolerance and, in particular, improves the failsafe nature of the system, taking that complexity and isolating it into functions which could be done once and put a lot of effort into making sure they're done the right way, verified, and not have to keep repeating that if you want to change something at the functional level.

So, our goal was to try to recognize there was at least two different parts to the software in these systems. There's the part that runs the system itself, and there is the part that does the protection function. We tried to design the software such that doing the part that does the protection function is very straightforward and very simple, and by putting all the complexity into the other part so that we could concentrate our efforts on doing that.

Now I'd like to make a couple of other clarifying comments. Number one, I think in talking about errors we have to recognize that we can't be too simple about it. Having a fault, or a flaw, shall we say, in the software does not necessarily have the potential to lead into a failure. Numbers of errors per lines of code were being bandied about. I don't know necessarily what that represents, but there are many different types of errors. There is a whole spectrum of different kinds of errors that you can have in software, many of which have zero potential for introducing failure into the system. You can have a discrepancy between a specification and the code that is a discrepancy and, therefore, an error that has no functional result whatsoever. On the other hand, you may have an error that will lead to a complete loss of the ability to do the function.

When you talk about whether or not we can tolerate errors, you have to keep taking into account that there are many different kinds of errors and many of them are much more tolerable than others. The ones that keep it from performing its function are not tolerable. The ones that are sort of like bookkeeping errors, so to speak, of the development process are a whole different category.

I'd also like to mention in passing that in the system we developed, which has quite a number of microprocessors, we do not use an operating system. We did not use an interrupt system. We do use a central approach to things, and this is in conformance, by the way, with the recommendations of IEC-880.

DR. LANCE A. MILLER (SAIC): I've heard several people from the audience give the statement that there is a disconnect with too much emphasis on problems of software relative to what they see is the need. I wanted to give some counter-examples from our point of view.

In 1990, John Bernard and Anne Washio from MIT published a book on expert systems in nuclear applications and identified 300 expert systems. None of them were safety-related, but they were all providing decision support. Since then, certainly, I know of at least 100 that they've been involved in. The thing that I'm certainly concerned about is you talk about safety

systems and control systems now, and that's just the beginning. I mean, really what we're concerned about, or I'm concerned about, is what happens if the Japanese control system gets popular over here where for example, in the Mitsubishi control room they have a scenario in which a gas comes in and all the operators go to sleep for 30 minutes. Everything has to run. In that scenario you have to have every possible capability of work stations that we have and high-level language to both program the software and to run it.

What about emergency operating procedure tracking systems? You have the operators there and it automates the EOPs for you and makes them available to you. Obviously it's not safety-related, but there is going to be enormous dependency on those. You're going to need extremely sophisticated software that can keep up with the data base and to implement such a system and yet it's a very integral system.

A third area that for the nuclear area is the management of aging plants. Things are breaking down, and you have to really worry about it. There are a number of systems that can help you decide, for example acoustic systems that can monitor the high-frequency characteristics of your container vessel and through an A&S system advise you if cracks or fractures are going to occur. The same way with turbine flows and pipes and so on. These have to be integrated into your control systems.

All of these will require extremely sophisticated software and software environment tools and capabilities to develop, and I think that's the area where some of us, at least, are looking forward. It's not the back-up safety system, it's these other things that will surge once digital gets into play.

MR. GALLAGHER: Thank you.

I think in that area there has been a lot of work done in the international arena, both by the IAEA and by the IEC to recognize that there are different levels of importance to safety and that based upon the level of importance to safety one tries to do the design, V&V and other software development processes lined up with that, so that you don't go overboard and try to do everything as though it was a Class IE system. Documents like that are now in place.

QUESTION: DAVID HOLCOMB (Oak Ridge National Laboratory): One of the hindrances to the application of microprocessor-based protection systems in power plants is the probability of common-mode failure that everybody talks about, but to my mind, given the relative simplicity of the functions that they have to do, and also the high reliability they have to achieve, I think it's possible to develop a system whose probability is not any different from the probability that we feel is due to the hardware. I think also that we can glean something from computers that are used in similar applications outside of the nuclear industry. For example, can any one of the members tell me the failure rates of fault-tolerant computers due to common software as opposed to the failure rate due to the hardware itself for, let's say, triple redundant computers?

DR. KNIGHT: If you're thinking about a system where there has been operational experience with multiple versions running on the same computer, to the best of my knowledge that data has never been released. The people who make the control systems, for example, on the Airbus A-310, never got any failure data back from the operational experience with that. It's not that people are withholding it in that case. It's that information has never been captured. I can't

think of any other industrial application of design diversity in that way where there is extensive operational experience, but I'm pretty confident that the data has not been collected for the few systems that I'm aware of.

MR. GALLAGHER: One final question, especially for the advanced light water reactors, your regulations will be heavily based upon the control of the process because there really is no product to look at. The only thing we can deal with is the process by which the product will be developed. If one looks, for instance, as the SEI process maturity model it shows that you really don't get in heavily to process management until you reach a Level 4, and so this brings up the question, is there a connection between putting a lot of emphasis on how the process is managed and the Level 4 that's dealt with in the SEI model? I wonder if anybody would like to comment on that as to what this might mean?

DR. KNIGHT: Something about where others fear to tread, I will jump in. The SEI maturity model is, as I think Win suggested this morning, is a tremendous piece of work because of the visibility that it has brought to software. All of a sudden managers all over the country have become aware that their companies were writing software because they had to find out what maturity level they were at.

The thing about the maturity model though is that it really is aimed at, I think, large-scale software development, major systems, that are being deployed in all kinds of different Government, in particular, systems. I don't think that that model was ever designed, or would it really fit, into these relatively small systems that I think are being discussed here, but where we have very high dependability requirements. Perhaps what the NRC might consider is learning from the SEI experience and developing their own maturity model. Applying it as it stands, I don't think it's really intended for that and I don't think it would serve the purpose, if I understood your question correctly.

MR. GALLAGHER: Anybody else?

DR. McHUGH: I agree.

MR. PAULITZ: What is being done to separate information that maintenance should have and that may be different from what the operator needs regarding both systems and component failures? I asked the question and it was supposedly answered before. All I was trying to point out is in present designs, everything but the kitchen sink was brought to the operator's attention. Somebody should know about and fix failures in the systems, but the operator should only be concerned when it's really challenging his plant. For example, is the air pressure getting low enough that it's going to cause a problem? The pilot and co-pilot, the operator and his assistant, should not be cluttering up their minds with other things that shouldn't have to pay attention to. It's true, there are a lot of things that need to be brought in, but not shared, necessarily, but discretely sent somewhere else and somebody else should pay attention to it.

What happens in the design is that somebody put a symbol on a drawing--it was a little triangle with an "A" in it and it was the enunciator, and we put another one in with a C and that became the computer. So, as the tape went on, they end up with about 2,000 of them

somewhere, which is often confusing, and I want to relate that when a plant trips it'll probably be like a Greek church at a sunrise service on Easter, but something should be done, even now probably, but certainly for the future with this.

DR. NASER: If I can just make a quick comment on that? I think your point is extremely well taken. I think that there has been a tremendous amount of effort going into control room design reviews and things like this at the utilities to try to take a look at the alarms that are in there, to restructure them and to try to find out which ones are really meaningful and which ones aren't. In many cases you've seen a number of alarms removed. We've seen them improved in various ways. But in addition to that there is work that's been going on to try to look at can you have some sort of an alarm diagnostic and processing system which tries to take a look at the context of the plant to determine what is really important information for the operator in that context. So, therefore I think your point is well taken and I think there are efforts that are going on, and I think that hopefully when we do the design for the advanced plants as well that we when we do the design in the first place we'll have done it right in what's really meaningful.

8 PREPARED STATEMENTS

At the close of the meeting, an opportunity was given for short position papers to be presented. Mr. Robert Mullens presented a paper written by Mr. Wayne Glidden (Duquesne Light Company) on behalf of the Nuclear Utilities Software Management Group (NUSMG).

8.1 NUSMG Presentation: Mr. Wayne Glidden (presented by Mr. Robert Mullens)

Nuclear Utilities Software Management Group

Wayne Glidden
Duquesne Light Company

I am Wayne Glidden of the Duquesne Light Company, Beaver Valley Plant. I am speaking today as a representative of the Steering Committee of the Nuclear Utilities Software Management Group (NUSMG).

NUSMG was organized in 1989 by several utilities with the objective of providing a formal organization which would:

1. Acquire and exchange information on the regulation, processes, and programs used to establish and maintain control over software configuration and quality.
2. Assist the membership in the implementation of quality assurance programs and procedures which reflect these requirements.

Membership is restricted to electric utilities who have or are developing software management programs. We currently have forty (40) members.

Our utility member representatives are required to be knowledgeable in and have practical experience with SQA. Presently, our official representatives are fairly equally divided in experience and responsibilities between quality assurance, engineering, operations, and information systems.

We meet semi-annually to share information, review and discuss recent Software Quality Assurance (SQA) developments, provide expert speakers on a variety of subjects, and provide a forum for the betterment of member's SQA knowledge, development, and implementation. Our next meeting is October 20-22, 1993.

NUSMG has established several goals to accomplish its mission;

1. Obtain active participation from representatives of each nuclear utility. As I mentioned earlier, we presently have 40 active member utilities representing over 90 nuclear plants.
2. Provide services needed by our members to assist them in achieving compliance with applicable requirements and SQA commitments.
3. Influence the development of practical and effective SQA in the industry, both from a regulatory and member standpoint.
4. Establish committees, as needed, to address critical SQA issues and provide consistent practical guidance on those issues to our members.

One of our first major actions to accomplish these goals was the completion of a self-assessment survey of our membership. The survey resulted in the generation of a report on SQA to the NRC through NUMARC in June, 1992. This report led to a subsequent meeting with the NRC to discuss the report and answer questions.

We have members involved with industry workgroups such as ASME/NQA, EPRI, ANSI, and IEEE. Information from this involvement is relayed to the rest of the membership via NUSMG bulletins, news letters, and the semi-annual meetings.

We have an active liaison with NUMARC, and through them, are developing relationships with regulatory bodies. We use these relationships to identify emerging SQA issues which need to be promptly addressed by utilities. NUSMG provides a focal point for developing industry consensus in responding to these issues.

We presently have three active committees working on issues;

1. Software commercial grade dedication.
2. Software procurement (safety related and important safety software).
3. Utility SQA program self-evaluation guidelines.

All three committees are scheduled to present final drafts of guidance documents to our membership at the next semi-annual meeting in October. We expect that these products will be employed by our membership and provide additional guidance for SQA development and improvement efforts.

Each of the NUSMG Committee work products are related to the issues discussed at this NIST Workshop and NUSMG would be pleased to discuss our efforts at a future meeting of the ACRS Subcommittee on Computers in Nuclear Power or a NIST workshop.

Through our activities and relationships, NUSMG strives to be the spokesperson for the nuclear industry in matters related to SQA, which include not only software quality assurance, but also related issues such as the use of class 1E digital systems including upgrading analog systems, commercial dedication, and the use of software in non-safety related applications (those important to the efficient and cost effective operation) at a nuclear utility.

Thank you for allowing me to introduce you to NUSMG. The utility community is extremely interested in the efforts of the NRC to develop effective regulatory policies and review plans for software driven systems. NUSMG is available to provide input to your efforts to assure that practical regulatory programs are developed.

If there are any questions, please call NUSMG at 215-582-5945.

9 NRC CLOSING REMARKS

This section contains the edited transcript of the NRC closing remarks made by Mr. Franklin Coffman (Chief, Human Factors Branch, Office of Nuclear Regulatory Research) and Dr. Cecil Thomas (Deputy Director, Division of Reactor Controls and Human Factors, Office of Nuclear Reactor Regulation). This editing consisted of minimal editing to correct grammar and remove extraneous references to microphone volume, etc.

9.1 Closing Remarks: Mr. Franklin Coffman

The workshop has provided an opportunity for the NRC both to receive and to share information concerning the orientation and focus of their programs addressing the introduction of advanced digital systems in nuclear power plant designs and into currently operating plants.

Over the last two days, we all have listened to and participated in discussions on a myriad of issues on software reliability and nuclear safety. Although each individual has naturally considered the issues and occasionally reacted based upon individual perspectives with either (1) "This makes sense", (2) "I don't agree with that", or (3) "It wasn't even discussed." There has been a deliberate attempt to achieve the objectives of the workshop by neither tallying nor deciding the reactions. Rather, priority was given to an intense exchange of important information. Certainly, it is not my place to tally reactions nor summarize findings on any of the issues discussed. Nor is it my place to identify any new issues or proposed positions.

However, it is my place to simply remind us of the issues for consideration that we have heard discussed. The list of issues that I am about to share with you is just a list and as such is tentative and initial. To my recollection the following twenty issues were included in the discussions:

1. The means to obtain a complete and precise translation of a using organization's needs into design specifications. This included the issues surrounding the role of formal methods for specification capture and analysis.
2. The question of allocating the requirements between the hardware and the software while defining the interface requirements between the digital system, the driving software, the human operators and maintainers, the plant systems, and the power conversion phenomena. Yet it is the total system that is to be evaluated including the consequences of software failure on the total-system's performance.
3. The issues surrounding the role of hazard analysis for defining the level of detail at which fault-tolerance is required.
4. Questions on doing common-mode-error analysis and questions on the technical basis for criteria to invoke diversity as a defensive measure. The questions include considerations of the net-benefit of diversity.

5. The question of adequate reliability metrics for important systems' properties like complexity, and the relationships of the metrics to the degree of safety obtained.
6. The potential for a response-time hazard in digital systems because they are incremental (in contrast with the continuous nature of analog-hardwired systems).
7. The issue of the role of specification-based and statistical testing requirements and acceptance criteria.
8. The acceptance or certification of Commercial Off-The-Shelf software for safety-critical applications.
9. The issues associated with the transition from analog to digital including 10CFR50.59 reviews and Unreviewed Safety Questions.
10. The issues associated with the net benefits to a system's reliability from developing software using structured processes and structured languages, and improved languages.
11. The questions associated with the use of CASE tools for design specifications, testing design, and safety reviews.
12. The role of V&V and the degree to which different techniques assure reliable software, and the degree to which the V&V must be independent of the designer.
13. The issue of standards or conventions for controlling software configurations.
14. Questions concerning the need to qualify or certify compilers and operating systems.
15. Questions of adequate isolation of non-safety related software from safety-related software.
16. Questions on the degree of domain knowledge needed by developers of software for nuclear applications.
17. Generally, the need for a comprehensive framework/outline for the scope and content of the technical basis and acceptance criteria for digital I&C systems.
18. The question of the need for further research and subsequent workshops on topics such as hardware and human factors.
19. Interest in the possibility of the NRC initiating a process where error experience is collected, analyzed, characterized, and distributed.

20. The impact of the trend toward the use of blocks of experienced code versus the conventional development of code.

Thank you statement

I want to openly thank Leo Beltracchi and Dolores Wallace for assembling and orchestrating a stimulating and meaningful workshop. I want to thank the RES and NIST staff that provided support for the conduct of the workshop.

We all thank the speakers for sharing their experience, insight, and (sometimes) specific solutions concerning the many perplexing issues associated with software reliability and nuclear safety. This has been a workshop rich in information.

We thank the session chairs for their attention to balancing the need for technical rigor while complying with the logistical constraints.

I thank you all for participating and especially those who asked questions and made comments that kept reality and candor before us.

Now Dr. Cecil Thomas, Deputy Director, Division of Reactor Controls & Human Factors, has some final remarks.

9.2 Closing Remarks: Dr. Cecil Thomas

This afternoon's dialogue reminded me of an observation that was made during this summer's Regulatory Information Conference. As Frank pointed out, I'm from the Office of Nuclear Reactor Regulation, and we're the ones that really take the Commissions's regulations and rules and guidance and apply them in a review of licensee's proposed changes to systems and applications and so on. But at this summer's Regulatory Information Conference, which our office puts on every year, it was observed that nuclear plant control rooms of the future would be radically different than those we see today. *[For example, they would be run by only two occupants, one person and one dog, both licensed of course. Some of you may think a person and a dog would be needed for diversity. That's not true. It's really defense-in-depth. If something happens that suggested to the operator he should do something, the dog's purpose is to bite the person, and the person's purpose is to feed the dog.]*

On a more serious note, the inevitable integration of digital systems with all of their advantages and all of their disadvantages, in both present and future nuclear power plant designs, presents the NRC with many unique challenges in carrying out its mission of protecting the health and safety of the public, especially when dealing with such rapidly evolving technologies as we've been talking about here. It's imperative that the NRC keep abreast of the state-of-the-art and obtain feedback on its research and regulatory programs from those most knowledgeable. These were the objectives of this workshop.

The outstanding presentations, the lively exchange of questions, answers and comments, and the innumerable one-on-ones in the hallways during the breaks, in the mornings and in the evenings have certainly fulfilled these objectives. So, I would personally like to thank everyone for their participation in this workshop, no matter what their role, and from our point of view the objectives have certainly been accomplished. Thank you very much.

10 SUMMARY AND CONCLUSIONS

Many speakers reinforced the need for a technical basis for regulatory requirements. Speakers in the technical sessions agreed on the need for defining software requirements in the context of system requirements and for the traceability of those requirements across the entire software lifecycle. While there was some debate among the speakers and audience about the degree of complexity inherent in safety systems, the speakers agreed that systems engineers need to clearly state the requirements, constraints, and assumptions for the safety system. One difficulty in communicating the requirements is that the terminology of the nuclear systems and software engineering communities are different. These differences can lead to miscommunications about the requirements which may have a safety impact. For example, in system development, the design phase includes the development of the software requirements and the software design. In the nuclear industry, design specifications could be system design specifications, the software requirements specification, and functions allocated to the operator. This problem is evident in the study of standards and guidelines conducted by NIST for the NRC [NUREG5930]. Another problem identified in that study and discussed by speakers in this workshop is the requirement for CM because most of the guidance documents did not invoke CM during the entire life cycle, nor was the impact on SCM clear.

Several speakers emphasized the importance of precise specifications which are traceable and maintainable throughout the development process. Dr. Knight introduced the subject by stressing the importance of systems and application engineers completely specifying the software requirements, and that software engineers are not qualified to make decisions about the system. Dr. McHugh, Mr. Poston, and Mr. Berlack all discussed the means for systems and software engineers to specify non-ambiguously the requirements and maintain the traceability of those requirements through the software development process. Dr. McHugh endorsed formal methods as a precise means of specifying requirements. Mr. Poston agreed that formal methods were a good choice, but advocated the use of tools to make the process easier and permit traceability between the tests and the requirements. Mr. Berlack stressed CM as a mechanism for precise continuing communication between the systems and software engineers. The CM method can allow the systems engineer to see how the software engineers allocated the requirements. Dr. Hanes also stressed the importance of maintaining the requirements allocation. Dr. Cuthill discussed maintaining traceability of the requirements through the design and coding phases. Mr. Fujii emphasized that software V&V had to refer back to the system specifications and involve system engineers. Finally, Ms. Scheper discussed the need for maintaining the system context for software components kept in a reuse library so that they can be included in future systems appropriately. Her approach for certifying software components and making them available for reuse in other systems could reduce the aggregate cost of software certification.

Another theme of all the technical sessions was the need for better acceptance and V&V testing. Mr. Fujii emphasized that software V&V can make a significant contribution to analyzing the allocation of functions, early in the system development. Mr. Poston, Ms. Lapassat and Dr. Everett stressed the need to automate testing. Measuring the reliability of software is a non-trivial problem but a measure of reliability is important for assuring safety. Mr. Rone,

Dr. Everett and Dr. Royce proposed partial solutions to the problem including incorporating specification-based testing and statistical testing. Dr. Royce also cited the value of a structured process and structured process language for effecting reliability.

Other major issues discussed by the speakers concerned the need for (system and software) hazard analysis requirements, fault tolerant requirements, and common mode error and diversity requirements.

Tables 1, 2, and 3 present a summary of the issues addressed from another perspective. This summary divides the topics discussed into the problems that the speakers identified, the theoretical solutions they proposed and any experience based solutions offered. These topics are further grouped by life cycle process. Table 1 contains topics raised in the software requirements and design processes. Table 2 contains topics for the processes of implementation and integration and installation, operations, and maintenance. Table 3 includes topics in the V&V and quality assurance areas.

While the speakers generally agreed on the importance of these issues, there was, at times, a discrepancy between what the speakers believed to be the issues facing the nuclear industry and what the audience (nuclear industry and regulatory personnel) believed to be the important issues.

Table 1. Requirements and Design Lifecycle Phases

	Requirements	Design
Problems Identified	Dr. Knight <ul style="list-style-type: none"> ■ Incomplete specifications ■ Formal methods not enough ■ Software engineers not qualified to deal with systems issues 	Dr. Royce <ul style="list-style-type: none"> ■ Few tools ■ System too complex to understand Mr. Rone <ul style="list-style-type: none"> ■ System too complex
Theoretical Solutions	Dr. McHugh <ul style="list-style-type: none"> ■ Use formal, precise specifications Mr. Poston <ul style="list-style-type: none"> ■ Use tools to generate specification & test cases 	Dr. Cuthill <ul style="list-style-type: none"> ■ OOD is potentially useful Dr. Miller <ul style="list-style-type: none"> ■ Allocate functions to operators
Application Experience	Mr. Joannou <ul style="list-style-type: none"> ■ Formal specification (Parnas) ■ Defined process ■ Developed OASES Framework ■ Tools for consistency checking Mr. Blauw <ul style="list-style-type: none"> ■ Standard IEEE P-7-4.3.2 ■ NUMARC Digital Upgrade Guideline 	Mr. Joannou <ul style="list-style-type: none"> ■ Fail safe and self check features ■ Defined process Mr. Sudduth <ul style="list-style-type: none"> ■ Probabilistic Risk Assessment ■ Alternate fault tolerant architectures ■ Fault tree and Failure Modes and Effects Analysis (FMEA) ■ Markov Models

Table 2. Implementation/Integration and Operation/Maintenance Lifecycle Phases

	Implementation & Integration	Installation/Operation/Maintenance
Problems Identified	Dr. Royce <ul style="list-style-type: none"> ■ No error measurement ■ Languages not designed for safety 	Dr. Hanes <ul style="list-style-type: none"> ■ No guidelines for reviewing human interfaces ■ Define extent of automation
Theoretical Solutions	Dr. Cuthill <ul style="list-style-type: none"> ■ C++ is potentially useful Ms. Scheper <ul style="list-style-type: none"> ■ Levels of criticality assurance for reusable software artifacts 	Dr. Cuthill <ul style="list-style-type: none"> ■ OO & C++ simplify maintenance Mr. Russell <ul style="list-style-type: none"> ■ Self-diagnostics and testing Dr. Hanes <ul style="list-style-type: none"> ■ Intelligent displays and aids ■ Computerized procedures
Application Experience	Mr. Berlack <ul style="list-style-type: none"> ■ Software Configuration Management 	Mr. Berlack <ul style="list-style-type: none"> ■ Software Configuration Management Mr. Joannou <ul style="list-style-type: none"> ■ Defined acceptance criteria

Table 3. Verification/Validation and Quality Assurance

	Validation	Quality Assurance
Problem Identification		Dr. Royce <ul style="list-style-type: none"> ■ No people certification ■ No organization certification
Theoretical Solutions	Mr. Poston <ul style="list-style-type: none"> ■ Tool generated tests from requirements Dr. Miller <ul style="list-style-type: none"> ■ Fault specific verification ■ Cost benefit tradeoff for V&V ■ Automation of V&V methods 	Mr. Rone & Ms. Olson <ul style="list-style-type: none"> ■ Define development process ■ Develop quality plan ■ Use cost, schedule & error detection models ■ Configuration management Ms. Scheper <ul style="list-style-type: none"> ■ Levels of certification for reusable s/w
Application Experience	Mr. Joannou <ul style="list-style-type: none"> ■ Software hazard analysis (Leveson) ■ Guided inspection of software Mr. Fujii <ul style="list-style-type: none"> ■ Estimation of V&V necessary ■ Process to select V&V methods Mr. Sudduth - ■ Simulation of system for testing Dr. Everett - ■ Isolation of safety-critical components for test acceleration Ms. Lapassat <ul style="list-style-type: none"> ■ Tools for independent verification ■ Software simulation & modeling tools 	Mr. Berlack <ul style="list-style-type: none"> ■ Configuration & change management ■ Status accounting and auditing ■ Subcontractor control Mr. Fujii - ■ System safety framework Mr. Joannou <ul style="list-style-type: none"> ■ Train personnel in methodologies and retrain as necessary Mr. Sudduth <ul style="list-style-type: none"> ■ Use proven components Ms. Lapassat <ul style="list-style-type: none"> ■ Tools for independent auditing

The speakers and audience seemed to disagree on the following:

1. The degree to which software may be a safety concern.
2. Simplicity versus complexity of the software needed in digital systems.
3. The perceived cost of assuring software versus the cost affordable by the nuclear industry.

The speakers and audience seemed to agree on the following:

1. Standards and criteria should develop from a defined technical basis, which is not currently available.
2. There are plenty of benefits from using digital systems, such as self-diagnostics and decreased human error, but operators must remain in charge.
3. There are opportunities to make improvements in some known failure classes (omitted function, unintended function).
4. A definition of diversity for software is needed.
5. Future workshops should address other components of NPPs (e.g., hardware, human operators).
6. The software engineering concepts presented at the workshop were not strongly connected to the problems faced by the nuclear industry, i.e., their vendors would be more suitable to deal with these issues.

The aggregate of technical presentations and issue perspectives leads to the conclusion overall that many of the management and technical problems of digital systems are not sufficiently mature for regulation. Research to define technical solutions and research into existing solutions in other industries is necessary.

11 REFERENCES

[ASMENQA2]

ASME NQA-2a-1990, Part 2.7, "Quality Assurance Requirements for Nuclear Facility Applications," The American Society of Quality Engineers, 1990.

[IEC880]

IEC 880, "Software for Computers in the Safety Systems of Nuclear Power Plant Stations," International Electrotechnical Commission, 1986.

[IEEE603]

IEEE Std 603-1980, "Standard Criteria for Safety Systems for Nuclear Power Generating Stations," The Institute of Electrical and Electronics Engineers, Inc., 1980.

[IEEE7432]

IEEE Std. 7-4.3.2-1993, "Application Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear power Generating Stations," American Nuclear Society, 1993.

[NUREG5930]

"High Integrity Software Standards and Guidelines," Wallace, Dolores R., Laura M. Ippolito, D. Richard Kuhn, NUREG/CR 5930, U. S. Nuclear Regulatory Commission, September 1992. (Also published as National Institute of Standards and Technology NIST SP 500-204.

APPENDIX A WORKSHOP AGENDA

Final Announcement

Digital Systems Reliability and Nuclear Safety

**September 13-14, 1993
Rockville Crowne Plaza Hotel
Rockville, Maryland**

U.S. Nuclear Regulatory Commission

**U.S. Department of Commerce
Technology Administration
National Institute of Standards and Technology**

ABOUT THE WORKSHOP

Workshop Co-Chairs:

Mr. Leo Beltracchi, U.S. Nuclear Regulatory Commission
Ms. Dolores R. Wallace, National Institute of Standards and Technology

Sponsored by:

The United States Nuclear Regulatory Commission

In Cooperation with:

The National Institute of Standards and Technology

As analog hard-wired process control systems and safety systems within nuclear power plants wear out, they are being replaced with systems using digital technology. There are many unique design and safety issues for digital systems. The Nuclear Regulatory Commission is developing regulations and guidelines to address these issues. This workshop will provide state of the art information to the Nuclear Regulatory Commission staff and to the nuclear industry. The purposes of this workshop are to:

- provide feedback to the NRC from outside experts regarding potential safety issues, proposed regulatory positions, and research associated with the application of digital systems in nuclear power plants, and
- continue the in-depth exposure of the NRC staff to digital systems design issues related to nuclear safety by discussions with experts in the state of the art and practice of digital systems.

AGENDA

Monday, September 13, 1993

8:00 Registration; Coffee

OPENING SESSION

Chair: Mr. Leo Beltracchi, U.S. Nuclear Regulatory Commission

8:30 Welcome

Commissioner Kenneth C. Rogers, U.S. Nuclear Regulatory Commission

8:45 Welcome and Opening Statement

Mr. Eric S. Beckjord, Director, Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission

9:00 Welcome and ACRS Perspective

Dr. J. Ernest Wilkins, Jr., Chairman, Advisory Committee on Reactor Safeguards
U.S. Nuclear Regulatory Commission

ISSUE SESSION: PERSPECTIVE FOR NUCLEAR POWER PLANTS

Chair: Mr. Joel Kramer, U.S. NRC

9:15 Presentation on NRC Regulatory Positions and Guidelines

Mr. William T. Russell, Associate Director for Inspection and Technical
Assessment, Office of Nuclear Reactor Regulation
U.S. Nuclear Regulatory Commission

9:45 NRC Research Activities

Mr. Leo Beltracchi, Senior Project Manager
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission

10:15 Industry Perspective

Mr. Richard Blauw, Commonwealth Edison Company

10:45 Break

11:00 Experiences from Application of Digital Systems in a NPP

Mr. Paul Joannou, Ontario Hydro

Monday, September 13, 1993 (cont.)

TECHNICAL SESSION: DIGITAL SAFETY SYSTEMS FOR NUCLEAR POWER PLANTS

Chair: Mr. Joseph Joyce, U.S. NRC

11:30 Hardware Aspects for Safety-Critical Systems
Mr. A.L. Sudduth
Duke Power Company

11:50 Software Aspects for Safety-Critical Systems
Dr. John Cherniavsky
National Science Foundation

12:10 Human Aspects for Safety-Critical Systems
Dr. Lewis F. Hanes
Nuclear Industry Independent Consultant

12:30 Questions and Discussion

1:00 Lunch

TECHNICAL SESSION: SOFTWARE ENGINEERING FOR HIGH INTEGRITY SYSTEMS

Chair: Ms. Dolores R. Wallace, NIST

2:30 Interaction Between Software and System Engineering for Safety-Critical Applications
Dr. John Knight, University of Virginia

2:55 Formal Methods for Requirements, Specifications
Dr. John McHugh, Portland State University

3:20 Software Test Cases Derived from Formal Requirements
Mr. Robert M. Poston, Interactive Development Environments

3:45 Break

4:00 Object Oriented Design for Safety-Critical Systems
Dr. Barbara B. Cuthill, National Institute of Standards and Technology

4:25 Questions and Discussion

Tuesday, September 14, 1993

8:00 Coffee

TECHNICAL SESSION: METHODS FOR REDUCING RISKS IN SOFTWARE SYSTEMS

Chair: Mr. Roger U. Fujii, Logicon, Inc.

8:30 Automated Tools for Safety-Critical Software
Ms. Anne-Marie Lapassat, Commissariat a L'Energie Atomique

8:55 Risks of Safety-Critical Software
Dr. Winston Royce, TRW, Incorporated

9:20 Software Metrics for Safety-Critical Applications
Mr. Kyle Y. Rone, IBM Houston, Texas

9:45 Software Reliability for Safety-Critical Applications
Dr. William Everett, AT&T Bell Laboratories

10:10 Questions and Discussion

10:30 Break

10:50 Software Configuration Management for Safety-Critical Applications
Mr. H. Ronald Berlack, Configuration Management International

11:15 How Much Software Verification and Validation is
Adequate for Nuclear Safety?
Mr. Roger U. Fujii, Logicon, Incorporated

11:40 Fault-Specific Verification (FSV)--An Alternative VV&T Strategy for High
Reliability Nuclear Software Systems
Dr. Lance A. Miller, Science Applications International Corporation

12:05 Certification of Software for Reuse into Safety-Critical
Applications
Ms. Charlotte O. Scheper, Consultant

12:30 Questions and Discussion

1:00 Lunch

Tuesday, September 14, 1993 (cont.)

- 2:30 PANEL: Application of Workshop to NRC activities
Moderators: Mr. Franklin Coffman - Office of Nuclear Regulatory Research
Mr. John Gallagher - Office of Nuclear Reactor Regulation

Panel Members

Dr. John Knight, University of Virginia
Dr. John McHugh, Portland State University
Dr. Joseph Naser, Electric Power Research Institute
Dr. Winston Royce, TRW Incorporated

Panel Issues:

- Are the proper issues being addressed?
- What other issues need to be addressed?
- Are proposed NRC regulatory positions complete and correct?
- What are the considerations for further research?

- 4:30 Questions and Discussion

- 5:00 Prepared Statements

- 5:30 NRC Closing Remarks

Mr. Franklin Coffman
Chief, Human Factors Branch, Office of Nuclear Regulatory Research

Dr. Cecil Thomas, Deputy Director, Division of Reactor Controls & Human Factors
Office of Nuclear Reactor Regulation

SUPPLEMENTARY INFORMATION:

The following documents relevant to the workshop are now in the NRC Public Document Room, located at 2120 L St., N.W. (lower Level), Washington, DC

1. Draft, "Operating Reactors Digital Retrofits, Digital Systems Review Procedures," Ver.1
2. Draft, "Branch Technical Position (HICB), Digital Instrumentation and Control Systems in Advanced Plants."

Two additional documents relevant to the workshop which participants may wish to review are:

1. International Electrotechnical Commission Standard 880, "Software for Computers in the Safety Systems of Nuclear Power Plants," 1986.
2. P-7-4.3.2, Draft 8, American National Standard, "Standard Criteria for Digital Computers in Safety Systems of Nuclear Power Generating Stations."

APPENDIX B AUTHOR INDEX

	<u>Page(s)</u>
Beckjord, Eric S.	v, vi, 1, 3, 9, 282, 288, 324
Beltracchi, Leo	v, vi, vii, viii, ix, 1, 17, 18, 31, 311, 323, 324
Berlack, H. Ronald	xii, xiii, xiv, xx, xxii, xxiii, 193-195, 229, 283, 315, 317, 318, 326
Blauw, Richard J.	vi, vii, ix, xxii, 2, 17, 18, 47, 317, 324
Cherniavsky, John C.	x, 79, 107, 108, 325
Cuthill, Barbara B.	xi, xii, xx, xxii, 131, 132, 163, 191, 315, 317, 325
Everett, William	xiii, xx, xxi, xxiii, 194, 225, 227, 228, 299, 315, 316, 318, 326
Fujii, Roger U.	xiii, xiv, xx, xxiii, 35, 42, 194, 195, 247, 254, 255, 315, 318, 326
Glidden, Wayne	291, 305
Hanes, Lewis F.	x, xx, xxii, 79, 109, 130, 315, 317, 325
Joannou, Paul K.	vi, vii, ix, xxii, xxiii, 2, 17, 19, 61, 78, 317, 318, 324
Knight, John	xi, xii, xiv, xv, xx, xxii, 107, 131-133, 136-138, 216, 255, 275, 277, 284, 286, 290, 293-295, 298, 302, 303, 315, 317, 325, 327
Lapassat, Anne-Marie	xii, xx, xxiii, 193, 197, 211, 315, 318, 326
McHugh, John	xi, xii, xv, xvi, xx, xxii, 131, 132, 139, 145-147, 191, 216, 275, 278, 283, 287, 292-297, 303, 315, 317, 325, 327
Miller, Lance A.	xii, xiii, xiv, xxii, xxiii, 78, 136, 193-195, 227, 257, 283, 301, 317, 318, 326
Musa, John	225
Naser, Joseph	xv, xvi, 138, 275, 280, 284, 285, 290, 293, 294, 296, 298, 299, 304, 327
Olson, Kitty M.	xxiii, 219, 318
Poston, Robert M.	xi, xii, xx, xxii, xxiii, 131, 132, 145, 149, 160, 161, 315, 317, 318, 325
Rogers, Kenneth C.	v, vi, 1, 3, 5, 282, 324
Rone, Kyle Y.	xiii, xx, xxii, xxiii, 194, 219, 224, 286, 315, 317, 318, 326
Royce, Winston	xii, xiii, xiv, xv, xvi, xxi, xxii, xxiii, 147, 193-195, 213, 216, 217, 254, 255, 275, 279, 284, 316, 317, 318, 326, 327
Russell, William T.	v, vii, xxii, 1, 17, 21, 286, 288, 317, 324
Scheper, Charlotte O.	xiii, xiv, xx, xxii, xxiii, 193, 195, 267, 274, 315, 317, 318, 326
Sudduth, A.L.	x, xxii, xxiii, 79, 81, 105, 106, 160, 289, 295, 317, 318, 325
Wilkins, J. Ernest, Jr.	vii, 1, 3, 13, 324

*Final Participants List***Digital Systems Reliability and Nuclear Safety Workshop****September 13-14, 1993****National Institute of Standards and Technology****Gaithersburg, Maryland**

Anil Agarwal
GPU Nuclear Corp.
1 Upper Pond Rd.
F1A
Parsippany, NJ 07054
USA

Iqbal Ahmed
U.S. NRC
11555 Rockville Pk.
OWFN-8H3
Rockville, MD 20852
USA

Paul Amrozowicz
Division of Naval Reactor
2521 Jeff. Davis Hwy.
NAVSEA Code 08
Washington, DC 20585
USA

Namsung Arne
Korea Electric Power Corp.
46 Wiley Rd.
Belmont, MA 02178
USA

S.V. Athavale
U.S. NRC, OWFN
11555 Rockville Pike
MS BH3
Rockville, MD 20852
USA

Michael Bangham
DHR Technologies, Inc.
10400 Little Patuxent
Ste. 310
Columbia, MD 21044
USA

Mario Barbacci
Carnegie Mellon Univ.
4500 Fifth Ave.
Rm. 2410
Pittsburgh, PA 15213-3890
USA

Thomas Barger
Sandia Nat'l. Labs.
P.O. Box 5800
Dept. 5931
Albuquerque, NM 87110
USA

Mel Barnes
AEA Technology Consulting
Risley, Warrington
TH4L7
Cheshire, WA63A
UK

Allen Barth
Baltimore Gas and Electric
1650 Calvert Cliffs
1st Floor NOF
Lusby, MD 20657
USA

Barney Bear
Siemens Power Corp.
155 108th Ave, NE
8th Fl.
Bellevue, WA 98004
USA

Eric Beckjord
U.S. NRC

Leo Beltracchi
U.S. NRC

Ron Berlack
Config. Manag. International
16 Main St.
P.O. Box 118
Amherst, NJ 03031-0118
USA

John Bernard
MIT Nuclear Reactor Lab.
138 Albany St.
NW12-208
Cambridge, MA 02139
USA

Jess Betlack
MPR Associates, Inc.
320 King St.
Alexandria, VA 22314-3238
USA

Siddharth Bhatt
EPRI
3412 Hillview Ave.
Palo Alto, CA 94303
USA

Debbie Blackstone
NIST
Bldg. 223, Rm. B266
Gaithersburg, MD 20899-0001
USA

Richard Blauw
Commonwealth Edison Co.
125 South Clark St.
Rm. 422
Chicago, IL 60603
USA

Stephen Blazo
Bechtel Corp.
9801 Washington Blvd.
1C2(R)
Gaithersburg, MD 20878
USA

Paul Bonnett
U.S. NRC

E. Thomas Boulette
Boston Edison Company
Rocky Hill Rd.
Pilgrim Nuclear Pwr.
Plymouth, MA 02360
USA

Edward Bradley
TVA
1101 Market St.
Chattanooga, TN 37402
USA

Kevin Bradley
PP&L
P.O. Box 467
Berwick, PA 18603
USA

Robert Brill
U.S. NRC
Nicholson Ln.
Washington, DC 20555
USA

John Calvert
U.S. NRC
475 Allendale Rd.
King of Prussia, PA 19406
USA

Jose I. Calvo
CSN
Justo Dorado, 11
Madrid, 28035
SPAIN

Jim Campbell
Entergy Operations, Inc.
P.O. Box 756
M&E Bldg.
Port Gibson, MS 39150
USA

John C. Catlin
Bechtel Corp.
9801 Washington Blvd.
7A-5(W)
Gaithersburg, MD 20878
USA

James Chelini
Raytheon Company
528 Boston Post Rd.
236
Sudbury, MA 01776
USA

Michael Cheok
NUS
910 Clopper Rd.
Gaithersburg, MD 20878-1399
USA

John Cherniavsky
NSF

Matthew Chiramal
U.S. NRC
11555 Rockville Pike
WFN 8H3
Rockville, MD 20851
USA

Greg Chisholm
Argonne National Lab.
9700 S. Cass Ave
MS 10
Argonne, IL 60439
USA

David Cleaves
The Mitre Corporation
7525 Colshire Dr.
MS NASA
McLean, VA 22102-3481
USA

Douglas Coe
U.S. NRC/ACRS
Rm. P-315
Washington, DC 20555
USA

Franklin Coffman
U.S. NRC

Robert Copyak
Arizona Public Service
411 N. Central
MS 1855
Phoenix, AZ 85004
USA

Dan Craigen
ORA Canada
265 Carling Ave.
Ste. 506
Ottawa
CANADA

Dan Crandall
TRW Systems
4243 Predras Dr. E.
Ste. 100
San Antonio, TX 78228
USA

Stephen D. Crocker
Trusted Information System
3060 Washington Blvd.
Glenwood, MD 21738
USA

Barbara Cuthill
NIST
Bldg. 225, Rm. B266
Gaithersburg, MD 20899-0001
USA

Ken Cutler
Phil. Electric Company
Peach Bottom Atomic
SMB 3-2
Delta, PA 17314
USA

Gustav Dahl
OECD Halden Reactor Pro.
OS Alle 13
N-1751
Halden
NORWAY

E. Terrence Dailey
Carnegie Mellon Univ.
4500 Fifth Ave.
Rm. 2410
Pittsburgh, PA 15213-3890
USA

Thomas G. DeVille
Bechtel Corp
50 Beale St.
San Francisco, CA 94925
USA

Ray DiSandro
Philadelphia Electric Co.
965 Chesterbrook Blvd.
63 A-1
Wayne, PA 19087-5691
USA

Stanley Dlugolenski
ABB Combustion Engineering
1000 Prospect Hill
9420-4BB
Windsor, CT 06095
USA

James Dukelow
Battelle Pacific NW Lab.
P.O. Box 999
MS K8-37
Richland, WA 99352
USA

Mike Engineer
Fluor Daniel/M&O
230 S. Tryon St.
DE&S, Ste. 300
Charlotte, NC 28021
USA

Larry E. Erin
Westinghouse Electric Corp.
902 Garden City Dr.
Monroeville, PA 15146
USA

Paul Eshleman
Engr. & Science Assoc.
6110 Executive Blvd.
Ste. 315
Rockville, MD 20852
USA

William Everett
AT&T Bell Labs.
P.O. Box 3030
Rm. 2L-503
Holmdel, NJ 07733-3030
USA

Yang-Ching Fan
Taiwan Power Co.
242 Roosevelt Rd.
Sec.3/20F/I&C Div.
Taipei
TAIWAN

Arthur Faya
AECB
280 Slater St.
Ottawa, K1P 559
CANADA

T. Edward Fenstermacher
PLG, Inc.
1615 M. St. NW
Ste. 730
Washington, DC 20036
USA

Stanley Focht
American Nuclear Insurers
29 South Main St.
Town Ctr., Ste. 3005
West Hartford, CT 06107
USA

Ken France
IBM
9321 Corporate Blvd.
Rockville, MD 20850
USA

Roger Fujii
Logicon, Inc.
222 W. 6th St.
San Pedro, CA 90731
USA

John Gallagher
U.S. NRC

John Ganiere
U.S. NRC
OWFN
MS 8-H-3
Washington, DC 20555
USA

Ronald Gardner
U.S. NRC
799 Roosevelt Rd.
DRS, Region III
Glen Ellyn, IL 60137
USA

Frank Gee
U.S. NRC
1450 Maria Ln.
Walnut Creek, CA
USA

William D. Ghrist
Westinghouse Electric Corp.
200 Beta Dr.
Pittsburgh, PA 15238
USA

Wayne Glidden
Nuclear Utilities Software
P.O. Box 415
Birdsboro, PA 19508
USA

Ed Goss
Union Electric
P.O. Box 620
MC 620
Fulton, MO 65251
USA

Ken Graff
Philadelphia Electric Co.
955 Chesterbrook Bldg.
52A-5
Wayne, PA 19087-5691
USA

Jim Graham
NIST
Bldg. 225, Rm. B226
Gaithersburg, MD 20899-0001
USA

Kevin Graney
Bechtel Corp.
9801 Washington Blvd.
Gaithersburg, MD 20879
USA

John Green
Detroit Edison
6400 N. Dixie Hwy.
220 TAC - Fermi 2
Newport, MI 48166
USA

Lewis Hanes
Nuclear Power Industry Consultant
2023 Wickford Rd.
Columbus, OH 43221
USA

Stephen Hamrock
Westinghouse Electric Corp.
200 Beta Dr.
Pittsburgh, PA 15238
USA

Jeff Hansen
EG&G Inc.
P.O. Box 1625
MS 3860
Idaho Falls, ID 83442
USA

Percy Haralson
Southern Calif. Edison Co.
P.O. Box 128
San Clement, CA
USA

Steve Harper
SAIC
1213 Jefferson Davis Hgwy.
Ste. 1300
Arlington, VA 22202
USA

John Harrison
Tenera Operating Company
7272 Wisconsin Ave.
Ste. 300
Bethesda, MD 20814
USA

Mahbubul Hassan
Brookhaven National Lab.
Bldg. 130
32 Lewis Road
Upton, NY 11973
USA

Jane Hayes
SAIC
1213 Jefferson Davis Hgwy.
Ste. 1300
Arlington, VA 22202
USA

Herb Hecht
SoHar, Inc.
8421 Wilshire Blvd.
Ste. 201
Beverly Hills, CA 90211-3204
USA

John Hefler
Pacific Gas & Electric Co.
333 Market St.
A10E, Rm. 1171
San Francisco, CA 94177
USA

Connie Heitmeyer
NRL
Code 5540
Washington, DC 20375
USA

Ali Hekmati
GE
175 Curtner Ave.
MC 765
San Jose, CA 95125
USA

Michael Hellums
Tennessee Valley Authority
1101 Market St.
MS LP5B
Chattanooga, TN 37402
USA

Nolan T. Henrich
Tennessee Valley Authority
1101 Market St.
Chattanooga, TN 37401
USA

Daryl Hershberger
Siemens Power Corp.
2101 Horn Rapids Rd.
MD #22
Richland, WA
USA

Steve Hetzel
Tennessee Valley Authority
1101 Market St.
Chattanooga, TN 37401
USA

James Hill
Northern States Power
1717 Wakonade Dr.
Welch, MN 55089
USA

Wes Hines
Ohio State Univ.
557 Blenheim Rd.
Columbus, OH 43214
USA

David Holcomb
Oak Ridge National Lab.
Bethel Valley Rd.
Bldg 3500, MS-6010
Oak Ridge, TN 37831
USA

Edward Hollis
NUS Corp.
910 Clopper Rd.
Gaithersburg, MD 20877
USA

Gary Hoscala
Detroit Edison
6400 N. Dixie Hwy.
220 TAC - Fermi 2
Newport, MI 48166
USA

William Hudnall
ABB Combustion Engineering
1000 Prospect Hill
MS 9420-9BB
Windsor, CT 06095
USA

Ewel Hughes
Entergy Operations, Inc.
P.O. Box 756
M&E Bldg.
Port Gibson, MS 39150
USA

Gordon Hughes
Nuclear Electric
Barnett Way
Barnwood
GL47RS
UK

Norman Ichiyen
Atomic Energy of Canada
2251 Speakman Dr.
SPIFI
Mississauga, Ontario, L5L 3C7
CANADA

Bill Immerman
The Mitre Corporation
7525 Colshire Dr.
MS W624
McLean, VA 22102-3481
USA

Laura Ippolito
NIST
Bldg. 225, Rm. B266
Gaithersburg, MD 20899-0001
USA

Simi Jilek
U.S. DOE
NE-73/GTN
Washington, DC 20585
USA

Paul Joannou
Ontario Hydro
700 University Ave.
Toronto, Ontario M5G 1X6
CANADA

John Johnson
Entergy Operations, Inc.
P.O. Box B
Killona, LA 70066
USA

Carl Johnson
U.S. NRC
5640 Nicholson Ln.
Rockville, MD 20852-2738
USA

Wayne Jouse
Univ. of Arizona
Tucson, AZ 85721
USA

Joseph Joyce
U.S. NRC

Seyavash Karimian
PSE&G
Hancock Bridge
NJ 08038
USA

Thomas Kaza
Rochester Gas & Elec. Corp.
89 East Ave.
Rochester, NY 14649
USA

Dennis Kelly
GPU Nuclear Corp.
1 Upper Pond Rd.
Parisippa, NJ 07054
USA

James Kenner
Software Engineering Tech.
12890 Edwin Dr.
Nokesville, VA 22123
USA

William Kerr
Univ. of Mich.
ACRS/USNRC
2355 Bonisteel Blvd.
Ann Arbor, MI 48109
USA

Michael Kirk
NUMARC
1776 I. St., NW
Ste. 300
Washington, DC 20006
USA

Judith Klein
IBM
9231 Corporate Blvd.
861/4D34
Rockville, MD 20850
USA

John Knight
Univ. of Virginia
Dept. of Computer Sci.
Thornton Hall
Charlottesville, VA 22903
USA

Shlomo Koch
Spectrum Technologies
133 Wall St.
Schenectady, NY 12305
USA

Kofi Korsah
Oak Ridge National Lab.
Bethel Valley Rd.
Bldg 3500, MS-6010
Oak Ridge, TN 37831
USA

Joel Kramer
U.S. NRC

Henry Kwok
Westinghouse Electric Corp.
P.O. Box 598
Pittsburgh, PA 15230
USA

Ann-Marie Lapassat
Commissariat a L'Energie Atomique
CEN/SACLAY-IRDI/D.
LETI/DEIN/SUR
9119 GIF SUR Yvette Cedex
FRANCE

John Larkins
U.S. NRC/ACRS
Rm. P-315
Washington, DC 20555
USA

Dennis Lawrence
Lawrence Livermore National Lab.
7000 East Ave.
L-632
Livermore, CA 94550
USA

Shaw-Cuang Lee
Inst. of Nuclear Energy
P.O. Box 3-3
CAEC
Lung-Tan, Taiwan, 32500
CHINA

Eric Lee
U.S. NRC

James Leivo
J.M. Leivo Associates
5615 Catoctin Ridge
Mount Airy, MD 21771
USA

Harold W. Lewis
U.S. NRC/ACRS
4184 Cresta Ave.
Santa Barbara, CA 93110
USA

Paul Lewis
U.S. NRC
5650 Nicholson Lane
NLN-316
Rockville, MD 20852
USA

Hulbert Li
U.S. NRC
11555 Rockville Pike
WFN 8H3
Rockville, MD 20851
USA

Arthur G. Lilienthal
Niagara Mohawk Power Corp.
301 Plainfield Rd.
Syracuse, NY 13212
USA

William Lindblad
U.S. NRC/ACRS
6770 SW Raleighwood
Portland, OR 97225-1923
USA

Richard Linger
IBM
20221 Darlington Dr.
Gaithersburg, MD 20879
USA

Pekka Liuhto
Finnish Centre for Radia.
P.O. Box 268
SF-00101
Helsinki,
FINLAND

Todd Logan
Pacific Nuclear
1111 Paquinelli Dr.
Ste. 100
Westmont, IL 60559
USA

Charles Longo
Penn Power and Light
2 North Ninth St.
A1-2
Allentown, PA 18101
USA

James Lyle
NIST

George MacDonald
U.S. NRC
101 Marietta St., NW
Ste. 2900
Atlanta, GA 30323
USA

Ernst-Ulrich Mainka
TUV-NORD
Grosse Bahnstr 2000
Hamburg 54
GERMANY

Wayne Manges
Oak Ridge National Lab.
P.O. Box 2008
MS 6007
Oak Ridge, TN 37831-6007
USA

Evanglos Marinos
U.S. NRC
11555 Rockville Pike
8H3
Rockville, MD 20851
USA

Ben Martin
Computer Eng. Serv.
240 Forrest Ave.
Ste. 102
Chattanooga, TN 37405
USA

Gerardo Martinez-Guridi
Brookhaven National Lab.
Bldg. 130
Upton, NY 11973
USA

Jerry Mauck
U.S. NRC
OWFN
Ste. 7
Rockville, MD
USA

Randall May
S. Levy Inc.
3425 S. Bascom Ave.
Campbell, CA 95008
USA

Subinoy Mazumdar
U.S. NRC
9 C14
Washington, DC 20555
USA

John McHugh
Portland State Univ.
Computer Science Dept.
P.O. Box 751
Portland, OR 97207-0751
USA

Kirklyn Melson
Hurst Consulting, Inc.
120 E. Myrtle
Angleton, TX 77515
USA

Carlyle Michelson
U.S. NRC/ACRS
1345 Oak Ridge Trpk.
P.O. Box 2500
Oak Ridge, TN 37831-2500
USA

Lee Miller
U.S. NRC
5700 Brainerd Rd.
Osborne Ofc. Ctr., 200
Chattanooga, TN 37411-4017
USA

Greg Miller
EG&G Idaho, Inc.
P.O. Box 1625
MS 3730
Idaho Falls, ID 83415
USA

Lance Miller
Science Appli. International Corp.
1710 Goodridge Dr.
P.O. Box 1301
McLean, VA 22102
USA

Jonathan D. Moffett
Univ. of York
Heslington
York YO1 5DD,
ENGLAND

Bruce Moore
Data Refining Technology
P.O. Box 893
Plaquemine, LA 70765
USA

Robert Mullens
Nuclear Utilities Software
P.O. Box 415
Birdsboro, PA 19508
USA

Jim Munro
Virginia Power
5000 Dominion Blvd.
IN-2NW
Glen Allen, VA 22553
USA

Sandra Murphy
Trusted Information System
3060 Washington Blvd.
Glenwood, MD 21738
USA

Joe Naser
Electric Power Research Inst.
3412 Hillview Ave.
P.O. Box 10412
Palo Alto, CA 94303
USA

Russell E. Neitert
Argonne National Lab.
9700 S. Cass Ave.
RA/208 G
Argonne, IL 60439
USA

Michael Novak
ABB Combustion Engineering
1000 Prospect Hill
9341-4BB
Windsor, CT 06095
USA

Ronald O'Rourke
Sorrento Electronics
10240 Flanders Court
San Diego, CA 92121
USA

Simon Oleson
601 Caroline St.
Ste. 700
Fredericksburg, VA 22401
USA

Lee Ostrom
EG&G Inc.
P.O. Box 1625
3855
Idaho Falls, ID 83415
USA

Loeser Paul
U.S. NRC
OWFN
8H3
Washington, DC 20555
USA

Frederick Paulitz
U.S. NRC
OWFN
8H3
Washington, DC 20555
USA

Stephens S. Payne
ARC Professional Services
601 Caroline St.
7th Fl.
Fredericksburg, VA 22401
USA

William Petrick
Capri Technology, Inc.
14125 Capri Dr.
Ste. 7
Los Gatos, CA 95030
USA

Bob Pikul
The Mitre Corporation
7525 Colshire Dr.
W779
McLean, VA 22102
USA

Kenneth Poorman
MIT
325 Franklin St.
#405
Cambridge, MA 02139
USA

Robert Poston
Interactive Development Envir.
4043 State Highway 33
Tinton Falls, NJ 07753
USA

K.C. Prasad
Toledo Edison Company
5501 N. State Rt. 2
MS 3105
Oak Harbor, OH 43449
USA

David Pullen
Duke Engineering & Services
230 S. Tryon St.
STO-3504
Charlotte, NC 28201-1004
USA

James Raleigh
Southern Technical Services
3 Metro Center
Ste. 610
Bethesda, MD 20814
USA

Ifiti Rana
SCS
42 Inverness Plaza
P.O. Box 2625
Birmingham, AL 35202
USA

D.M. Madhu Rao
Principal Engineer
200 Beta Dr.
Pittsburgh, PA 15238
USA

Howard Rathbun
U.S. NRC
OWFN
MS 8H3
Washington, DC 20555
USA

Dorelle Rawlings
Sorrento Electronics
10240 Flanders Court
San Diego, CA 92121
USA

Ron Reeves
TVA
1101 Market St.
Chattanooga, TN 37402
USA

Michael Rencheck
Duquesne Light
P.O. Box 4
BV-SEB2
Chippingport, PA 15077
USA

Raymond J. Rettberg
GPU Nuclear Corp.
1 Upper Pond Rd.
Parsippany, NJ 07054
USA

Sang Rhew
U.S. NRC
11555 Rockville Pike
8G8
Rockville, MD 20815
USA

Thomas Roberts
U.S. DOE
Office of Nuclear Enrgy
NE-60
Washington, DC 20585
USA

Kenneth C. Rogers
U.S. NRC

Kyle Rone
IBM
3700 Bay Area Blvd.
Houston, TX 77058
USA

Winston Royce
TRW, Inc.
12900 Federal Systems Park Dr.
MS FP1/7165
Fairfax, VA 22033
USA

Steven Rudisail
U.S. NRC
101 Marietta St., NW
Ste. 2900
Atlanta, GA 30323
USA

George C. Rudy
NUS
910 Clopper Rd.
EPM
Gaithersburg, MD 20878
USA

Bill Ruland
U.S. NRC
475 Allendale Rd.
King of Prussia, PA 10406
USA

William Russell
U.S. NRC

Nancy Salgado
U.S. NRC
101 Marietta St., NW
Ste. 2900, DRS
Atlanta, GA 30323
USA

Uma Satyen
The Mitre Corporation
7525 Colshire Dr.
MS S337
McLean, VA 22102
USA

Carl Schaefer
The Mitre Corporation
7525 Colshire Dr.
MS Z646
McLean, VA 22102-3481
USA

Charlotte Scheper
Consultant
708 Misty Isle Place
Raleigh, NC 27615
USA

Edward Schweibinz
U.S. NRC
799 Roosevelt Rd.
Glen Ellyn, IL 60137
USA

Philip Sedgwick
Control Systems Analysis
747 Aquidneck Ave.
Middletown, RI 08842
USA

Gregory L. Sensmeier
Sargent & Lundy
55 East Monroe
25F26
Chicago, IL
USA

Mark Serhal
Wolf Creek Nuclear Oper.
1550 Oxen Lane Corp.
WC-PDE
Burlington, KS 66839
USA

Shiv Seth
The Mitre Corporation
7525 Colshire Dr.
MS W779
McLean, VA 22102-3481
USA

Milton Shymlock
U.S. NRC
101 Marietta St., NW
DRS-PSS
Atlanta, GA 30323-0199
USA

Louis Silva
441 Gateswood Dr.
West Chester, PA 19380
USA

Lynn Simms
Logicon, Inc.
16343 Dahlgren Rd.
Dahlgren, VA 22448
USA

Barry Simon
GE Nuclear Energy
175 Curtner Ave.
MC 765
San Jose, CA 95125
USA

Ian Smith
AEA Technology Consulting
Winfrith
343/B41
Dorchester, Dorset, DT2 8DH
ENGLAND

James Smith
B & W Nuclear Service
3315 Old Forest Rd.
Lynchburg, VA 24506-0935
USA

Rick Smith
SC Electric & Gas
P.O. Box 88
563
Jenkinsville, SC 29065
USA

James Spadafore
Niagara Mohawk Power Corp.
P.O. Box 63
ISEG, R-1
Lycoming, NY 13093
USA

Deirdre Spaulding
U.S. NRC
8H3
Washington, DC 20355
USA

Jack Spraul
U.S. NRC
Rm. OWFN
MS 4H3
Washington, DC 20555
USA

Richard Srodawa
Detroit Edison
6100 West Warren
H-62, WSC
Detroit, MI
USA

George Stark
The Mitre Corporation
7525 Colshire Dr.
MS NASA
McLean, VA 22102-3481
USA

Mark Stella
U.S. NRC/ACRS
Rm. P-315
Washington, DC 20555
USA

Janice Stevens
TRW Systems
12900 Federal Sys. Park
FP1/4166
Fairfax, VA 22033
USA

James Stewart
U.S. NRC
OWFN 8H3
Washington, DC 20555
USA

Jeff Stock
SC Electric & Gas
P.O. Box 88
563
Jenkinsville, SC 29065
USA

Al Sudduth
Duke Power Company
MS EC 10B
P.O. Box 1006
Charlotte, NC 28201-1006
USA

Greg Suski
Lawrence Livermore National Lab.
700 East Ave.
L-632
Livermore, CA 94550
USA

Richard P. Taylor
Atomic Energy Control Board
P.O. Box 1046
Station B
Ottawa, Ontario, K1P 559
CANADA

David Theriault
Westinghouse Electric Corp.
200 Beta Dr.
Pittsburgh, PA 15238
USA

Cecil Thomas
U.S. NRC

Brian Tollefson
Baltimore Gas and Electric
1650 Calvert Cliffs
1st Floor NOF
Lusby, MD 20657
USA

Ray Torok
Electric Power Research Inst.
3412 Hillview Ave.
Palo Alto, CA 94303
USA

Lyne Tougas
AECB
280 Slater St.
Ottawa, Ontario, K1P559
CANADA

Carmen Trammell
Univ. of Tennessee
107 Avers Hall
Knoxville, TN 37996
USA

Steve Troisi
Arizona Public Service
5801 S. Winterburg
Sta. 7465
Tonopah, AZ 85354-7529
USA

Terrence Tuite
Westinghouse Electric Corp.
1740 Golden Mile Hwy.
MS 26
Monroeville, PA 15146
USA

Robert E. Uhrig
Univ. of Tennessee
306 Pasqua Engr. Bldg.
Knoxville, TN 37996-2300
USA

Louis Valeo
Westinghouse Electric Corp.
P.O. Box 79
07DD
W. Mifflin, PA 15122
USA

Carl Vitalbo
Westinghouse Electric Corp.
4350 Northern Pike
MS 410H
Monroeville, PA 15146
USA

Jerry Voss
Tenera Operating Company
7272 Wisconsin Ave.
Ste. 300
Bethesda, MD 20814
USA

John Wado
Westinghouse Electric Corp.
200 Beta Dr.
Pittsburgh, PA 15238
USA

Charles Waite
PSE&G
P.O. Box 236
MC NY7
Hancocks Bridge, NJ 08079
USA

Dolores Wallace
NIST
Bldg. 225, Rm. B266
Gaithersburg, MD 20899-0001
USA

Michael Waterman
U.S. NRC
8H3
Washington, DC 20555
USA

Luther Watson
EG&G Idaho, Inc.
P.O. Box 1625
EG&G
Idaho Falls, ID 83415-2070
USA

Robert Webb
Pacific Gas & Electric Co.
333 Market St.
A1410
San Francisco, CA 94177
USA

Amy Weiss
International Access Corp.
1901 Penn Ave.
Ste. 300
Washington, DC 20009
USA

Mark Wilken
American Electric Power
1 Riverside Plaza
20th Fl.
Columbus, OH 43215
USA

J. Ernest Wilkins
U.S. NRC

Victor Willems
Gilbert Commonwealth, Inc.
P.O. Box 11498
Reading, PA 19603
USA

Robert Williams
TVA
1101 Market St.
LP 4H-C
Chattanooga, TN 37402
USA

John Williams
Univ. of Arizona
Tucson, AZ 85721
USA

Suzie Witterberg
U.S. NRC
OWFN, MS 11A1
Washington, DC 20555
USA

See-Meng Wong
Brookhaven National Lab.
Bldg. 130, BNL
Upton, NY 11973
USA

Charles Wylie
U.S. NRC/ACRS
9610 Lawyers Rd.
Charlotte, NC 28227
USA

Charles Youman
SETA Corporation
6858 Old Dominion Dr.
Suite 200
McLean, VA 22101
USA

Zita A. Yurko
Westinghouse Electric Corp.
5042 Impala Dr.
Pittsburgh, PA 15239
USA

George Ziff
TRW Systems
1 Federal Sys. Park Dr.
FP1/4166
Fairfax, VA 22033
USA

**ANNOUNCEMENT OF NEW PUBLICATIONS ON
COMPUTER SYSTEMS TECHNOLOGY**

Superintendent of Documents
Government Printing Office
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Institute of Standards and Technology Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NIST *Technical Publications*

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published bimonthly for NIST by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program in support of the efforts of private-sector standardizing organizations.

Consumer Information Series—Practical information, based on NIST research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the above NIST publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.

Order the following NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency Reports (NISTIR)—A special series of interim or final reports on work performed by NIST for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce

National Institute of Standards and Technology
Gaithersburg, MD 20899

Official Business

Penalty for Private Use \$300