NISTIR 4656

Proceedings of the Forum on Standards for High Integrity Software (DOD, Government, Industry); Gaithersburg, MD: June 28, 1991

Dolores R. Wallace
Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

Michael Brown
Naval Surface Warfare Center
Dahlgren, VA 22448

Archibald McKinlay VI
McDonnell Douglas Corporation
St. Louis, MO 63166

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR QC100 .U56 4656 1991

NISTIR 4656

Proceedings of the Forum on Standards for High Integrity Software (DOD, Government, Industry); Gaithersburg, MD: June 28, 1991

Dolores R. Wallace
Computer Systems Laboratory
National Institute of Standards and Technologjy
Gaithersburg, MD 20899

Michael Brown
Naval Surface Warfare Center
Dahlgren, VA 22448

Archibald McKinlay VI
McDonnell Douglas Corporation
St. Louis, MO 63166

September 1991

DEPARTMENT OF COMMERCE ★ UNITED STATES OF AMERICA ★

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

ABSTRACT

This report provides information related to the Forum on Standards for High Integrity Software (Department of Defense, Government, and Industry) held at the National Institute of Standards and Technology on June 28, 1991. At the forum, software engineering experts presented their perspectives on the role of software engineers in software safety, a comparison for safety and computer security issues in standards, hazard analysis, assurance standards, and software certification. Future directions for NIST activities for assurance of high integrity software were proposed.

Keywords: assurance; certification; computer security; hazard analysis; software safety; standards.

ACKNOWLEDGMENT

The coordinators of the Forum on High Integrity Software express their gratitude to the following persons who made the forum an informative event:

o Al Friend of the U.S. Navy Space and Naval Warfare Systems Command for suggesting that the National Institute of Standards and Technology and the COMPASS organization host a forum on high integrity systems immediately following the COMPASS '91 Conference,

o presenters John Knight, Dev Raheja, Frank Houston, and Janet Dunham, and

o rapporteurs Michael Brown and Archibald McKinlay.

TABLE OF CONTENTS

1. INTRODUCTION

The Forum on Standards for High Integrity Software was held at the National Institute of Standards and Technology (NIST) on June 28, 1991. This forum followed the COMPASS '91 Conference on computer assurance and was sponsored by both NIST and the COMPASS organization. Representatives of industry, government, academia, and defense organization from the United States, Canada, and England attended the forum.

The purpose of this forum was to provide opportunity for the exchange of information on activities concerning standards for the assurance of high integrity software and opportunity for commentary on related issues. The coordinators of this forum expect the ideas and recommendations in this report to stimulate other insightful thoughts and recommendations. [1]

Section 2 of this report provides a brief summary of the NIST Workshop on Assurance of High Integrity Software, January 22-23, 1991; the complete report appears in NIST Special Publication 500-190 [1]. Sections 3 -5 provide descriptions of presentations at the Forum on June 28. Section 3 contains a summary of the presentation by Dr. John Knight concerning software safety and the role of the software engineer.

Section 4 contains an overview of the comparison of software safety and computer security standards, prepared by Richard Kuhn and Dolores Wallace of NIST. When the study is completed, it will be published as a NIST report.

Section 5 provides a brief synopsis of the final three presentations:

- o Dev Raheja of Technology Management, Inc., on Hazard Analysis,
- o M. Frank Houston of the Food and Drug Administration (FDA) on his perspective about assurance standards, and
- o Janet Dunham, Director of the Center for Digital Systems Research at Research Triangle Institute, on software certification.

Section 6 lists plans for future workshops and forums. Appendix A. lists names of forum participants. Appendix B contains the presentation materials.

[1] Coordinators are Dolores Wallace and Rick Kuhn of NIST and John Cherniavsky of the National Science Foundation. Send comments to Dolores Wallace, National Institute of Standards and Technology, Technology Building, B266, Gaithersburg, MD 20899.

2. OVERVIEW OF NIST WORKSHOP IN JANUARY 1991

NIST hosted a Workshop on the Assurance of High Integrity Software on January 22-23, 1991. While there were opening and closing plenary sessions, most of the workshop consisted of parallel working group sessions. The technical topics and the chairs of the sessions are the following:

- o Assurance Techniques, Dr. Susan Gerhart, MCC,
- o Cost-Benefit Framework, Dr. John Knight, University of Virginia,
- o Hazard Analyses, Michael Brown, Naval Surface Warfare Command, and
- o Controlled and Encouraged Practices, Archibald McKinlay, McDonnell Douglas Corporation.

Each session chair has expressed willingness to continue work on the charter of its working group. The efforts need to be coordinated. The work of these groups and NIST will be presented at annual workshops at NIST.

The techniques session proposed a template for describing assurance techniques. Many techniques may be ready for transition from the research community into practice. The group selected seven techniques for initial study. NIST will coordinate work on these techniques and will seek sponsorship by interested parties. While the cost-benefit session selected an initial model for representing choice of techniques relative to cost, the group needs to collect substantial amounts of data concerning the use of assurance techniques.

The hazard analysis group identified differences in criticality assessments based on computer security objectives and software safety objectives. They presented a model useful to both communities. Additional work at NIST will describe the types of hazard analyses and techniques used to support them. The controlled and encouraged practices group suggested that design and development practices which are difficult-to-analyze now may become less so as technology improves. The group recommended an approach for combining a difficult-to-analyze practice with a technique that may verify correct usage of the practice or may override inherent dangers of the practice.

While each group worked individually, presentations at the closing session showed consensus on many issues; the following three topics are among them.

Definitions: Any standards and guidance documents must define principal terms and the context in which they are used.

Software safety and computer security: COMPASS '89 provided discussion on differences and similarities between assurance for computer security and assurance of software safety. The distinctions are not well understood. As a result of the workshop recommendation, NIST is analyzing a collection of standards to identify the similarities and differences in assurance requirements.

No single standard: Many existing and evolving standards address high integrity issues. The workshop participants agreed that no single standard could address all issues. NIST is attempting to identify how to coordinate an integrated body of standards and guidance that adapts or adopts existing standards wherever possible. NIST is examining many standards and guidance documents on high integrity systems. Features that are being studied include definitions, consistency of requirements, conflicts of requirements, differences between security approaches and software safety approaches, and use of levels of assurance. All required techniques are being cataloged. NIST is conducting an effort to correlate the required techniques to functions existing in Computer-Aided Software Engineering (CASE) tools; the objective is to identify those techniques which need to be automated.

3. SOFTWARE SAFETY AND THE ROLE OF THE SOFTWARE SAFETY ENGINEER

At the Forum on June 28, 1991, Dr. John Knight of the University of Virginia discussed the role of the software engineer in the software safety process. With the increased use of software in safety critical systems, more and more standards are beginning to impact the software engineer's work. This requires that the software engineer be trained in these standards. However, papers on software safety focus on system safety without defining precisely what software safety is. Dr. Knight notes that software engineers will have to be able to distinguish between software safety and system safety in order to define their role in the development of high integrity systems. This also requires definition of individual responsibility within the context of individual efforts. "Good Software Engineering Practices" are discussed in much of the literature but there is little consensus on much of the issue.

Dr. Knight noted that in isolation, software is not hazardous but that does not imply that software safety is meaningful only in the context of the entire system. The Software Engineer is not qualified to deal with the system engineering issues, including hazard analyses; hazards should not appear in the software specifications. The treatment of the hazard must be in the software specification and these requirements must be implemented correctly with very high assurance. The systems engineer must be the one who evaluates the hazards, assesses the risks and establishes the required probabilities. Specification errors must be considered the responsibility of the systems engineer and implementation errors must be the responsibility of the software engineer. The software engineer must understand and identify the system level effects of software.

Another perspective on software safety and the role of software engineers may be found in [2].

4. SOFTWARE SAFETY & COMPUTER SECURITY STANDARDS

Dolores Wallace presented an overview of software safety and computer security standards, for Rick Kuhn, also of NIST, who was unable to attend the forum.

Standards may serve many purposes and in particular may provide the following:

- o a yardstick for comparing systems,
- o a means of specifying requirements, and
- o a means of improving the state of practice.

Forty-four standards, draft standards, and guidance documents comprise the basis for the analysis presented at the forum. These are national or international standards that address security in general, electronic funds transfer (EFT), weapons systems, safety in general, process control, medical devices, civil aviation, and verification and validation. While these documents address high integrity issues, most address either computer security or software safety. Some, such as standards for software verification and validation, do not differentiate between the two.

Safety standards concern systems where the consequences of failure may involve damage to human life, property, and the environment. Security standards concern system that must guarantee the properties of confidentiality, integrity, and availability in spite of accident or malicious attack.

The general security and EFT standards tend to have the most stringent requirements for software, and are often at the functional level. While the weapons systems and safety standards have stringent requirements, the requirements tend to consist more of activities that must be performed rather than specific functions that must be implemented to assure safety. Product attributes may be suggested by safety standards rather than required as in many security standards. Overall there seems to be little relationship between degree of risk and the rigor of the standard.

Those standards that have levels of assurance seem to have improved the state of practice by providing a well understood means of differentiating products. There needs to be an efficient evaluation program of products whose manufacturers claim conformance to standards. In some areas, the assurance process for safety and security is similar; thus it may be possible to develop some common assurance requirements and standards.

5. POSITION STATEMENTS

A forum provides an opportunity for participants to express their opinions on topics related to a specific forum. For this forum, there were three requests to present information on the following topics:

o Hazard analysis,

o Thoughts about standards for high integrity software, and

o Software certification.

5.1 Hazard Analysis - Dev Raheja

Dev Raheja, of Technology Management, Inc., presented his views on hazard analysis. Software reliability and maintainability are elements of software availability. Software safety, human factors (including robustness) and software security are elements of software dependability. Both availability and dependability are elements of the overall software integrity. One approach to support assurance of these attributes is through the use of software hazard analysis. Software hazard analysis is the process of identifying (evaluating and controlling) safety critical software components and events in the early design stage and for the entire life cycle from concept to retirement.

Most techniques for software hazard analysis have their origin in hardware engineering. There is a need to develop automated tools that are not independent of the design process but are integrated into it. Comments on this recommendation generated discussion on the roles of Total Quality Management (TQM), software verification and validation (V&V), and independent V&V. One task of independent V&V may be to perform a hazard analysis at each step of development. Some participants believe that the notion of TQM means implicitly that V&V processes must be conducted as part of development, and those processes include hazard analysis.

5.2 Standards for High Integrity Software - Frank Houston

Frank Houston, of the Food and Drug Administration (FDA), has had opportunity to study the effectiveness of software standards. From this perspective, he has made observations about some features standards must have to be effective; his paper, which is included in Appendix B, represents his viewpoint only, not that of the FDA.

Standards for high integrity software may lead to one of the four following results:

1. Firms will conform to the standard and product safety will improve.

2. Firms will not conform to the standard but product safety will improve anyway.

3. Firms will conform to the standard and product safety will not improve.

4. Firms will not conform to the standard and product safety will not improve.

Only one of these outcomes is favorable. What kind of compliance can we achieve with standards? Evidence indicates that standards have been of tremendous benefit in many areas. Hardware items are built to accepted standards which provide assurance that the items will perform as expected and reliably. However, there are no such standards related to software for medical devices or most other products.

Does having a standard mean that things are necessarily improved? Standards are useful only if they result in an improvement in the quality, reliability, safety or other attributes of the system. Standardized definitions are necessary such that everyone understands the word or concept in the same context. Some proposed definitions appear in Appendix B in the paper by Houston.

The problems that standards address must first be clearly defined. The quality of the development process is a necessary consideration. Can things improve from the mere existence of standards, even if the manufacturers do not comply? We must ensure that standards fully address the principal concerns and that results of using of using them can be measured. Ideally things will improve as result of full compliance with the standards.

5.3 Software Certification - Janet Dunham

Janet Dunham, Director of the Center for Digital Systems Research of the Research Triangle Institute (RTI) presented plans for a program she is developing at RTI leading to software certification and accredited laboratories for software certification.

The ability to provide software certification will derive from development of many components. The "roadmap" components consist of the following:

o supporting technology,

o education and training,

o developer and certifier qualifications, and

o standards.

The certification process includes all activities conducted to assure product integrity. It must be based on a certifiable development methodology that permits diversity of the process itself. It must provide evidence that the implemented process and product are of high integrity. The certification process must also impose standards and recommend guidelines on the development process as well as evidence produced. The effort to certify a development methodology is very slow and needs additional work.

Two specific tasks that are required are the development of the supporting technology and education and training. Supporting technology includes the certifiable development technology, support for the development process, requirements on certification evidence and support for the evaluation process. Education is required for the end users, developers, and evaluators / certifiers. One question that arises is "How can a certification process be adapted for application-specificity?" Another problem is to identify the technology that is ready for transition into practice.

6. PLANNING FOR NEXT NIST WORKSHOPS

John Knight of the University of Virginia has offered to host a meeting at Charlottesville, VA, to discuss basic definitions. For this meeting, a core group of about 12 people would be ideal. Terms that are unique or frequently used need to be determined and definitions captured or developed. Existing standards will be used as sources. The definitions must be structured such that they lend themselves to the certification of the final product; they must be defined so that there is an identifiable attribute or quality to the term.

To continue the efforts begun by the session chairs at the January 22-23, 1991, workshop, session chairs may convene working group meetings. These may be scheduled at sites other than NIST if hosts can be identified.

The plans call for an annual one day workshop at NIST in the spring to provide opportunity for comments on results of working groups and progress at NIST on the assurance of high integrity software.

7. REFERENCES

[1] Wallace, Dolores R., D. Richard Kuhn, and John C. Cherniavsky, "Proceedings of the Workshop on High Integrity Software; Gaithersburg, MD; Jan.22-23, 1991," NIST SP 500-190, National Institute of Technology and Standards, Gaithersburg, MD 20899, 1991.

[2] Brown, Michael L., "Software Systems Safety and Human Errors," COMPASS '88, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08855-1331.

APPENDIX A. FORUM PARTICIPANTS

Edward Addy	Logicon, Incorporated
Paul Anderson	U.S. Navy Space and Naval Warfare Command
Leo Beltracchi	U. S. Nuclear Regulatory Commission
John Boone	Institute for Defense Analyses
Michael Brown	Naval Surface Warfare Center
Jeff Bulow	General Electric
Dominic Chan	Sheridan Park Research (CAN)
John Cherniavsky	National Science Foundation
Janet Dunham	Research Triangle Institute
Clif Ericson	The Boeing Company
Al Friend	U.S. Navy Space and Naval Warfare Systems Command
Jan Feisenger	The Mitre Corporation
Roger Fujii	Logicon, Incorporated
Louis Gieszl	Johns Hopkins University/APL
Janet Gill	Patuxent Naval Air Test Center
Larry Hatch	National Security Agency
Henry Heffernan	EDP News Services
Jill Hill	Rolls-Royce & Associates Limited
Mark Horvath	Fairchild Defense Corporation
Jim Ionata	Naval Underwater Systems Center
Diane Jachinowski	Nellcor Incorporated

Joe Jungbluth	Fairchild Defense
Bill Junk	University of Idaho
Darrell Kienzle	University of Virginia
Arch McKinlay	McDonnell Douglas Corporation
Bill McMinn	Fairchild Defense
Dave Murray	GE-GESD
Wendy Peng	NIST
Ted Ralston	MCC
Dev Raheja	Technology Management, Inc.
Cliff Richmond	Booz Allen & Hamilton
Charlotte Schepper	Research Triangle Institute
Philip Sedgwick	Control Systems Analysis
Don Sova	NASA Headquarters
Laura Strigel	NIST
Antonio Tomasome	Ontario Hydro
Dolores Wallace	NIST
Harold Walpert	PEO
Dr. Peter Wilkinson	Dept. of Trade & Industry, UK
Phil Windley	University of Idaho
Ed Zieglar	Department of Defense

APPENDIX B. PRESENTATIONS

FORUM
Standards for High Integrity Software

AGENDA

o REGISTRATION

o OPENING; DISCUSSION: NIST JANUARY WORKSHOP
Dolores Wallace, NIST

o SOFTWARE SAFETY & THE ROLE OF THE SOFTWARE ENGINEER Dr. John Knight, University of Virginia

o COMPUTER SECURITY & SOFTWARE SAFETY STANDARDS
Rick Kuhn, NIST

o BREAK

o POSITION STATEMENTS:

- Hazard Analysis	Dev Raheja, Consultant
- Standards for High Integrity Software	Frank Houston, FDA
- Software Certification	Janet Dunham, RTI

o BREAK

o PLANNING for NEXT NIST WORKSHOPS
Dolores Wallace, Rick Kuhn, NIST
John Cherniavsky, National Science Foundation

o ADJOURN by 1:30 p.m.

FORUM

Standards for High Integrity Software

(DOD-GOVERNMENT-INDUSTRY)

June 28, 1991
NIST, Gaithersburg, MD

coordinated by

Dolores R. Wallace
NIST (301) 975-3340, wallace@swe.ncsl.nist.gov

Rick Kuhn
NIST (301) 975-3337, kuhn@swe.ncsl.nist.gov

John C. Cherniavsky
National Science Foundation, (202) 357-7349
chernia@note.nsf.gov

FORUM
Standards for High Integrity Software

AGENDA

- **REGISTRATION**

- **OPENING; DISCUSSION: NIST JANUARY WORKSHOP**
  Dolores Wallace, NIST

- **SOFTWARE SAFETY & THE ROLE OF THE SOFTWARE ENGINEER** **Dr. John Knight, University of Virginia**

- **COMPUTER SECURITY & SOFTWARE SAFETY STANDARDS**
  Rick Kuhn, NIST

- **BREAK**

- **POSITION STATEMENTS:**

- Hazard Analysis	Dev Raheja, Consultant
- Standards for High Integrity Software	Frank Houston, FDA
- Software Certification	Janet Dunham, RTI

- **BREAK**

- **PLANNING for NEXT NIST WORKSHOPS**
  Dolores Wallace, Rick Kuhn, NIST
  John Cherniavsky, National Science Foundation

- **ADJOURN by 1:30 p.m.**

REPORT: NIST WORKSHOP ON ASSURANCE OF HIGH INTEGRITY SYSTEMS

WORKSHOP DESCRIPTION

- o SCOPE: HIGH INTEGRITY SOFTWARE
- o WHY HIGH INTEGRITY?
- o WHY NIST EFFORT?

REPORT: NIST WORKSHOP ON ASSURANCE OF HIGH INTEGRITY SYSTEMS

- CONCERNS AND RECOMMENDATIONS
  - Basic definitions
  - Safety versus security
  - Practical assurance limit
  - Evaluation, tracking use of techniques
  - Techniques: design flaws & quality
  - One set of techniques inappropriate; levels
  - No single standard
  - Resources in the literature
  - Experiment
  - Conformance, accreditation, certification
  - Bibliography
  - Education
  - CASE and tool vendors

REPORT: NIST WORKSHOP ON ASSURANCE OF HIGH INTEGRITY SYSTEMS

WORKING GROUPS

- TECHNIQUES
  - Template
  - Initial Study: Cleanroom; Formal specification languages EHDM, FDM/Inajo, Estelle, Larch; Petri-net based tool IDEF0; Traces; Verification and Validation; Statecharts
- COST-BENEFIT FRAMEWORK
  - Dependent on info from other groups
  - Model
- HAZARD ANALYSIS
  - Identify techniques
  - Hazard Criticality Chart
- CONTROLLED, ENCOURAGED PRACTICES
  - Eliminate "forbidden practices"
  - Use controlled and encouraged practices

REPORT: NIST WORKSHOP ON ASSURANCE OF HIGH INTEGRITY SYSTEMS

CONSENSUS ON APPROACH, TASKS

- o RECOMMENDATIONS
    - NO EXISTING OR EVOLVING STANDARD WILL SATISFY THE BREADTH OF NEEDED STANDARDS AND GUIDANCE.
    - NIST SHOULD COORDINATE AN EFFORT TO PRODUCE AN INTEGRATED BODY OF GUIDANCE.
    - NIST SHOULD ADAPT OR ADOPT EXISTING STANDARDS AS APPROPRIATE.
    - NIST SHOULD CONTINUE ITS EFFORT TO MAKE HIGH INTEGRITY ISSUES HIGHLY VISIBLE!

REPORT: NIST WORKSHOP ON ASSURANCE OF HIGH INTEGRITY SYSTEMS

NIST TECHNICAL PROGRESS

- o STANDARDS BIBLIOGRAPHY
- o HAZARD ANALYSIS EFFORT
- o ERROR ANALYSES
- o FORMAL METHODS
  - Laboratory
  - Medical device
  - Correctness proving
  - Specification of standards
  - Slicing

REPORT: NIST WORKSHOP ON ASSURANCE OF HIGH INTEGRITY SYSTEMS

<u>PROGRESS AND PLANS FOR NIST EXTERNAL ACTIVITIES</u>

- o CONTACTS AND PARTICIPATION WITH MAJOR STANDARDS GROUPS AND ASSOCIATIONS

- o CONTINUATION OF CURRENT ACTIVITIES

- o FUTURE WORKSHOPS

  - Forum on Friday June 28

  - Small study or review groups

  - Annual meeting

- o FUTURE TOPICS

SOFTWARE SAFETY
THE ROLE OF THE SOFTWARE ENGINEER

John C. Knight Darrell M. Kienzle

Department of Computer Science
University of Virginia

SOFTWARE SAFETY

- Public Exposure To Digital Systems Increasing - Serious Problem
- Several Standards Written Or Being Written
- Lots Of Papers On Software Safety:
  - Extremely Valuable And Important Contributions To The Topic
  - But, They Tend To Stress System Safety
- Precisely What Is Software Safety?
- What Is The Role Of The Software Engineer?
- What Is The Responsibility Of The Software Engineer?
- What Is ''Good Engineering Practice'' In This Case?
- What Technology Is The Software Engineer Required To Use?

UVA
Department of Computer Science

SOME TIME-HONORED ANECDOTES

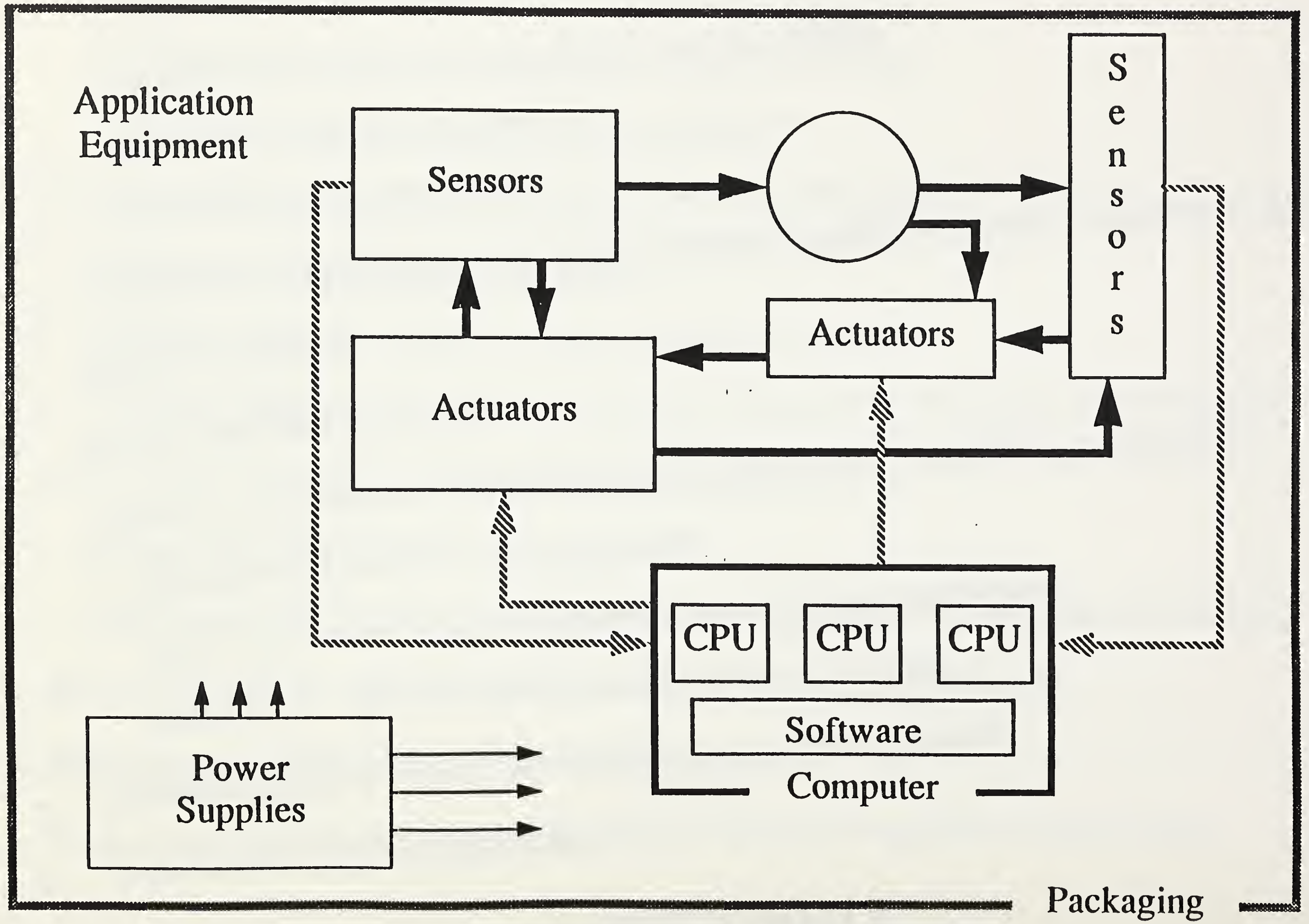
- Aircraft Landing Gear Raised While Aircraft On Ground:
  - Test Pilot Input During Ground Test, Aircraft Damaged
  - ''Operational Profile or Specification In Error''
- Computer Controlled Chemical Reactor Seriously Damaged:
  - Mechanical Alarm Signal Generated
  - Computer Kept All Controls Fixed - Reactor Overheated
  - ''Systems Engineers Had Not Understood What Went On Inside The Computer''
- F18 Missile Clamped To Wing After Engine Ignition:
  - Aircraft Out Of Control
  - ''Erroneous Assumption Made About Time For Engine To Develop Full Thrust''
- All Are Important, Very Serious Incidents - Valuable Insight Gained
- What *Exactly* Is The Safety Issue In Each Case?
- What *Exactly* Is The Responsibility Of The Software Engineer In Each Case?

SYSTEM SAFETY

- Informally, Safety Is Subjective
- Systems Engineers Have Formalized The Notion Of Safety:
  - Definitions - Hazard, Risk, Acceptable Level Of Risk
  - View System As A Well-Defined Collection Of Components
  - Established Practices And Procedures
- Software Researchers And Engineers Trying To Do The Same For Software
- So Far, Success Is Limited
- Within A System:
  - Software Is Merely Part Just Like Computer Hardware, Sensors, Actuators, Etc.
  - Software Can Cause Failure
  - Software Can Prevent Failure
  - Software Can Stand By And Watch Failure Happen
  - So Can Any Other Part

UVA
Department of Computer Science

SOFTWARE SAFETY vs. SYSTEM SAFETY



UVA
Department of Computer Science

SYSTEM CONTEXT FOR SOFTWARE

- Common Observation - In Isolation, Software Is Never Unsafe:
  - True, It Cannot Be Executed In Isolation Either
  - Software Is Useless In Isolation
- In Isolation, Software Is Removed From The Notion Of Hazard:

  This Does Not Imply That Software Safety Is Meaningful Only In The Context Of The Entire System
- The Software Engineer Is Not Qualified To Deal With Systems Engineering Issues
- Hazards, Risks, Etc. Should Not Appear In The Software Specification
- The *Required Treatment* Of Hazard Must Be Present In The Software Specification
- Software Safety Specifications Are Those Specifications That Must Be Implemented Correctly With Very High Assurance.
- Required Probabilities Are Determined By The Systems Engineer

UVA
Department of Computer Science

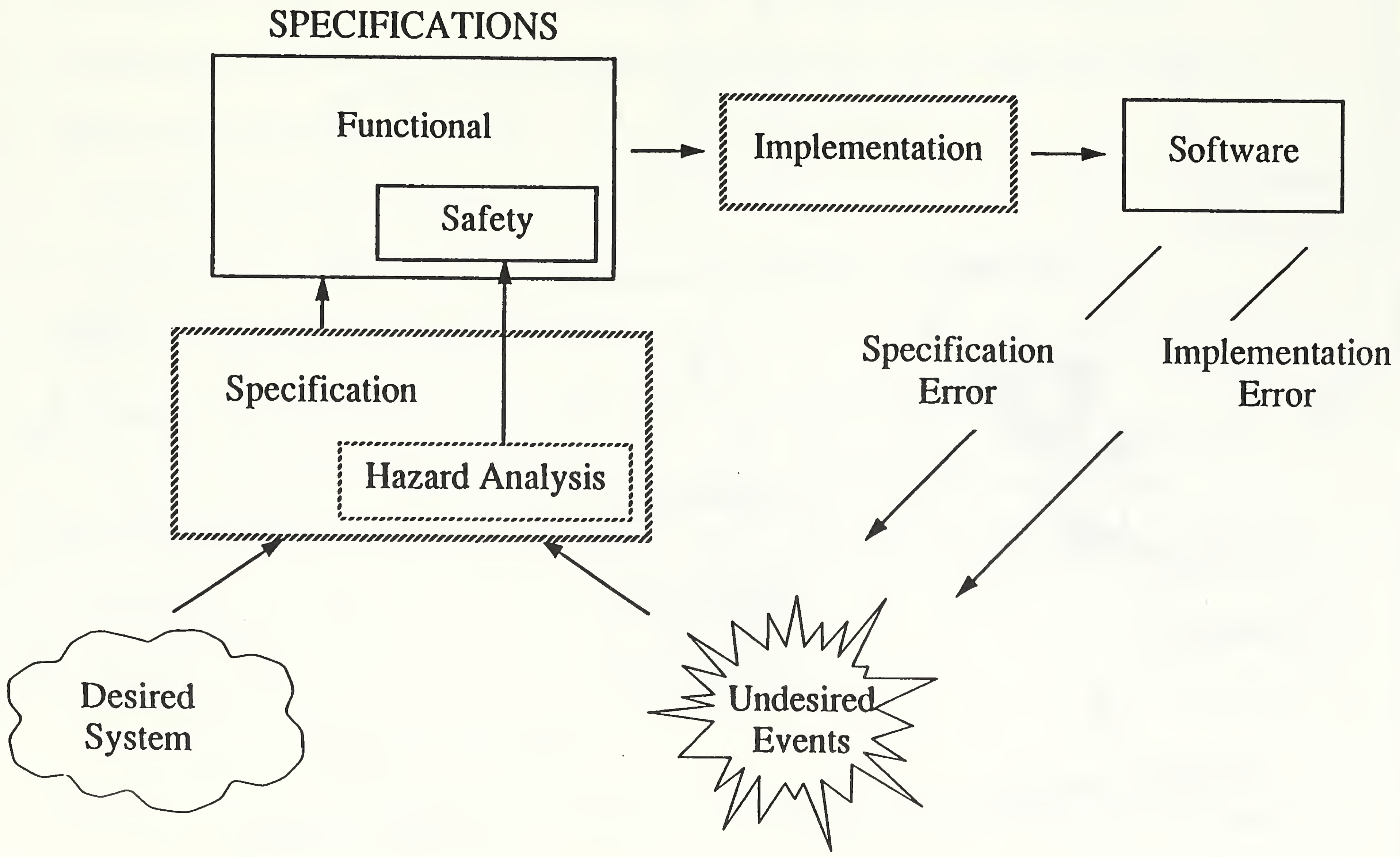
THE ANECDOTES AGAIN

- Aircraft Landing Gear Raised While Aircraft On Ground:
  - ''Operational Profile or Specification In Error''
  - *Systems Engineer's Responsibility*
- Computer Controlled Chemical Reactor Seriously Damaged:
  - ''Systems Engineers Had Not Understood What Went On Inside The Computer''
  - *Systems Engineer's Responsibility*
- F18 Missile Clamped To Wing After Engine Ignition:
  - ''Erroneous Assumption Made About Time For Engine To Develop Full Thrust''
  - *Probably The Systems Engineer's Responsibility*
- In General, Software Engineer Is Not Trained To Identify Hazards, Consider:

  A Flight Control System Commands An Aircraft To Flare On Final Approach At An Air Speed Of 128 Knots, Height 180 Feet, 15 Knot Headwind, Throttles At 75%, MLS On, Fuel At 14%.
- Is This A Hazard?

UVA
Department of Computer Science

CONCEPTUAL FRAMEWORK



UVA
Department of Computer Science

Software Safety and Security Standards

D. Richard Kuhn
NIST, TECH B266
Gaithersburg, Md. 20899
301/975-3337
kuhn@swe.ncsl.nist.gov

Safety Critical Systems

- Systems "where the consequences of failure may involve danger to human life, property, and the environment."

  - N. Leveson, CACM, Feb. 1991

- Examples: civil aviation, medical, nuclear power, weapons systems, electronic funds transfer

Security Critical Systems

- Systems that must guarantee properties of confidentiality, integrity, and availability in spite of accident or malicious attack.

- Examples: intelligence systems, medical data bases, weapons systems, electronic funds transfer

Purposes of Standards

- Provide a yardstick for comparing systems
  - commonly understood reference point
- A means of specifying requirements
  - less duplication of effort
  - greater precision/ less ambiguity
- A means of improving the state of practice
  - common sets of requirements
  - mandatory standards & exceptions
  - levels in standards

Safety and Security Standards
- Database of 44 standards

- security (general) - 10
- electonic funds transfer - 9
- weapons systems - 5
- safety (general) - 5
- nuclear power - 5
- process control - 1
- medical devices - 1
- civil aviation - 1

- (generic software stds - 7)

Safety and Security Standards

- Database of 44 standards

- Security standards
  - more product oriented
  - process requirements too
  - more specific, mandatory
  - emphasis on response to threats

- Safety standards
  - more process oriented
  - product attributes tend to be suggested, not required
  - emphasis on identifying hazards

- Overall
  - little relationship between degree of risk and rigor of standard
  - safety requirements $\subseteq$ security requirements

Rigorous Standards

- Trusted Computer Security Evaluation Criteria
  - U.S. DoD, 1983
- Software System Safety
  - U.S. Air Force, 1985
- Criteria and Procedures for Certifying Message Authentication Devices for Federal EFT Use
  - U.S. Treasury, 1986
- Information Technology Security Evaluation Criteria
  - France, Germany, Netherlands, U.K., 1990
- Defence Standard 00-55
  - U.K. MoD, 1991
- Security Requirements for Cryptographic Modules
  - NIST, 1991

Lessons of Security Standards

- Subsetting of requirements works
  - separates concerns
- Reference monitors work
  - where policy to be enforced is small
- Levels help improve the state of practice
- Efficient evaluation program is critical

(tentative) Conclusions

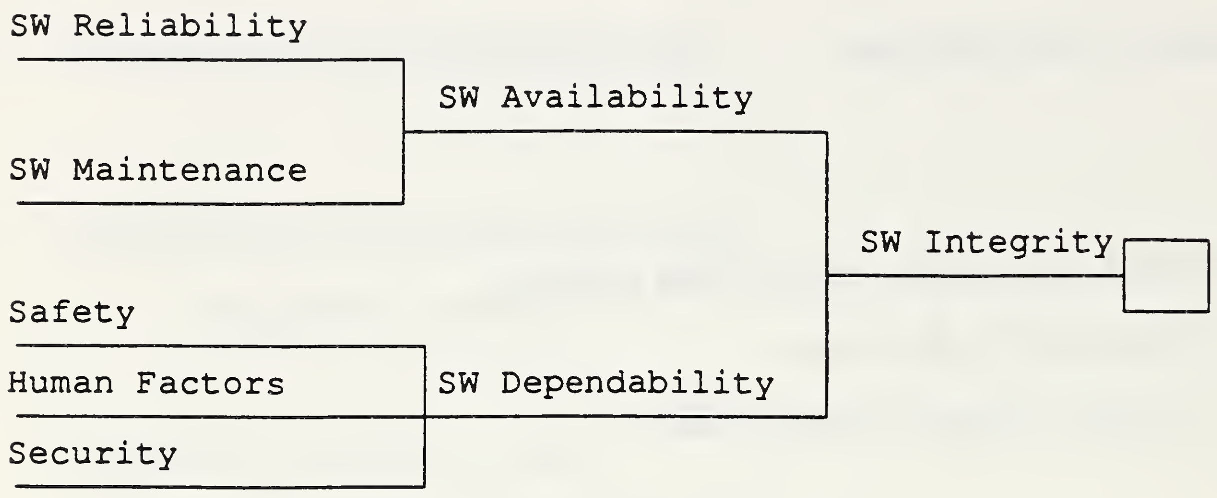
- Assurance process for safety and security
  is essentially the same
  - should become more similar

- Common assurance requirements/standards
  for safety and security are practical
  - at least a large subset in common
  - moving in this direction

- (Security) integrity is closer to
  safety than is confidentiality
  - but neither is well understood

HAZARD ANALYSIS POSITION STATEMENT

by Dev Raheja



<u>WHAT IS SOFTWARE HAZARD ANALYSIS?</u>

It is a <u>process</u> of identifying safety critical software components and events in early design cycle, for the entire life cycle from concept to retirement.

CURRENT TECHNIQUES (mostly from hardware world)

requirements level hazard analysis
high level design hazard analysis
detailed design hazard analysis
code level hazard analysis
preliminary hazard analysis
subsystem hazard analysis
fault tree analysis
software system FMEA
Petri nets
precondition/post-condition hazard analysis
proactive/reactive system hazard analysis
cutset analysis
common cause analysis
event trees
sneak condition analysis
HAZOP

NEEDS

Integration of individual techniques in a system hazard analysis model

Development of automated tools which are not independent of design process but integrated into it

Training of project managers to allow for budgets which are grossly misdirected

Development of new techniques for software stand alone to be used by programmers

A PERSONAL POSITION ON STANDARDS FOR HIGH INTEGRITY SOFTWARE

by M. Frank Houston

Please take the following treatise in the spirit of skeptical support of standards for high integrity computer systems. I personally agree with the principles that have led to this gathering of experts, but I have some reservations and some suggestions that I strongly believe to be pertinent. The issues I raise may be old, general and philosophical questions, but I think they need to be addressed because the answers will bear directly on the success of this project.

I fear that we, just as the software developers whom some of us occasionally vilify, may have decided to implement before considering our goals sufficiently. If we produce a standard, four different scenarios may follow: 1) Firms will conform to the standard and product safety will improve, 2) Firms will not conform to the standard but product safety will improve anyway, 3) Firms will conform to the standard but product safety will not improve, or 4) Firms will not conform to the standard and product safety will not improve.

Three of the four possible outcomes are unfavorable. (If product safety improves despite nonconformance, then why spend the time and effort to write a standard?) How will we know if the standard is effective? How will we know if it is not? "Garbage In: Garbage Out" applies universally, and ill-framed goals will lead to wasted effort. I very much want the fruits of my work to be effective, hence the discussion that follows.

Goals and Objectives: Ends and Means

What is the goal of this activity? Our employers, whether they be taxpayers or stockholders, deserve to know. We should decide clearly and remain constant to the decision, unless insurmountable obstacles turn us aside.

As I see it, the participants divide into roughly two camps with different views of the goal. One camp may believe the goal is a standard, assuming that the problems are well defined, the need has been established, and the form of the standard is obvious. When the standard is published, their commitment probably ends.

The other camp, in which I count myself, may believe the goal is to solve problems. A standard is only one of many ways to solve problems. To me, the problems are not well defined, the need for a standard is open to debate, and the form of such a standard is far from obvious. Holding such a view, I am committed to the work indefinitely.

If the goal, by group consensus, is to write a standard, what should the standard address? Typical goals for standards come under the heading of uniformity including such attributes as performance, measurement, process, compatibility, and so forth. Should this standard define a uniform or minimum acceptable process, or should it define a process for producing systems with a uniform or minimum incidence of problems.

Is the standard the end we seek or is it a means to that end? The answer to that question determines how we may measure the success of this project or the lack thereof. Is it success if we achieve consensus and write a standard within a set number of months? Is it success if we produce a standard to which the whole world conforms? Is it success if we produce a standard that reduces the incidence of problems for those who use it? Is it failure to have accomplished none of these ends but to have learned enough to mount a better effort?

In my experience with voluntary standards, no committee has ever considered monitoring the performance of a standard, except to count the groups that claim to conform. Nevertheless, I suggest that we should try to measure other outcomes of this work because we are addressing systems that may jeopardize lives and health as well as wealth. I shall return to the issue of measurement after I discuss problem definition.

Defining the Problem

I believe there is a problem to solve, whereas some folks may disagree. All of us have read about the software crisis [Shore 1985], the Bugs in the Program, [Paul 1989] and the Computers at Risk [NRC 1991]. I submit that we must state the problem carefully because our definition of the problem will affect the role or roles we choose for a standard.

In my opinion, the problem centers on trust. Individuals and organizations want to trust or need to trust computerized products or systems, but sometimes the products or systems have failed that trust. How should we state this problem? Here are several alternatives:

- Trust in computerized systems and products is misplaced.
- Computerized systems and products cannot be trusted.
- Computerized systems and products often fail in one or more trusted applications.
- Computerized systems and products fail too frequently in one or more trusted applications.

- Computerized systems and products fail in one or more trusted applications more often than one should expect given current knowledge.

The first two statements suggest that trust itself is the problem. Some experts might agree; however, "Attempting to dissuade people from this course [using computers in process control] because we do not yet have perfect software engineering techniques would be hopeless." [Leveson 1991] Trust is a social phenomenon which we have little power to change, and I do not believe that any practical standard can solve the problem. We technologists have uncorked the bottle, and the genie is loose.

The third and fourth statements may be true depending on several variables, among them, the potential outcome, the likelihood of that outcome, and the effect of the outcome on both the customer and the purveyor. Can any practical standard cover the multitude of situations implicit in those three variables?

The last statement may or may not be true, but one can devise quantitative ways to test it. The words "often" and "frequently" appear in three of the five problem statements. A standard for measuring frequency of failure or "mishap," might be useful.

Should we try to convert mistrust into trust? I think not; healthy skepticism is just about the only way to prevent inappropriate computer applications. Should we try to ensure that trust (misplaced or not) can be justified to the maximum degree consistent with current knowledge? Of course, but then we must stay current with new knowledge. Should we try to devise standard ways to measure and estimate the trustworthiness of computerized systems? The issue of measurement comes up again and again.

Measurement and Standards

So much for the global problem. What operational problems should this work address? I suggest that credible standards will pass three tests: (1) One can observe all of the standardized attributes of an object. (2) The choice of standardized attributes furthers the goal of the standard. (3) The user can measure some of the standardized attributes.

In my opinion, the software standards that I have read pass the first test. They pass the second test if the goal is to impose a standard process for producing system documentation, but they fail miserably if the standard writers had any other goal. Representatives of industry continually ask me, "What is the connection between documenting software development and the safety or fitness of the final product?" I have no answer. As for the third test, I suggest that none of the current software standards specify any measurable attributes for the product.

Standards need a more concrete basis than opinion or theory. As technological benchmarks, they must stand on foundations of factual knowledge based on measurements. Current software standards address uniformity of process, but as we all know, uniform software processes do not guarantee uniform results. According to current quality assurance literature one must attend to the quality of a process, but one cannot ignore the quality of the result. [Imai 1986]

Here are some ideas for needed definitions. Some of them are attributes or results that one can observe or measure.

ACCIDENT	An unexpected and unwanted release of energy or dangerous substances that leads to an unacceptable [intolerable] loss. One can observe and count accidents. (I substitute intolerable for unacceptable partly from deference to work in progress "across the pond" and partly because I cannot think of an acceptable loss, but I can think of losses that I might tolerate in spite of their being in some sense unacceptable. To me, an accident is like the snake's striking a hiker who steps on it.)
AVAILABILITY	The probability that a system will operate correctly at a specified time. The ratio of system "up" time to total operating time. [IEEE 1983] One can measure availability.
EXPOSURE	The cumulative measure of all the instances and the total duration of JEOPARDY [see JEOPARDY defined below] involving one or more objects and a specific hazard. (This is a suggested definition for a new term. Exposure measures the amount of time all the hikers in a field are likely to spend within striking range of a specific snake.) One can measure or estimate the duration of each instance of jeopardy and calculate this parameter.
FUNCTIONALITY	An attribute that specifies a task or combination of tasks and the operating conditions for a system. One can observe and test functionality.
FITNESS-FOR-USE	A combination of attributes that determines whether system may be acceptable to consumers that require its services. One can measure or observe this attribute in terms of other attributes, such as functionality and safety, that define it and in terms of consumer's opinions.

HAZARD	A condition or combination of conditions within a system that can lead to a MISHAP or an ACCIDENT. (A hazard is like a snake in the grass.) Theoretically one can observe hazards.
JEOPARDY	A situation where a HAZARD will lead to an ACCIDENT. (This is a suggested definition for a new term. The object(s) of potential loss and a mishap must come together. Jeopardy is the situation where a hiker's path crosses a snake in the grass.) One can count instances of jeopardy.
MISHAP	An unplanned event or series of events that <u>can</u> lead to an unacceptable [intolerable] loss, such as death, injury, illness, property loss, or property damage. [Leveson 1991] (Again, I substitute intolerable for unacceptable. A MISHAP is a potential accident, e.g., a hiker steps on a snake. An ACCIDENT is what happens when the snake strikes and injects the hiker with venom.) One can observe a mishap.
RISK	The expected value, expressed in appropriate units, of undesirable outcomes based on the probability of their occurrence over the useful life of a system. (This is a well established calculation; however we have no consensus about the appropriate parameters or units to describe risks to safety. Moreover, we have no consensus about ways to predict probabilities of events that software controls.)
SAFETY	The ability of a system to operate for a specified time under stated conditions without a MISHAP. (This is a suggested definition, similar to the standard definition of reliability.) One can theoretically measure safety defined this way, but we haven't enough information to make safety predictions trustworthy.

Notice the strategies for mitigating risk implicit in these definitions. Eliminate hazards, i.e, exterminate as many snakes as one can find in the field. Try to avoid mishaps, i.e., watch for snakes on the ground ahead to avoid stepping on one. Prevent mishaps from becoming accidents, i.e., wear high, heavy boots. Prepare for accidents, i.e., carry a snake-bite kit and a C-B radio. Reduce exposure, i.e., walk through snake infested fields only when one must and then as fast as one <u>prudently</u> can go. There are many ways to improve system integrity, but plunging blindly, ill-protected and ill-prepared into an unfamiliar field is not one of them.

Good Process: Good Product?

As I mentioned earlier, many potential customers for software standards believe that the quality of software relates poorly to the quality of the development process. These individuals appreciate how a worker's performance affects quality. They may not appreciate how a process affects quality. One tends to forget that people vary in ability and that most groups contain average, above-average, and below-average performers, most of whom are competent. To be credible, a standard should help one develop evidence that good procedures enhance the ability of competent people to produce trustworthy systems.

I believe that we should try to learn whether process standards improve the quality of high integrity software. To do this, we need to establish standard measurements, like SAFETY as defined above, and apply them to current systems. Doing so will establish a baseline for comparison when the standard takes effect. Then we need to assess conforming software according to the standard measurements, compare the results, and see which measurements change from the baseline, if any do. In this way we and our customers can determine objectively which development processes improve the integrity of software.

Bibliography:

[Shore 1985]	The Sachertorte Algorithm and Other Antidotes to Computer Anxiety. Viking Penguin Inc., 1985.
[Paul 1989]	Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation. U.S. Government Printing Office, 1989.
[NRC 1991]	Computers at Risk: Safe Computing in the Information Age. National Academy Press, 1991.
[Leveson 1991]	"Software Safety in Embedded Computer Systems," Communications of the ACM. V34, No. 2 (Feb. 1991), pp 34-46.
[Imai 1986]	Kaizen, the Key to Japan's Competitive Success. McGraw Hill, 1986.
[IEEE 1983]	IEEE Standard Glossary of Software Engineering Terminology. ANSI/IEEE Std 729-1983, IEEE 1983, p. 10.

SOFTWARE CERTIFICATION

Presented at NIST Forum on Standards for
High Integrity Software

June 28, 1991

Janet R. Dunham, Director
Center for Digital Systems Research
Research Triangle Institute
Research Triangle Park, North Carolina 27709
(919) 541-6562

OUTLINE

- View of software certification
- Needs
- Proposed roadmap components

RTI

0691-Jrd-030

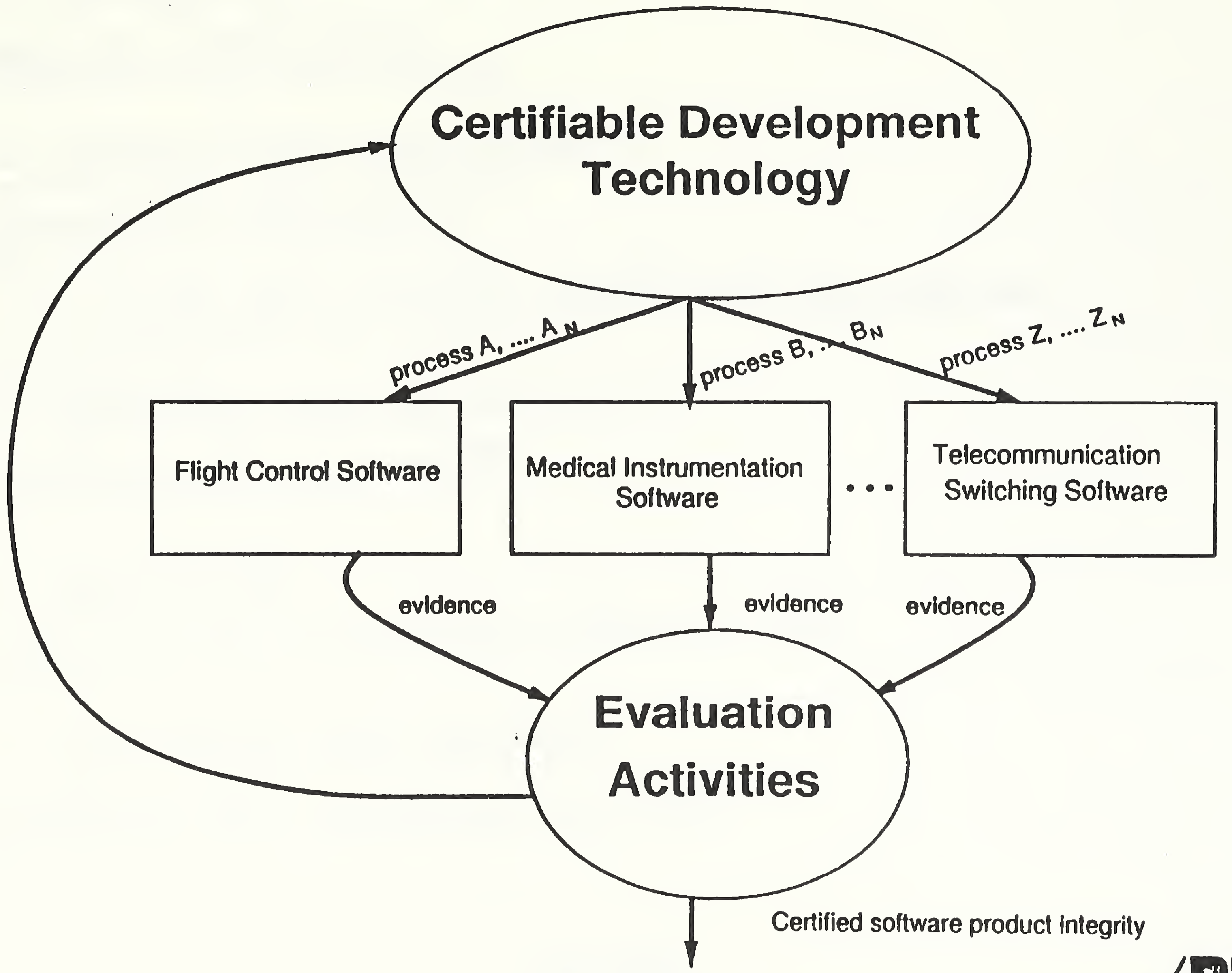
VIEW OF SOFTWARE CERTIFICATION

- Includes all activities conducted to assure product integrity
- Based on a certifiable development technology
- Permits diversity of the development process
- Provides evidence that the implemented process and product are of high integrity
- Imposes standards and guidelines on the development process and the evidence produced

RTI

0691-jrd-030

VIEW OF A SOFTWARE CERTIFICATION PROCESS


Certifiable Development Technology
process A, ... A N
process B, ... BN
process Z, ... ZN
Flight Control Software
Medical Instrumentation Software
Telecommunication Switching Software
evidence
evidence
evidence
Evaluation Activities
Certified software product integrity

RTI

0691-jrd-030

NEEDS

- Indirect-user awareness of risks
  - *Education and training*
- Indirect-user protection against risks
  - *Standards*
- Supporting technology
  - *Research and development*
  - *Packaging of current tools and techniques*
- Developer experience
  - *Qualification and training*
- Certifier experience
  - *Qualification and training*

RTI

0691-Jrd-030

PROPOSED ROADMAP COMPONENTS

- Supporting technology
- Education and training
- Developer and certifier qualitications
- Standards

RTI

0691-Jrd-030

SUPPORTING TECHNOLOGY

ACTIVITY	TARGETS	TOPICS
• Research and Development	• Certifiable Development Technology • Support for Development Process • Requirements on Certification Evidence • Support for Evaluation Process	• Safety-Critical Design Rules • Certifiable Building Blocks • Model Development Process • Specification of Certification Evidence • Model of Evaluation Process • Automation to Support Certification Activities • Certification Credit Guidelines • Certification Metrics • End-Product Testing

RTI

0691-jrd-030

EDUCATION & TRAINING

ACTIVITY	TARGET GROUPS	MECHANISMS
• Education and Training	• End-Users • Developers • Evaluators/Certifiers	• Tutorials • Seminars • Workshops • Videotapes • Multi-Media Demonstrations • Handbooks/Guidebooks • Technical Reports • Case Studies/Demonstration Projects • Consumer Reports

RTI

0691-jrd-030

IEC 730 REQUIREMENTS FOR ELECTRONIC CONTROLS (Draft Standard)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
2.14.20 Inspection	X		X		X		X		X		X		X		X	
2.14.56 Walk-through		X		X		X		X		X		X		X		X
2.14.44 Static analysis	X	X							X	X						
2.14.11 Dynamic analysis			X	X							X	X				
2.14.16 Hardware analysis					X	X							X	X		
2.14.18 Simulation							X	X							X	X
2.14.13 Failure rate calculation	X	X	X	X	X	X	X	X								
2.14.14 EMEA									X	X	X	X	X	X	X	X
2.14.27 Operational test	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Acceptable Hardware Development Strategies

RTI

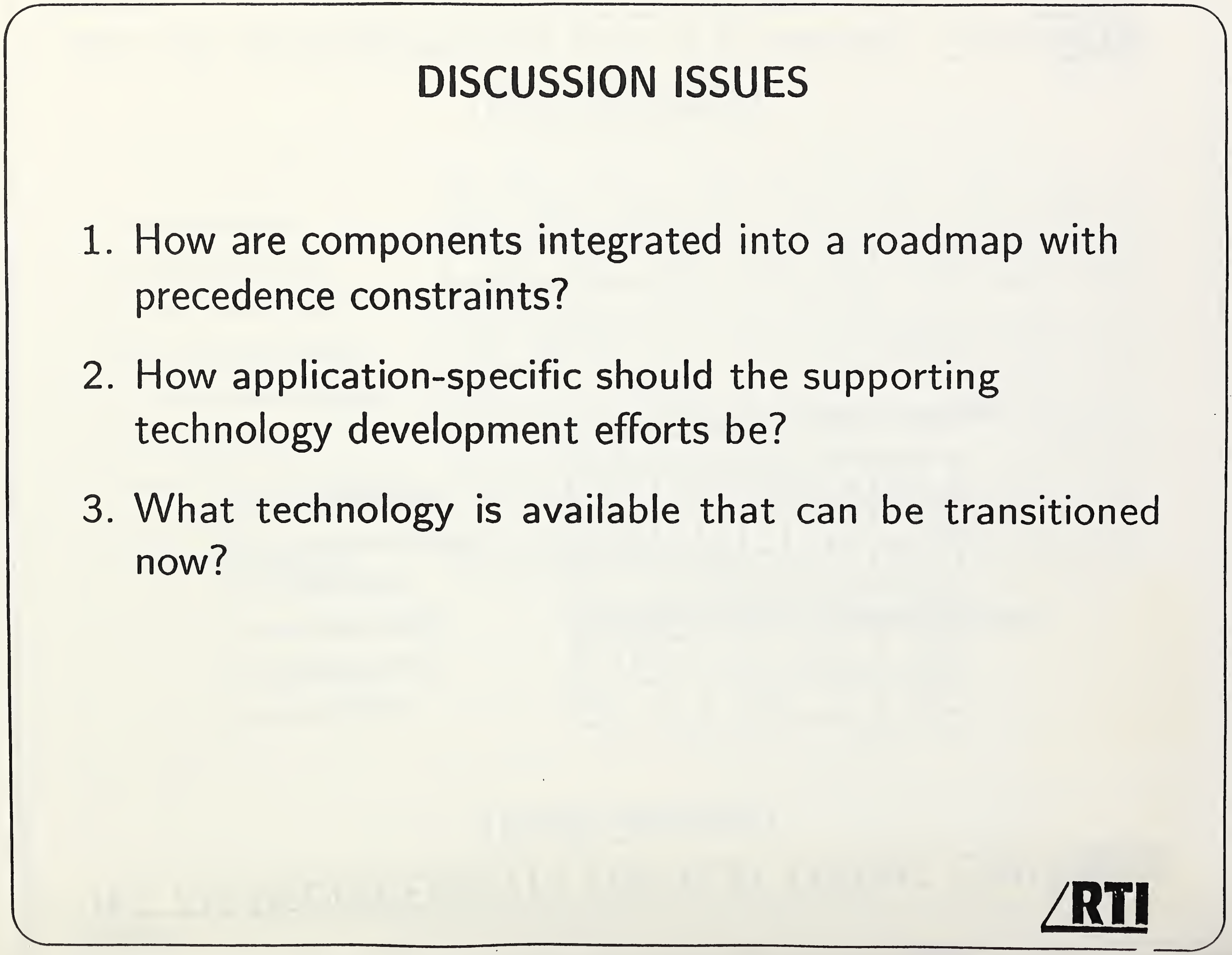
0691-Jrd-030

IEC 730 REQUIREMENTS FOR ELECTRONIC CONTROLS (Draft Standard)

	a	b	c	d	e	f	g	h	k
2.14.20 Inspection	X				X				X
2.14.56 Walk-through		X				X			X
2.14.45 Static analysis			X				X		X
2.14.43 Simulation				X				X	
2.14.48 Systematic tests	X	X	X	X					X
2.14.47 Statistical test					X	X	X	X	

Acceptable Software Development Strategies

0691-jrd-030



DISCUSSION ISSUES

1. How are components integrated into a roadmap with precedence constraints?
2. How application-specific should the supporting technology development efforts be?
3. What technology is available that can be transitioned now?

RTI

0691-jrd-030

NIST-114A
(REV. 3-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4656

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
SEPTEMBER 1991

4. TITLE AND SUBTITLE
Proceedings of the Forum on Standards for High Integrity Software (DOD, Government, Industry)

5. AUTHOR(S)
Dolores R. Wallace, Michael Brown, Archibald McKinlay VI

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)
U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

This report provides information related to the Forum on Standards for High Integrity Software (Department of Defense, Government, and Industry) held at the National Institute of Standards and Technology on June 28, 1991. At the forum, software engineering experts presented their perspectives on the role of software engineers in software safety, a comparsion for safety and computer security issues in standards, hazard analysis, assurance standards, and software certification. Future directions for NIST activities for assurance of high integrity software were proposed.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

assurance; certification; computer security; hazard analysis; software safety; standards

13. AVAILABILITY

[✓ x] UNLIMITED
[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
[✓ x] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
62

15. PRICE
A04

ELECTRONIC FORM