Special Report
CMU/SEI-93-SR-004
ESC-SR-93-004

Dependable Software Technology Exchange

Charles B. Weinstock
Fred B. Scheneider

June 1993

Special Report
CMU/SEI-93-SR-004
ESC-SR-93-004
June 1993

Dependable Software Technology Exchange

Charles B. Weinstock
Software Engineering Institute
Dependable Real-Time Software

Fred B. Scheneider
Cornell University

Unlimited distribution subject to the copyright.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

(signature on file)

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Copyright © 1993 by Carnegie Mellon University.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

Requests for permission to reproduce this document or to prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTIBILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through Research Access, Inc., 800 Vinial Street, Pittsburgh, PA 15212.
Phone: 1-800-685-6510. FAX: (412) 321-2994. RAI also maintains a World Wide Web home page. The URL is http://www.rai.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145. Phone: (703) 274-7633.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Table of Contents

Dependable Software Technology Exchange

Abstract: On March 18 and 19, 1993, the Dependable Real-Time Software project hosted a Dependable Software Technology Exchange. The exchange, sponsored by the Air Force Space and Missile Systems Center and the Office of Naval Research, brought together researchers and system developers, providing an opportunity for the researchers to learn the needs of the developers and for the developers to learn about techniques being investigated by the researchers. This report summarizes what transpired at the meeting.

1 Introduction

Dependability is important in avionics, vehicle control, logistics systems, and command, control, and communications systems, to name but a few. As these systems increasingly rely on computers, software becomes more important to the dependability and safety of their users. Computer hardware has become more reliable, so software is now the bottleneck for achieving dependability. In recent years, however, a number of new approaches have emerged for building software for dependable system applications.

One way to facilitate technology transfer between researchers and practitioners is to get them talking to each other. With this mind, on March 18 and 19, 1993, the Software Engineering Institute's Dependable Real-Time Software project hosted the first in what is hoped to be a series of Dependable Software Technology Exchanges. This initial technology exchange was sponsored by the Air Force Space and Missile Systems Center and the Office of Naval Research.

The purpose of the exchange was to bring researchers and system builders together to provide an opportunity for technology transfer in both directions. System builders are often too busy to attend conferences or to write papers; researchers need exposure to "real world" concerns. The technology exchange provided a forum where each side could educate the other.

The exchange was coordinated by a steering committee consisting of:

- Dr. Charles B. Weinstock, Software Engineering Institute
- Dr. Fred B. Schneider, Cornell University
- Dr. Yitzak Levendel, AT&T Bell Laboratories
- Dr. Ravi Iyer, University of Illinois
- Dr. Walter L. Heimerdinger, Honeywell Systems and Research Center
- Mr. Joseph Chiara, Air Force Space and Missile Systems Center
- Dr. Gul Agha, University of Illinois

The focus was on four topic areas that were deemed to be critical for achieving dependability: formal methods and verification, requirements, operating system support, and object-oriented methods and design. Each area was allocated a half day session divided into three parts. The session started with a one hour presentation from a researcher followed with a one hour long presentation from an experienced practitioner. After a short break, a 75 minute panel discussion was held. Each panel consisted of the two speakers, plus four additional panelists. Audience participation was encouraged throughout the sessions. The technology exchange was videotaped.

By all accounts, the first Dependable Software Technology Exchange was successful. It was well attended—demand almost exceeded capacity. Of perhaps greater importance is the diversity of backgrounds represented by the attendees. The over 70 attendees represented organizations including:

Industry

- AEG/Westinghouse Transportation Systems
- AlliedSignal
- AT&T Bell Laboratories
- Booz-Allen and Hamilton
- Digital Equipment
- Honeywell
- Hughes Aircraft
- IBM Federal Systems
- Lockheed Missile and Space
- Martin Marietta
- MCC
- Mind Tools Corporation
- ObjecTime Limited
- Rockwell International
- Siemens
- SOHAR
- SPARTA, Inc.
- System Technology Development
- Tandem Computers
- Texas Instruments
- Transarc Corporation
- Universal Hi-Tech Development
- Weinberg & Associates

Academia

- Brown University
- Carnegie Mellon University
- Cornell University
- McMaster University
- University of Arizona
- University of California
- University of Illinois
- University of Pittsburgh
- University of Texas
- University of York

Government

- AFOSR
- AF/SMC
- NASA
- NIST
- NRL
- NSWC
- ONR
- SDIO

Other

- Aerospace Corporation

We clearly achieved our goal of bringing researchers and practitioners together.

At the end of the technology exchange, attendees completed a feedback form to help us evaluate the success of the meeting. Although not every respondent found all of the talks useful, every talk was useful to some of the respondents. Virtually every respondent was enthusiastic about holding further technology exchanges, and a variety of topics were suggested.

The next sections describe the four sessions in detail, including the speakers and panelists as well as a summary of the discussion. Following those sections is a synopsis of the comments from participants. A complete list of attendees appears as an appendix.

2 Formal Methods and Verification

The morning session of Thursday, March 18, 1993, focused on formal methods and verification. The technology lecture was given by Dr. John Rushby from SRI International. The title of his lecture was “Formal Methods Technology: What Can it Do? How Best to Use it?” Dr. David Lorge Parnas of McMaster University gave the application lecture, “The Application of Mathematical Methods for Documentation and Inspection of Critical Software.” In addition to these speakers, the panel consisted of Mr. Ricky Butler of NASA Langley Research Center, Mr. John Dehn of IBM Federal Systems, Dr. Bill McKeeman of Digital Equipment, and Dr. Fred Schneider of Cornell University. The session was coordinated by Dr. Schneider.

2.1 Technology Lecture: Dr. John Rushby

Most engineering disciplines have their foundations in applied mathematics. Mathematics is then used as a notation for describing the artifacts being studied and as an analytical tool for predicting the behavior of these artifacts. John Rushby (SRI International), the first speaker in the Formal Methods and Verification session, argued that mathematical logic can play this role for software. He explained that when writing specifications, one is concerned with giving assumptions about the world in which a system will operate, stating requirements concerning what the system is suppose to do, and describing how the system will satisfy its requirements. Logic can be used in all three of these tasks. Moreover, given formal specifications for these aspects of a system, it becomes possible to check for various forms of consistency and completeness, better understand the system by posing challenges, and prove that a design will satisfy the requirements under the given assumptions.

According to Rushby, applying formal methods need not be synonymous with performing complete proofs of correctness. Researchers in formal methods are standardizing on four levels of rigor:

0. No use of formal methods.
1. Using ideas from formal methods, but with an ad hoc notation and proofs based on ad hoc arguments. This is the way most mathematics is done today.
2. Using formal logic for specifications but doing proofs informally.
3. Using formal logic for specifications and employing automated theorem provers to check proofs.

One must select the correct level of rigor for the task at hand. Workers on the two sides of the Atlantic also seem to emphasize different ends of the spectrum. European work is more concerned with levels 1 and 2; work in the U.S. has concentrated on level 3.

Rushby was careful to discuss limitations of formal methods. One cannot entirely formalize a proof, because it is not possible to verify the correctness of a specification for requirements or the accuracy of any formal description for the physical reality that underlies a computing device. Second, level 3 formal methods are expensive, much too expensive to be used on a system of significant size. Still, formal specifications do add value during system design. For documentation, they provide precise, unambiguous descriptions of interfaces and functions. This in turn supports reuse of components. Having formal specifications also provides traceability for design decisions. For analysis, the existence of formal specifications permits early validation of requirements and helps to identify assumptions that must be validated through other means. Finally, formal methods are the only way to verify “negative” properties (i.e., the absence of unintended effects).

In reflecting on the current state of the art, Rushby identified three traditions that now seem to be converging. The first tradition starts with a specification notation, where an expressive notation is chosen without concern for how difficult it would be to construct a supporting theorem prover. The second tradition is motivated by a drive to build powerful theorem provers. The logic of such a theorem prover is then used as the specification notation. A third tradition, now becoming prevalent, seeks to combine the other two by making compromises in the specification language and power of the theorem prover in order to obtain a coherent system.

2.2 Application Lecture: Dr. David L. Parnas

David Parnas (McMaster University) was the second speaker of the session. Parnas has used formal methods in two large mission-critical systems: the software that controls a U.S. fighter plane (the A7) and the safety-critical software for a nuclear power station in Darlington, Ontario. Parnas’ talk outlined experiences with a formal specification notation that he used in the Darlington project. The notation is easily read by engineers, making it possible for the client to understand specifications. Parnas emphasized the importance of this attribute, as it allows the customer to catch errors that software engineers are not qualified to find. And although the notation looks informal, it has precise Predicate Logic semantics. Thus, the specification notation enjoys the same benefits and potential for automated analysis as any specification notation based on a formal logic.

According to Parnas, success in implementing safety-critical software depends on three elements. First, one must be prepared to write precise, organized, mathematical documentation and review it systematically. Second, extensive testing is necessary to allow quick discovery of gross errors and shared oversights. Third (and perhaps the most worrisome) is that qualified people must be involved and an approved process employed. Parnas has found that writing and reviewing rigorous specifications is basically a boring task. Without people having the right skills, it is simply too easy to overlook a flaw. In the Darlington project, non-mathematicians were employed in writing and analyzing the specifications. All the participants required extensive training and supervision in the formal methods being used by the project. Being conversant in the application, however, meant that the participants were effective in reviewing specifications.

The Darlington project and the A7 project both employed level 2 formal methods. Proofs remained informal, and only a limited degree of automated support was available. Parnas argued that formal specifications should be used for documentation and specification now.

Otherwise, we can have no hope of using them in the future as input to rigorous mathematical analysis. The success of the Darlington project certainly argues that level 2 formal methods can have a real payoff.

2.3 Panel Discussion

The next four presentations were from people who had used or tried to use formal methods. Ricky Butler (NASA Langley Research Center) was the first speaker. Butler is trying to prototype a fully verified reliable computing platform for use in flight control. His group has specified and verified redundancy management, task scheduling, and clock synchronization protocols for such a system. They have had experience using many of the currently available tools—Butler reports that it takes about 6 months for him to master a new theorem prover. Still, in contrast to Parnas, Butler believes that wrestling with a mechanical theorem prover is a necessary prerequisite for writing good specifications. Butler has found, however, that the technicalities associated with getting a proof through a mechanical theorem prover are a distraction.

Jon Dehn (IBM Federal Systems Company) next spoke about his experiences in using formal methods for the development of AAS, the next-generation Air Traffic Control System that he is working on. This is a large system—over 1 million lines of source code (Ada), with each installation running on a network of over 200 processors linked by several local area networks. The system must satisfy stringent reliability requirements. Dehn reported that his successes with formal methods were all achieved when small teams had to solve particular aspects of a problem or when the method being employed could be mechanized and applied to an entire system or subset. As a success, Dehn cited the technique being used to manage the application state data. Here, formal analysis allowed potential race-conditions to be detected automatically. Dehn cautioned that formal methods have not worked when appropriate mechanizations did not exist and when the method being employed involved too many questionable assumptions about the system being analyzed.

The third presentation was by Bill McKeeman (DEC) about his experiences moving the LARCH specification language and method from Digital's research laboratory into an industrial software development organization. The plan was to extend LARCH to support the C programming language and then use LARCH to support commercial development activities. Largely for cultural reasons, the desired technology transfer did not occur. The impediments to adopting the system included the following. First, a significant fraction of the user community were excluded from participating by the choice of C as the language being supported. Second, using a system like LARCH is really feasible only for new code, and most software work by the target users was concerned with extending existing code. And finally, the engineers regarded formal specifications as constraining their designs and did not perceive a real pay off in the time horizon of their projects.

The final presentation was by Fred Schneider (Cornell). Schneider argued that formal methods could have leverage today in analyzing small parts of systems and in analyzing simple properties of entire systems. Based on his experience in applying formal methods at IBM and

DEC, he proposed that the position of a “formal methodist” be added to the skills roster of systems projects. Such a person would design special-purpose, lightweight and disposable formal methods for the problem at hand. Being successful, however, would require that a formal methodist be an active participant in the system design—to understand what are the real problems and what approximations of reality are possible. It also would require access to powerful and flexible tools that can be easily integrated.

3 Requirements

The afternoon session of Thursday, March 18, 1993 focused on the problem of specifying requirements for dependable software systems. The technology lecture was given by Prof. John A. McDermid from the University of York. The title of his talk was “Dependability Requirements: Orthodoxy and a Goal-Structured Approach.” Mr. Gerry E. Pasek of Lockheed Missile and Space gave the application lecture, “Dependable Software: A Challenge.” In addition to these speakers, the panel consisted of Dr. Walter L. Heimerdinger of Honeywell Systems and Research Center, Dr. Carl Landwehr of the Naval Research Laboratory, Dr. David Parnas of McMaster University, and Dr. John Rushby of SRI International. The requirements session was coordinated by Dr. Heimerdinger.

3.1 Application Lecture: Prof. John A. McDermid

Requirements analysis is a difficult subject, made all the more so because the customer may think that the requirement means one thing while the developer sees it as something altogether different. The first speaker, John McDermid (University of York), claimed that in an attempt to overcome this, requirements tend to be overly detailed, bordering on implementations. This gives the system designers and implementers little freedom, and hampers the development of efficient systems.

According to McDermid, the usual requirements specification is not much more than a list of function definitions that cover normal functioning, abnormal functioning, dependability properties, performance, quality, and expected changes. These specifications contain too much detail, forcing early design decisions. This hampers flexibility, making it difficult to incorporate down-stream changes. Furthermore, it is hard to check the consistency between requirements so specified.

What is needed, McDermid argued, is a systematic basis for the specification of the fundamental requirements of the system. These are the top-level goals of the stakeholders (e.g., the customer, the users, etc.). The fundamental requirements are then refined into a set of derived requirements that take into account constraints of the real-world (e.g., size limitations, power consumption, etc.). McDermid suggests that, in general, there is still a need to identify conflicts between requirements and to devise strategies that provide acceptable trade-offs. Dependability requirements often expose conflicts.

Representing goals precisely is a difficult problem, according to McDermid. All of the assumptions behind the goals must be enumerated systematically, even hidden assumptions (e.g., the aerodynamics of each wing of an airplane are different). Constraints must also be enumerated, and each goal has to have an “owning” stakeholder so that trade-offs can be evaluated, perhaps by negotiating between stakeholders.

In addition to goals, McDermid asserts that there are three other components to a requirement: there must be a way to tell when a requirement has been met, there must be a justification for the requirement, and finally there must be a strategy for achieving the requirement (e.g., goal sub-structuring).

A conceptual process for requirements definition, then, is to:

- Identify goals—the stakeholders' main aims and desires.
- Model the environment and causal properties.
- Derive more detailed requirements based on real-world constraints.

McDermid was not aware of any single tool that can take the designer through this process. However, tools such as STEPS, ARE, KAOS and TARDIS can form a base.

3.2 Application Lecture: Mr. Gerry E. Pasek

The practitioner's viewpoint was presented by Mr. Gerry Pasek (Lockheed). He suggested that dependable software development is an iterative process that is almost "done in" by requirements analysis. Just when a system starts to satisfy the requirements, as specified, they change, or the customer's expectations change. So the current paradigm is to "code a little, test a little, and document some." Pasek considers 2167A too rigid, and primarily for bureaucrats and lawyers.

According to Pasek, the realities of the procurement process make requirements analysis even more important than it should be, but at the same time the requirements hobble innovation. For instance, the development of the F22 fighter has an integrated 10-year master plan with monthly milestones to be met. This means design decisions are made entirely too early in the development process.

In the normal case, there is little if any user involvement in the requirements specification process. Since developers and users have different viewpoints, this is detrimental to the quality of the final system. Even in testing, developers and users will base their testing on their individual viewpoints of what the system should do. Pasek gave the example of an F22 crash at Edwards AFB that was traced to a pilot misuse of the aircraft, which the software wasn't designed to handle. The software failed, and its error recovery mechanism was faulty. Too much attention is given to testing, and not enough to specifying, documenting, and validating user functionality.

Another problem, Pasek claimed, is the incredible number of requirements typical for a large system. As the number of requirements increases, it gets difficult to deal with interactions. In one system, there were 300 system-level requirements, 40 real requirements, and 45,000 specific requirements. Until recently, it wasn't possible to trace the requirements chain from the specific to the general. In addition to tracability back to the stakeholder/owner of a requirement, the rationale behind each requirement must be documented. This allows the need for a specific requirement to be challenged and defended. Electronic communications is helping here, and database tools are being developed, but tracability remains a major problem.

From a dependability standpoint, too little attention is given to defining what can go wrong. Systems are therefore inherently undependable in their initial phases. Pasek suggested that quality does not imply dependability. Specifying the fault set (especially credible faults) to be accommodated is a major challenge, especially if the hardware has not been defined.

Pasek was a big fan of the SEI level-three process, which he claimed goes a long way toward getting the “right stuff” into products, but he feels we need to concentrate more on defining the project before it starts. In particular, there needs to be a way to capture the thought process leading to requirements.

3.3 Panel Discussion

Walt Heimerdinger (Honeywell) started the panel discussion by talking about requirements that have worked well and those that have not. He divides the former into two classes: process and function. By “process,” he means how the product is built. This covers design constraints, reviews, etc. As examples, he cited DO-178A and MOD 55 (U.K. Ministry of Defense). They are relatively easy to specify and follow, but they may not correlate with the actual behavior of the product. By “function,” he means what the product does. This includes state machines, scenarios, fault-tree analysis, etc. These can help to identify specific faults and hazards and can include specific error-handing procedures.

According to Heimerdinger, requirements that don’t work well include attributes (because it is difficult to cover the entire product), and statistical objectives (because it is hard to get a statistically significant sample for ultradependable systems). Heimerdinger suggests reuse and benchmarks as possible solutions to these problems.

Carl Landwehr (NRL) talked about domains where he has dealt with requirements for dependable software. These included security, safety, and real-time systems. Techniques that have served him well included natural language, cookbooks (e.g., the Yellow Book, MOD 55/56), structured informal techniques (e.g., assertions), semi-formal techniques (e.g., SCR), and formal techniques, both manual and automated. The techniques have proven useful for capturing intended critical properties precisely, facilitating communications between stakeholders, revealing conflicts and errors early in system development, easing the design process, facilitating system assurance, and training users. On the flip side, available techniques don’t work well when dealing with combinations of critical requirements or in supporting the composition/decomposition of requirements onto design entities, especially in the realm of security.

David Parnas (McMaster University) contested at least one of Heimerdinger’s points, stating his belief that some of the process mechanisms hinder the precise specification of requirements. Many aspects of a system are never documented because they never cause trouble.

The final panelist was John Rushby (SRI International), who discussed areas where SRI International had specified dependability properties. These included fault-tolerance for flight control, the “Jet Select” function for the Space Shuttle Orbit DAP, Microprogram correctness, simple real-time properties, and security properties. In general this worked well. Expressive formal methods can be used to specify almost anything in an easily understood way, given talented and skilled users. The crucial need is for transferable methodologies tailored to each application area. In almost all areas, considerable work is needed to create transferable methodologies.

Comments raised by members of the audience, led by Herb Hecht and George Gilley, involved the procurement process. The procurement process is the major problem. You are serving an amorphous, volatile, customer due to frequent personnel changes. Hundreds of people are involved, each with a different set of expectations. The result is a changing set of requirements every time a new person is put in charge. There is only one shot at getting the requirements right, even though iteration will be inevitable. If a requirement is eliminated downstream, some bean-counter is going to want a “give-back”.

4 Operating System Support

The morning session of Friday, March 19, 1993, focused on operating system support for dependable software. The technology lecture was given by Dr. Keith Marzullo of Cornell and the University of California at San Diego. The title of his talk was “Distributed Fault-Tolerant Programming Using Group Programming Tools.” Dr. Doug Jewett of Tandem Computers gave the application lecture, “Tandem Fault Tolerant Systems.” In addition to the speakers, the panel consisted of Dr. Edward Balkovich of Digital Equipment, Mr. Jon Dehn, of IBM Federal Systems, Mr. Craig Hatfield of IBM Federal Systems, and Dr. Alfred Spector of Transarc. This topic area was coordinated by Dr. Ravi Iyer of the University of Illinois and the session was chaired by Dr. Charles B. Weinstock of the Software Engineering Institute.

4.1 Technology Lecture: Dr. Keith Marzullo

Abstractions for implementing fault tolerance are frequently supported in an operating system. Consequently, this session concerned the various abstractions that have been proposed for supporting fault-tolerance and included representative users’ experiences with these abstractions. The first speaker was Keith Marzullo (Cornell/U.C. San Diego). Marzullo summarized the “group programming” approach to distributed computing. This approach, typified by the ISIS system, allows the programmer to assume that events are seen in the same order by all members of a distributed group of processes. Having the illusion of a total order on distributed events simplifies programming. Solutions to the election problem illustrated simplifications that become possible when primitives to support group programming are available.

Marzullo explained that systems to support group programming usually implement certain key abstractions. At the lower levels are protocols to implement the causal delivery of a message to the members of a process group, protocols to detect failures of members, and protocols to maintain a consistent view of a group’s membership. High-level abstractions found usually include ordered multicasts, state transfer protocols so that new members can be added to a group, coordinator-cohort and other group coordination protocols, and support for logging and recovery.

As a non-trivial example of group programming, Marzullo described his RNFS replicated NFS file system. This system provides to clients the same interface as the SUN NFS network file system. However, in RNFS, files can be replicated on multiple file servers. Different replicas are kept consistent by RNFS. Marzullo also described ongoing projects that are exploring group programming support. Although the list is dominated by university research efforts, the approach has been employed, for example, by IBM in two product efforts (AAS air traffic control and TSAF VM/370 support).

4.2 Application Lecture: Dr. Doug Jewett

Doug Jewett (Tandem Computers Inc.), the second speaker in this session, surveyed Tandem’s efforts to provide fault-tolerant computing. Their first and best known offering was built in the mid 1970’s using a custom operating system running on top of custom hardware. The target customer application was online transaction processing. Tandem’s system is based on using “process-pairs” for system processes. Each process pair has a primary and backup. The primaries write periodic state checkpoints, which allows a backup process to take over when a primary process fails. Unfortunately, Tandem has found that designing process-pairs is difficult. Moreover, the mechanism is not immune to correlated failures arising from programmer errors.

The difficulty in building process-pairs led Tandem to implement operating system support for transactions. Again, a primary and backup are employed, but now the usual transaction semantics simplifies the checkpointing problem. By writing transactions rather than process-pairs, users of Tandem systems are able to implement fault-tolerance in their applications. The current system, then, is a hybrid: critical software (including a transaction manager) is written as process-pairs and gives continuous service across failures; most user software is written in terms of transactions and, therefore, is restarted in response to a failure.

Tandem is also designing a fault-tolerant UNIX system. During his lecture, Jewett briefly outlined this system as well. The goal is to support existing UNIX applications program interfaces and be able to survive hardware failures. A variety of design changes are being used to harden the kernel, as UNIX was not originally intended for this domain. Jewett also discussed some of the difficulties encountered in debugging such a system.

4.3 Panel Discussion

The next four presentations discussed systems efforts to support fault-tolerance. Edward Balkovich (DEC) started by describing three representative system architectures for implementing different degrees of fault-tolerance. From these, Balkovich then concluded that managing redundancy in storage, processing, and the communications network must be integral to the design of an operating system. The use of journaling (i.e., logging on stable storage) was found to be quite useful for implementing critical services and is frequently used in DEC systems applications. Balkovich also pointed out that fault-tolerance support could be employed to allow system upgrades without shutting down a system.

The next presentation was by Craig Hatfield (IBM Federal Systems Company) concerning an operating system that IBM is building for a defense application. Among the system’s capabilities are replication of critical files and restart without reloading. The system runs on multiple communicating processors and has a master and slaves. A slave can take over for the master, if that master fails. Echoing Tandem’s experiences, Hatfield found that writing applications programs was complicated by the need to take checkpoints for failure recovery. Hatfield also reported that coping with power failures was particularly difficult, because some hardware interfaces had to be reinitialized following power outages.

Alfred Spector (Transarc) then reported on the use of transactions as a way of providing fault-tolerance. Spector discussed elements of current OSF DCE and Transarc's Encina systems in order to illustrate the practicality and acceptance of this approach. The use and implementation of transactions is well understood; the hardest part at this point is getting all the details right so that an implementation correctly deals with system partitions and heterogeneous processors. Spector did identify some outstanding research issues. Work needs to be done in understanding replicated datatypes and in automated support for application development. Spector also called for simplified administration of replicated systems.

The final speaker of the session was Jon Dehn (IBM Federal Systems Company). Dehn briefly outlined how group programming is employed in IBM's AAS system. Application programmers do not see a group programming interface. Rather, group programming is used to implement a primary-backup style of fault-tolerance. The replication that group programming would entail for the application levels of the system was deemed to be too expensive to be practical.

5 Object-Oriented Programming and Design

The afternoon session of Friday, March 19, 1993, focused on object-oriented programming and design as a means of achieving dependable software. The technology lecture was given by Dr. Gul Agha of the University of Illinois. The title of his lecture was “Object-Oriented Dependable Computing.” Dr. T. L. Wang of AT&T Bell Laboratories gave the application lecture, “Object Oriented Design of a Highly Available Switching System.” In addition to these speakers, the panel consisted of Dr. Jacob Abraham of the University of Texas at Austin, Dr. Jim Coplien of AT&T Bell Laboratories, Mr. Bran Selic of ObjecTime, and Dr. Peter Wegner of Brown University. The object-oriented programming and design topic was coordinated by Dr. Yitzak Levendel of AT&T Bell Laboratories and Dr. Gul Agha of the University of Illinois.

5.1 Technology Lecture: Dr. Gul Agha

Dr. Agha’s (University of Illinois) talk began with a short tutorial on object-oriented programming. He discussed procedure abstraction, data encapsulation, data abstraction, and the idea of object inheritance. The key use of abstractions is to separate how the code is implemented from its functional specification. It is also common to want to separate the time that something is done from the actual task. For instance, asynchronous communication allows computation to proceed in parallel with that communication. This leads to the need for synchronization.

According to Agha, the advantage of object orientation are in data abstraction, modeling power, and reuse. Abstractions are provided by encapsulation and hidden representation. Data abstraction’s modeling power comes from the hierarchical organization and through specialization (achieved through inheritance). Reuse is achieved through incremental refinement of software components.

Agha then went on to describe a methodology for dependability. The four pieces of this methodology include modularity, reusability, flexibility, and composability. He introduced the actor model. In this model, the universe contains computational agents called actors. Each actor has a mail address, a behavior, and a local state. Behaviors can be dynamically replaced and new actors can be created. Actors communicate using a point-to-point, asynchronous, buffered protocol. Mail addresses can be communicated, which allows for dynamic topology. Arrival order and activation order form the fundamental synchronization mechanisms.

Agha believes that three aspects of the behavior of an actor are necessary and sufficient for most dependability protocols. These are the structure of its mail queue, the dispatcher that handles message sends, and the state of the actor. Each of these components is a separate object, independent of the application. He then went on to describe the Broadway Kernel and the HAL language.

As an example of a specific dependability mechanism, Agha showed how actors can be replicated. The changes are relatively minor. The dispatcher in new copies of the actor is modified to send all messages to the original actor’s dispatcher. The original actor’s mail queue is modified to broadcast to all of the replicated copies, and the behavior of the original actor's dispatcher is modified to collect messages from the copies, vote the results, and pass the resulting message to other actors.

5.2 Application Lecture: Dr. T. L. Wang

Dr. Wang (AT&T) discussed applying object oriented methods to the design of a highly available switching system. Switching systems are real-time systems where continuous operation is of paramount importance. Some errors (e.g., dropped or misrouted calls) are tolerable, as the customer can retry or redial. But once a call is completed, the goal is that the call be maintained. The switch has a three minutes per year maximum downtime requirement. Over time, they have observed that outages are caused by hardware failures 20% of the time, software failures 40% of the time, and procedure failures 40% of the time. The design goal of high availability means that repairs must be made quickly when there is a failure. A major research issue is software repair.

As Wang described it, the logical hardware architecture of the switch is functionally distributed across several processor families. These include call processing, operation, accounting and maintenance, interconnect, and peripheral control. Each of these families is its own environment with its own availability strategy. Fault tolerance is provided by having spares for each family. They are considering a heterogeneous environment (different architectures) to avoid common mode failures.

The software architecture is component oriented. All of the components are based on the same basic framework. Implementations of specific components use inheritance to help implement the specific functionality. The architecture supports flexible distribution of software components. Wang suggested that this allows for load balancing and for dynamically relocating software modules to minimize the size of repair groups. The architecture is made of both static components (e.g., those representing the switch itself) and dynamic objects (e.g., those representing a particular call).

The design makes heavy use of an object model with complete data and text encapsulation. No single component knows what any other component is doing. Multiple inheritance provides a way to resolve name conflicts and priorities. Individual designers use incremental redefinition of classes and methods. According to Wang, this provides a modular structure for the system architecture, provides for localization of recovery and repair actions, and coordinates hardware and software initializations. A virtual machine memory model provides safe access to program and data objects.

The error checking aspects of the system described by Wang include run-time type checking, application signaled errors, and customized stack recovery actions. This simplifies exception handler design and recovery actions, and reduces the need for audits during execution.

Wang closed with a summary of the key attributes of the system that enhance availability, which include repairable interfaces, flexible software configuration, update on-the-fly, elimination of memory errors, graceful error recovery, run-time error detection, and standardized maintenance frameworks.

5.3 Panel Discussion

The panel discussion started with a brief talk by Jacob Abraham (University of Texas). He has used object-based techniques in application software development and as a basis for increasing system dependability. In particular, it was used for development of CAD tools for VLSI test and verification as well as for tools to design and evaluate fault-tolerant systems. This includes a fault injection system written in C++.

Object-oriented techniques have, in general, worked well for Abraham. During the development of prototypes, the ability to change algorithms, modify data structures, etc., in parts of the tool has allowed easy exploration of alternatives. It has also made it easy to integrate tools written by different programmers. However, there is a need for an object-based language that can capture specifications of a program or design. Also needed is support for designing systems seamlessly at the software and hardware levels.

James Coplien (AT&T) talked about “Objects, Multiple Paradigm Integration and Organization Structure.” His thesis is that complexity is the root of many software woes and that abstraction is one way to attack complexity. Object-oriented techniques are one tool for dealing with abstractions, but they aren’t suitable for every domain. A good abstraction encapsulates change. Object partitioning doesn’t always accomplish this and may lead to unsuitable coupling. Objects are bad at encapsulating change in real-time embedded systems.

Bran Selic (ObjecTime) talked about work he has done in the telecommunications area for Northern Telecom. The system he is involved with has over 20 million lines of code, making individual methods all but invisible. In this system, requirements are continually being added, the level of distribution is changing, the structure is changing, and it is mostly soft real-time.

In this environment the object model has resulted in a significant productivity improvement (measured at 5:1). This is due primarily to a more appropriate programming paradigm, reuse, and an exceptional development environment (Smalltalk-80). The object model is inherently architectural. The basic programming model is a network of interacting components, which is a natural model for distributed real-time systems. According to Selic, the built-in abstractions mechanisms encourage specification of evolvable systems.

However, according to Selic, all is not perfect with the architectural model. Object-oriented concepts are unnecessarily restricted to the programming language level, and programming languages are not conducive to architectural modeling. Consequently one cannot exploit useful object-oriented concepts at the architectural level where the payback would be the greatest.

Peter Wegner's talk concluded the object oriented panel. He presented a list of research issues for object-oriented programming and design, including how to use Ada in this environment, interface-oriented programming, type safety versus incremental class flexibility, languages for talking about interconnection and communication, reflective architectures, constraints, concurrency and synchronization of objects, persistence, and heterogeneous objects.

In the ensuing discussion, one of the key themes was how object-oriented programming and design applies to dependability. One response was that it helps to constrain the design by providing rules for interactions between objects. However, Jim Coplien claimed that fault tolerance needs a much deeper understanding of the world than the basic system design. Recovery is not a part of the objects, but rather it is in the backplane of the system. Dr. Wang then said that the object paradigm is a starting point, not a complete solution; it's just easier to start with objects.

6 Participants’ Comments

All of the participants attending the Technology Exchange were asked for written comments on the meeting. Over 25% of the attendees provided this feedback. This section summarizes those comments.

Participants completed a form that asked what their expectations about the exchange were and how well these expectations were met. For the most part, the participants’ expectations were met:

- The meeting allowed participants to find out about potentially useful state-of-the-art work in dependable software.
- The discussions helped set an agenda of technology issues that are ripe for further exploration.
- Participants had a chance to meet new colleagues and compare experiences.

In general, participants found the selected topics to be appropriate (average 4 out of 5), the technical depth of talks was almost right (average 3 out of 5), and the format was useful (average 4+ out of 5).

When asked which talk they liked most and which they liked least, the responses were mixed. Most of those responding found the formal methods and verification session to be outstanding. But, good and bad things were said about all of the sessions:

> “Gul Agha’s talk, although more abstract, was full of excellent ‘future’ practical implementation suggestions. Some moments in the panel discussion were very informative.”
>
> “Gul Agha’s presentation was not well thought out and was not appropriate for the audience.”
>
> “The formal methods and verification topics were useful—these are important areas for our company.”
>
> “The formal methods section was interesting but provided little that I can implement in a tight schedule, minimum budget environment.”

There was some unhappiness with the industry lectures. As one respondent put it, “The industry presenters were not that helpful—either their material was not very thoughtful or it was description of large, untractable (sic), definitely not safety-critical systems.” Yet, at the same time, the respondents felt that “the idea of having different types of representation, to get a good mix of practitioners and technology makers” was worthwhile.

Nearly all of the respondents would like to see additional exchanges (5- average out of 5). Topics that they suggested include:

- Metrics
- Analysis of results on real life cases.

- Implementation issues on real hardware.
- Design of small, real-time kernels for safety critical systems.
- Testing of safety critical systems.
- More on formal methods.
- More on requirements (but more focused).
- Reliability.

Most participants would attend additional exchanges and would like to see one held within a year. One respondent suggested the next exchange be in 4 months. Six months was the typical time frame suggested.

A mailing list “exploder” depend-sw@sei.cmu.edu has been created to reach the people who attended the exchange.

Appendix A List of Participants

Jacob Abraham
University of Texas in Austin
Computer Engineering Research Center
2201 Donley Drive
Suite 395
Austin, TX 78758
(512) 471-8983
jaa@cerc.utexas.edu
FAX: (512) 471-8967

Leonor Abraido-Fandino
Siemens Corporate Research
(609) 734-3387

Gul Agha
University of Illinois
Department of Computer Science
1304 W. Springfield Avenue
Urbana, IL 61801

Ed Balkovich
Digital Equipment Corporation
Cambridge Research Lab
One Kendall Square
Cambridge, MA 02139
(617) 621-6630
eeb@crl.dec.com
FAX: (617) 621-6650

Mario R. Barbacci
Program Manager
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7704
mrb@sei.cmu.edu
FAX: (412) 268-5758

Steve Bravy
SDIO/GMN
19142 Roman Way
Gaithersburg, MD 20879
(703) 693-1595
FAX: (703) 693-1700

Mark Breland
Project Leader
MCC
3500 West Balcones Center Drive
Austin, TX 78759-6509
(512) 338-3509
breland@mcc.com
FAX: (512) 338-3900

Donald Brown
AT&T
263 Shuman Blvd
Room 1A143
Naperville, IL 60566
(708) 713-5050
FAX: (708) 713-5398

Rickey Butler
Fault-Tolerant Systems Research Engineer
NASA
Langley Research Center
Mail Stop 130
Hampton, VA 23681-0001
(804) 864-6198
rwg@air16.larc.nasa.gov
FAX: (804) 864-4234

Stephen Cha
Member of Technical Staff
The Aerospace Corporation
Trusted Computer Systems Department
(M1/055)
2350 E El Segundo Blvd.
El Segundo, CA 90245-4691
(310) 336-7977
cha@aero.org
FAX: (310) 336-5833

Joe Chiara
U.S. Air Force Space & Missiles Systems
Center
Air Force Space Center
P.O. Box 92960
Los Angeles, CA 90009-2960
(310) 363-3521
chiara@cnb.laafb.af.mil
FAX: (310) 363-0265

James Coplien
AT&T Bell Laboratories
1000 East Warrenville Road
P.O. Box 3013, Room IHC-1G341
Naperville, IL 60566-7013
(708) 713-5384
cope@research.att.com
FAX: (708) 713-4982

Mark Cornwell
President
Mind Tools Corporation
2 Davis Drive
P.O. Box 12076
Research Triangle Park, NC 27709
(919) 990-9105
cornwell@rock.concert.net
FAX: (919) 990-8561

Jon Dehn
IBM Corporation
Systems Intergration Division
9201 Corporate Blvd.
Rockville, MD 20850
(301) 640-2912
dehn@ibm.vmt.com
FAX: (301) 640-3103

Jorge Diaz-Herrera
Member of Technical Staff
Software Engineering Institute
Products and Services Division
Rm 4120
Pittsburgh, PA 15213-3890
(412) 268-7636
jldh@sei.cmu.edu
FAX: (412) 268-5758

Gary Falacara
SPARTA, Inc.
30100 Town Center Drive
#438
Laguna Niguel, CA 92677
(714) 363-9101
gnf@orion.oac.uci.edu

D. Helen Gill
Principal Scientist
MITRE Corporation
1820 Dolly Madison Boulevard
McLean, VA 22102
(703) 883-7980

George Gilley
The Aerospace Corporation
2350 East El Segundo Blvd.
El Segundo, CA 90045
(310) 336-1552
gilley@aero.org
FAX: (310) 336-8266

Peter Goddard
Section Head
Hughes Aircraft Company
Ground Systems Group
P.O. Box 3310 618/W308
Fullerton, CA 92634
(714) 732-7754
FAX: (714) 732-2613

B.K. Gogia
Vice President
Datamat Systems Research, Inc.
13955 South Spring Drive
Clifton, VA 22024-2453
(703) 222-5996
FAX: (703) 222-5996

Craig Hatfield
IBM Federal System Corporation
9500 Godwin Drive
Manassas, VA 22110
(703) 367-5879
FAX: (703) 367-4259

Herbert Hecht
President
SOHAR
8421 Wilshire Blvd.
Suite 201
Beverly Hills, CA 90211-3204
(213) 653-4717
sohar!herb@cs.ucla.edu
FAX: (213) 653-3624

Walter Heimerdinger
Honeywell Systems & Research Center
3660 Technology Drive
MN 65-2100
Minneapolis, MN 55418-1006
(612) 951-7332
walt@src.honeywell.com
FAX: (612) 951-7438

Raymond C. Hoppes
Advisory Engineer
IBM Corporation
Federal Systems Company
P.O. Box 9023
003B
Boulder, CO 80301
(303) 924-9752
hoppes@blofum9.ibm.com

Frank Houston
Weinberg & Associates
1440 Conchester Highway
Boothwyn, PA 19061
(215) 459-2732 x625
FAX: (215) 459-8381

Chuck Howell
MITRE Corporation
Software Engineering Center, J80
7525 Colshire Drive
MS W197
McLean, VA 22102-3481
(703) 883-6080
howell@mitre.org
FAX: (703) 883-1339

Michelle Hugue, Ph.D.
Member of Technical Staff
Allied-Signal
9140 Old Annapolis Road
Columbia, MD 21045-1998
(410) 964-4158
michelle@batc.allied.com
FAX: (410) 992-5813

Bruno Jambor
Senior Staff Engineer
Martin Marietta
P.O. Box 179
MS:T320
Denver, CO 80201
(303) 977-1972
bjambor@hellcat.den.mmc.com
FAX: (303) 977-1145

Doug Jewett
Tandem Computers
14321
Tandem Blvd.
Austin, TX 78728
(512) 244-8273
dej@mpd.tandem.com
FAX: (512) 244-8588

Mary Jones
Universal Hi-Tech Development Inc.
20 West Gude Drive
Rockville, MD 20850
(301) 340-8899
FAX: (301) 217-0131

John C. Kelly
Group Leader
Jet Propulsion Laboratory
Software Product Assurance
4800 Oak Grove Drive
MS 125-233
Pasenda, CA 91109
(818) 354-4495
jckelly@spa1.jpl.nasa.gov
FAX: (818) 393-6682

Nick Klavin
IBM Corporation
9221 Corporate Blvd.
Rockville, MD 20850
(301) 640-4135
FAX: (301) 640-4010

Gary Koob
Chief of Naval Research
United States Navy
Code 1133
800 N. Quincy Street
Ballston Tower One
Arlington, VA 22217-5660
(703) 696-0872
koob@itd.nrl.navy.mil
FAX: (703) 696-0934

Carl Landwehr
Naval Research Laboratory
United States Navy
Code 5542
Washington, DC 20375-5337
(202) 767-3381
landwehr@itd.nrl.navy.mil
FAX: (202) 404-7942

Yitzak Levendel
AT&T Bell Laboratories
263 Shuman Blvd
P.O. Box 3050, Room 2S-202
Naperville, IL 60566-7050
(708) 979-1310
levendel@att.com

Randall Lichota
Hughes ESC/AVS
Building 1704
Room 107
Hanscom AFB, MA 01731-5000
(617) 377-2520
lichotar%emis2.decnet@v5.hanscom.af.mil

David Littman
Asst. Professor of Computer Science
George Mason University
Department of Computer Science
Fairfax, VA 22030-4444
(703) 993-1545

Keith Marzullo
University of California, San Diego
Department of Computer Science and Engineering
9500 Gilman Drive
La Jolla, CA 92093-0114
(619) 534-3729
marzullo@cs.ucsd.edu
FAX: (619) 534-4428

John McDermid
Professor of Software Engineering
University of York
University of York
Department of Computer Science
York, Y01 5DD
ENGLAND
(449) 044-3272 6
jam@uk.ac.york.minster
FAX: (490) 443-2708

W.M. McKeeman
Digital Equipment Corporation
100 Spit Brook Road
ZK02-N30
Nashua, NH 03062
mckeeman@tle.enet.dec.com

Steven Miller
Software Verification Specialist
Rockwell International
Collins Commercial Avionics Engineering
400 Collins Road NE
MS 124-211
Cedar Rapids, IA 52498
(319) 395-8008
spm@hwking.cca.cr.rockwell.com
FAX: (319) 395-4068

Daniel Mosse
University of Pittsburgh
Department of Computer Science
Pittsburgh, PA 15260
(412) 624-8923
mosse@cs.pitt.edu
FAX: (412) 624-8854

Arup Mukherjee
Carnegie Mellon University
School of Computer Science
5000 Forbes Avenue
Pittsburgh, PA 15213
(412) 268-3047
arup@cmu.edu

Tony Ng
IBM Corporation
Federal Systems Company
9221 Corporate Blvd.
Rockville, MD 20850
(301) 640-4025
ngt@wmavm7.vnet.ibm.com
FAX: (301) 640-4010

David Parnas
Professor
McMaster University
Communications Research Laboratory
McMaster University
Hamilton, Ontario
CANADA L8S 4K1
(416) 525-9140 x7353
parnas@triose.eng.mcmaster.ca
FAX: (416) 525-9140

Jerry Pasek
Lockeed Missile & Space
1111 Lockheed Way
Org:6930, Building 592
Sunnyvale, CA 94089-3504
(408) 756-5955

Ragunathan Rajkumar
Member of the Technical Staff
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213
(412) 268-8707
rr@sei.cmu.edu
FAX: (412) 268-5758

Chuck Roark
Senior Member of the Technical Staff
Texas Instruments
Defense Systems & Electronics Group
P.O. Box 869305
M/S 8435
Plano, TX 75086
(214) 575-3537
roark@skvax1.ti.com

Arthur Robinson
President
System Technology Development
Corporation
1035 Sterling Road
Suite 101
Herndon, VA 22070
(703) 478-0687
FAX: (703) 478-0689

Michael Rodbell
Booz-Allen & Hamilton, Inc.
891 Elkridge Landing Road
Linthicum, MD 21090
(410) 684-6249
rodbellm@asq8.ads.com
FAX: (410) 684-6475

Steven Rogers
MCC
3500 West Balcones Center Drive
Austin, TX 78759-6509
(512) 338-3691
srogers@mcc.com

John Rushby
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
(415) 859-5456
rushby@csi.sri.com

Richard Scalzo
Naval Surface Warfare Center
10901 New Hampshire Avenue
Silver Spring, MD 20903
(301) 394-2926
rscalzo@nswc-wo.nswc.navy.mil
FAX: (301) 394-1164

Richard Schlichting
Associate Professor
University of Arizona
Department of Computer Science
Tucson, AZ 85721
(602) 621-4324
rick@cs.arizona.edu
FAX: (602) 621-4246

Fred Schneider
Professor
Cornell University
Department of Computer Science
4130 Upson Hall
Ithica, NY 14853
(607) 255-9221
fbs@cs.cornell.edu

Zary Segall
Carnegie Mellon University
Department of Elec. & Computer
Engineering
5000 Forbes Avenue
Pittsburgh, PA 15213
(412) 268-3736
2sdcs.cmu.edu
FAX: (412) 268-3890

Bran Selic
Vice President R & D
ObjecTime Limited
340 March Road
Kanata
ONTARIO, CANADA
(613) 591-3435
bran@objectime.on.ca
FAX: (613) 591-3784

Daniel P. Siewiorek
Professor
Carnegie Mellon University
SCS and ECE
Wean Hall
Pittsburgh, PA 15213-3890
(412) 268-2570
arpanet:dps@a.gp.cs.cmu.edu
FAX: (412) 681-5739

Gunnar Skogsholm
AEG Westinghouse Transportation Systems
1501 Lebanon Church Road
Pittsburgh, PA 15236-1491
(412) 655-5824

James G. Smith
Program Manager Information Systems
Office of Naval Research
Code 1267
800 N. Quincy Street
Arlington, VA 22217-5000
(202) 696-4715
jgsmith@ltd.nrl.navy.mil
FAX: (202) 696-0308

Thomas Smith
Lead Scientist
MITRE Corporation
Navy & Info Systems Division
7525 Colshire Drive
MS 2645
McLean, VA 22102
(703) 883-7992
TSmith@mitre.ARPA

Alfred Z. Spector
Transarc Corporation
The Gulf Tower
707 Grant Street
Pittsburgh, PA 15219

Jay K. Strosnider
Asst. Professor - ECE
Carnegie Mellon University
ECE
Schenley Park
Pittsburgh, PA 15213
(412) 268-6927
jks@gauss.ece.cmu.edu

Steven Suddarth
United States Air Force
AFOSR-Boling AFB
110 Duncan Avenue
Suite B115
Washington, DC 20332-0001
(202) 767-4939

Neeraj Suri
Allied-Signal
9140 Old Annapolis Road
Columbia, MD 21045-1998
suri@batc.allied.com

Joseph E. Tatem
Member of Technical Staff
Texas Instruments
Central Research Labs
3825 Furneaux Lane
Carrollton, TX 75007
(214) 995-0385
tatem@csc.ti.com
FAX: (214) 995-0304

Dolores R. Wallace
Computer Scientist
National Institute of Standards and Technology
Division 872
Building 225/Room B266
Gaithersburg, MD 20899
(301) 975-3340
wallace@swe.ncsl.nist.gov
FAX: (301) 590-0932

Chris Walter
Allied-Signal
9140 Old Annapolis Road
Columbia, MD 21045-1998
chris@batc.allied.com

T.L. Wang
AT&T Bell Laboratories
363 Schuman Blvd.
Room 1U102
Naperville, IL 60566
(708) 713-4004
attma-hiexist!tlw
FAX: (708) 713-5398

Peter Wegner
Professor
Brown University
Department of Computer Science
Box 1910
Providence, RI 02912
(401) 734-3387
pw@cs.brown.edu
FAX: (401) 863-7657

Charles B. Weinstock
Senior Member of the Technical Staff
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7719
weinstock@sei.cmu.edu
FAX: (412) 268-5758

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS
None

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for Public Release
Distribution Unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
CMU/SEI-93-SR-04

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Software Engineering Institute

6b. OFFICE SYMBOL (if applicable)
SEI

7a. NAME OF MONITORING ORGANIZATION
SEI Joint Program Office

6c. ADDRESS (city, state, and zip code)
Carnegie Mellon University
Pittsburgh PA 15213

7b. ADDRESS (city, state, and zip code)
HQ ESC/ENS
5 Eglin Street
Hanscom AFB, MA 01731-2116

8a. NAME OFFUNDING/SPONSORING ORGANIZATION
SEI Joint Program Office

8b. OFFICE SYMBOL (if applicable)
ESC/ENS

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F1962890C0003

8c. ADDRESS (city, state, and zip code))
Carnegie Mellon University
Pittsburgh PA 15213

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO	PROJECT NO.	TASK NO	WORK UNIT NO.
63756E	N/A	N/A	N/A

11. TITLE (Include Security Classification)
Dependable Software Technology Exchange

12. PERSONAL AUTHOR(S)
Charles B. Weinstock and Fred B. Scheneider

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM TO

14. DATE OF REPORT (year, month, day)
June 1993

15. PAGE COUNT
30 pp.

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
dependable real-time software
Dependable Software Technology Exchange

19. ABSTRACT (continue on reverse if necessary and identify by block number)

On March 18 and 19, 1993, the Dependable Real-Time Software Project hosted a Dependable Software Technology Exchange. The exchange, sponsored by the Air Force Space and Missile Systems Center and the Office of Naval Research, brought together researchers and system developers, providing an opportunity for the researchers to learn the needs of the developers and for the developers to learn about techniques being investigated by the researchers. This report summarizes what transpired at the meeting.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED ■ SAME AS RPT □ DTIC USERS ■

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified, Unlimited Distribution

22a. NAME OF RESPONSIBLE INDIVIDUAL
Thomas R. Miller, Lt Col, USAF

22b. TELEPHONE NUMBER (include area code)
(412) 268-7631

22c. OFFICE SYMBOL
ESC/ENS (SEI)

ABSTRACT — continued from page one, block 19