

# Proceedings of the Real-Time Systems Engineering Workshop

B. Craig Meyers

Peter H. Feiler

Ted Marz

*August 2001*

SPECIAL REPORT  
CMU/SEI-2001-SR-022





CarnegieMellon  
**Software Engineering Institute**

---

Pittsburgh, PA 15213-3890

# Proceedings of the Real-Time Systems Engineering Workshop

CMU/SEI-2001-SR-022

B Craig Meyers  
Peter H. Feiler  
Ted Marz

*August 2001*

**Dynamic Systems Program**

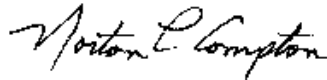
Unlimited distribution subject to the copyright.

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



Norton L. Compton, Lt Col, USAF  
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright © 2001 by Carnegie Mellon University.

Requests for permission to reproduce this document or to prepare derivative works of this document 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 MERCHANTABILITY, 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 F19628-00-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.

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

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (<http://www.sei.cmu.edu/publications/pubweb.html>).

---

# Table of Contents

<b>Abstract</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Presentations	1
1.2 Working Session	2
1.3 Workshop Materials	2
<b>2 Foundation Issues</b>	<b>3</b>
<b>3 Management Considerations</b>	<b>5</b>
3.1 Acquisition	5
3.2 Education	6
<b>4 Technical Considerations</b>	<b>9</b>
4.1 Characteristic-Specific Issues	9
4.1.1 Predictability	9
4.1.2 Fault Tolerance	11
4.1.3 Interoperability	12
4.2 Additional Considerations	14
4.2.1 Standards	14
4.2.2 COTS Products	15
4.2.3 Modeling	15
4.2.4 Verification and Validation	16
4.2.5 Evolution	16
<b>5 Community Considerations</b>	<b>19</b>
5.1 Knowledge Sharing	19
5.2 Toward a Practice of Real-Time Systems Engineering	20
5.3 Consortium	20
<b>6 Summary</b>	<b>23</b>
<b>Appendix: Attendees</b>	<b>25</b>



---

# Abstract

This report presents the results of a workshop on the topic of real-time systems engineering. The workshop was held as part of the SEI Symposium in Washington, DC, during September 2000. The objective of the workshop was to identify key issues and obtain feedback from attendees concerning real-time systems engineering. Two broad keynote presentations were presented, followed by topical presentations concerning predictability, fault tolerance, and interoperability. This report summarizes the workshop in terms of foundation, management, and technical topics, and it contains a discussion related to developing a community of interest for real-time systems engineering.





---

# 1 Introduction

The Software Engineering Institute (SEI) has initiated a project to improve the practice of systems engineering for mission-critical real-time systems. The goal of the project is to establish a real-time systems engineering practice with a focus on both management and technical aspects. Primary management aspects include acquisition planning, metrics, and risk management. Primary technical aspects include predictability, fault tolerance, and interoperability. It is especially important to recognize that a system is usually part of a larger system (*system of systems* perspective).

As part of this project, a workshop was held at the SEI Symposium in September 2000 to identify key issues and obtain feedback on the topic of real-time systems engineering. The workshop had the following objectives:

- Outline an approach to evolve a practice from both acquisition/management and technical perspectives.
- Obtain feedback from the participants regarding key aspects of a practice.
- Begin to develop a community interested in collaborating in the development of a practice.

The workshop was organized as a small set of invited presentations followed by working sessions for open discussions. The workshop attendees are listed in the appendix at the end of this report.

## 1.1 Presentations

The morning session consisted of two keynote presentations that were broad in their scope of addressing the topic of real-time systems engineering. These were followed by three presentations related to predictability, fault tolerance, and interoperability. The agenda follows:

### Workshop Agenda

8:00	Mr. John Foreman, SEI, Welcome and Workshop Overview
8:30	Dr. John Short, Naval Undersea Warfare Center, <i>The Challenge of Real-Time Systems Engineering of Submarine Systems in the Second Millennium</i>

9:15	Mr. Frank DeBritz, Lockheed Martin Naval Electronics and Surveillance Systems, <i>Real-Time Systems Engineering</i>
10:30	Mr. Joe Gwinn, Raytheon Electronic Systems, <i>Predictability — The Key to Real-Time</i>
11:00	Mr. Chris Walter, WW Technology, <i>Issues Related to Fault Tolerance</i>
11:30	Mr. Henry Gruner and Peter Smith, American Systems Corp., <i>Battle Force Design and Interoperability; A "System of Systems" Engineering Integration and Management Problem</i>

## 1.2 Working Session

The afternoon consisted of a working session which was an open discussion of significant issues in the broad domain of real-time systems engineering. This session allowed participants to express their views and concerns about this domain. The discussion was moderated and the attendees' comments were recorded. The attendees' comments, in addition to information in the presentations, were used to produce this report.

## 1.3 Workshop Materials

The slides used by the workshop presenters, as well as this report, are available on the following SEI Web site: <[http://www.sei.cmu.edu/dsu/rtse\\_workshop/](http://www.sei.cmu.edu/dsu/rtse_workshop/)>.

---

## 2 Foundation Issues

By the term *foundation* we mean the collective intellectual capital that forms the basis for real-time systems engineering. Although it is accepted that systems engineering is a recognized discipline, it is not clear that there is a discipline for the systems engineering of real-time systems.

In the domain of real-time systems engineering, attention is paid to particular system attributes. Examples, some of which were highlighted in this workshop, include predictability, fault tolerance, and interoperability. The workshop attendees agreed that there are sub-disciplines associated with, for example, fault-tolerant systems. However, it was not clear that there was an *integration* of these various sub-disciplines in the context of the overall system. This lack of integration applies also to the application context (in addition to operating system and other services), where there is a lack of underlying principles on which to perform an acquisition. The challenge of system attribute integration is fundamental to the successful acquisition of today's systems.

It is clear that there is a close relation between systems engineering and software engineering. Over time the relation has changed, with the emphasis shifting more toward software than systems issues, due to the increasing importance (and cost) of software. One point made was that software engineering practices should be used as leverage to address systems engineering issues.

There are other aspects of the systems vs. software approach that relate to the contracting aspect of acquisition. For example, the contracting and systems engineering communities may emphasize functional aspects of a system. However, in the software engineering community the emphasis may be placed on an object-oriented perspective. How does one effectively relate these different world views?

A final theme that was brought out dealt with the larger context referred to as *system of systems* (i.e., a system that is part of a larger system). It was felt that there were several fundamental issues in this larger context, for example:

- Is there a need for a new paradigm to better address a system of systems?
- What is the right systems engineering process for the challenging system of systems context?

In each case identified above, the attendees recognized that there are new challenges, both technical and managerial in nature. Some issues mentioned included horizontal integration, more front-end systems engineering, and the education of management about fundamental challenges that lie ahead. The attendees also stated that there was a need for an overall framework to support a new process, recognizing there is more value in architecture than in code, for example.

Taken together, the above considerations may be viewed as presenting a serious challenge to the community interested in real-time systems engineering. Is there a foundation for this community, and if so, what is it? And can the foundation be scaled to handle the system of systems challenges?

---

## 3 Management Considerations

A number of management-related concerns were identified and discussed during the workshop. These are summarized in terms of acquisition and education.

### 3.1 Acquisition

Program managers are directly involved in the acquisition process and it is natural for them to be concerned about real-time systems engineering. The role of acquisition reform and the anticipated changes associated with it have created a new environment in which the program manager must operate. Yet it is not clear how the new acquisition regulations can be applied in a successful manner. It is clear, however, that streamlining the acquisition process will increase the need for greater use of risk management approaches.

One point was made, in a number of ways, that there is a need for acquisition guidelines. Although the burden may shift to a contractor for developing a system, the government still must evaluate proposals and manage risks. Thus, it was felt that guidelines for selected topics in the real-time systems engineering domain would be of special utility to a program manager. One example in particular was a request for guidelines relating to a system architecture, including approaches to evaluate a real-time system architecture.

Further relating to the acquisition are considerations for the rapid upgrade of a system. There is much greater pressure to provide additional functionality or capability to the user community. This comes at a price: there are tradeoffs that must be considered as part of the systems engineering for technology insertion. While it might be nice if a system were sufficiently modular for one to substitute piece-parts, that is seldom the case. The existence of subtle dependencies and side-effects, particularly for real-time systems, can be treacherous ground when a system is upgraded without an adequate foundation in place.

Another topic dealt with cost considerations, a key concern of program managers. In particular, what are the ways in which the systems engineering process can be applied to make a system more affordable? It was noted that previous acquisition approaches, with a heavy emphasis on the initial specification of requirements, produced enough failures to cause different approaches to be considered.

The Congressional emphasis on the use of commercial off-the-shelf (COTS) products was felt to be one approach that could decrease cost. However, it was not clear whether COTS products

will be sufficient to solve all the problems in the real-time domain. Further, if there are “some spectacular COTS failures looming on the horizon” (as one attendee put it), what is the alternative approach to further mitigate potential failures?

A final topic that was mentioned was removing legal impediments to allow easier insertion of new technologies. Of concern here were legal issues associated with reengineering. For example, reengineering a product to gain an understanding of behavior or to identify defects may be necessary for integrating a commercial product with a new application. However, there could be serious legal issues in reengineering a commercial product.

## 3.2 Education

One general concern that was raised is the perception that real-time systems engineering is more a collection of experiences than it is a discipline. This premise, if true, implies a serious need that must be addressed for the successful acquisition of real-time systems. What, for example, is the structure of a framework that embodies systems engineering for real-time, or fault-tolerant, interoperable systems?

In general terms, the attendees agreed that there has been a loss of systems engineering expertise, in part to other industries. This loss was perceived to be in both the knowledge of processes and technical depth and knowledge concerning the details of the manner in which a process is performed. The loss of expertise has serious impacts on the amount of sophistication that can be brought to bear in system acquisition. This problem is exacerbated by the decrease in size of program offices. This decrease in the size of program management offices plus the loss of significant expertise presents a difficult risk that must be addressed.

A number of general approaches were identified to deal with the loss of expertise and the need for better education. Perhaps most prevalent was a need for better education, including handbooks, for the real-time systems engineering community. It was also stated that the appropriate curriculum elements need to be identified. Some other topics that would help in educating the community are discussed in Section 5.1 on page 19.

In the specific context of program management, it was pointed out that a lack of education for program managers contributes to misunderstanding between a program manager and the technical staff. This illustrates the importance of providing sufficient education to program management staff in the key issues for the real-time systems engineering domain. Some suggestions to help the manager included the following:

- Valid arguments need to be demonstrated to program managers about the importance of a system architecture, and program managers need to understand this point.

- A list of 20 questions that managers should ask and an explanation of how to understand the consequences of responses should be developed.
- Availability of checklists would help managers easily assess key technical points. Note, however, that a checklist is not a substitute for depth of either knowledge or understanding!

An undercurrent that persists in the education of management is the potential tension between the management staff and the technical staff. For example, the technical staff need managers to understand the value of front-end systems engineering work. However, this directly conflicts with a perceived “rush to judgment” to get a system deployed. A common understanding of the entire process by all relevant staffs is needed to narrow the gap in understanding.





---

## 4 Technical Considerations

In this section we discuss technical considerations that were identified at the workshop. First we discuss technical issues associated with the characteristics of predictability, fault tolerance, and interoperability. Then we describe some additional considerations that were noted.

### 4.1 Characteristic-Specific Issues

#### 4.1.1 Predictability

*“Real-time does not mean really fast.”*  
- Workshop participant

As used in this workshop, *predictability* refers to the ability of a system to satisfy timing properties. This is a characteristic of real-time systems, which typically are expected to

- respond predictably to urgent events
- have a high degree of schedulability (meet specified deadlines)
- maintain stability under transient overloads

The real-time characteristic is further divided into

- hard real-time: systems where failure to meet a deadline is interpreted as a system failure
- soft real-time: systems where failure to satisfy a deadline is not considered a failure. In this case, the response to events may be specified in terms of a (bounded) statistical distribution function.

Real-time systems must typically satisfy various timing requirements. Among them are

- responsiveness: bounded and predictable response to some event
- efficiency: ability of underlying components, notably middleware and operating system, to provide services, such as bandwidth, to an application
- overload response: reaction to an overload situation. An overload situation can arise naturally or in response to a fault. Also, the system must be able to maintain or release control as appropriate. For example, if an application blocks forever waiting for the arrival of a

message, it can do no other productive work. In a case such as this, the use of time-outs or asynchronous services is necessary.

Much of the application support to achieve predictable performance is provided by the operating system, or real-time kernel. One way to partition operating system implementations is by their typical responsiveness, namely:

- order of tens of microseconds
- hundreds of microseconds
- thousands of microseconds

Another partition of operating systems can be based on the source of its specification. Some interesting observations include

- real-time kernels: typically several hundred thousand lines of code, principally designed for embedded real-time control applications
- UNIX kernels: typically one to two million lines of code, designed for general purpose use
- Windows variants: over 40 million lines of code that are intended for the desktop environment
- hybrids (such as RT Linux): typically a real-time operating system bolted on top of a UNIX-like system so that they share characteristics of each

Successful development of a real-time system requires that care be given to the application development and the manner in which the application uses services provided by the operating system. An example of such a concern is priority inversion, where a high-priority thread of control must wait for a shared resource before continuing to execute. If there is unbounded priority inversion, where the amount of time a thread waits is not bounded, there are serious implications for the application to satisfy timing requirements. The near-loss of the Mars Rover is a classic example of the consequence of unbounded priority inversion.

One approach to dealing with possible priority inversion is to employ services that are designed to manage a shared resource. Two examples are the priority inheritance protocol and the priority ceiling protocol. In the case of the former, for example, the priority of the shared resource is adjusted to that of the priority of the highest waiting thread.

Simply providing services to meet real-time requirements, such as a protocol for management of shared resources, is necessary, but not sufficient. Multiple implementations may provide similar (functional) services, yet they differ dramatically in their performance. It is therefore necessary to develop means by which an application developer can gain an understanding of the performance characteristics of implementations, especially COTS products.

One approach to understanding performance characteristics of a COTS product is through the use of benchmarks. Various types of benchmarks can be constructed, with one classification

scheme based on the scope of the benchmark. On the one hand, small-scale benchmarks can be used to measure a particular service provided by an operating system. On the other hand, benchmarks can be constructed to mimic the overall performance of an application.

One might expect that a standard for some component, such as an operating system, could include a specification of benchmarks, reflected as performance metrics. Such an approach was tried on two occasions in the portable operating system (POSIX) standardization effort. However, neither of these attempts was successful. In one case, it was not possible to specify a set of metrics that were simple enough to implement, but complex enough to represent a faithful model of an actual application. There are also considerations of vendor “optimization” efforts to recognize benchmarks and respond in a particular manner. In the second case, there were serious concerns raised by vendors that providing certain qualitative information about their implementation might cause them to lose competitive advantage. The inability to develop standards for benchmarks is a risk that needs to be managed in an acquisition.

## 4.1.2 Fault Tolerance

*“It’s hard to design a system that won’t fail.”*  
- Workshop participant

Systems that are currently being developed, and those expected in the future, are more complex, include more stringent real-time requirements, incorporate significant COTS products, have greater demands on dependability, and are expected to operate in less restrictive environments. It is inevitable, however, that systems will fail. How then can we develop systems that satisfy increasing requirements, but at the same time provide a corresponding increase in assurance that if a failure occurs, it is handled appropriately? The approach presented at the workshop included several facets of the above question and involved consideration of

- management concerns
- technical approaches
- verification and validation

Management was identified as a concern because experience shows that most program managers do not have a thorough understanding of fault tolerance. There are consequences to this lack of understanding. For example, program managers may not understand the consequences of choices that are made in the development effort (which may take longer than desired to achieve a sufficiently robust design). If program managers view fault tolerance as too complex, they may be likely to settle, or press for, an oversimplified approach to meet cost or schedule constraints. Although traditional risk management approaches apply to fault tolerance, there is a certain amount of knowledge of fault tolerance that is required to manage risks in that domain successfully. As with other topics discussed at the workshop, the education of program managers is an issue that must be addressed directly.

The design and development of fault-tolerant systems requires special considerations. A number of issues were specified, including

- identification of fault models
- identification of failure semantics
- use of transactional fault-tolerant (atomic) mechanisms
- role of online detection and diagnosis of faults
- load balancing

It was emphasized that each of the above issues requires a *system* perspective. This perspective includes an understanding and specification of the problem, flexible architectures to support the use of different approaches, and the use of a theoretical approach to the problem. This is especially true in the real-time domain, where there may be performance issues associated with the use of various fault-tolerant approaches. Although there is a need to engineer for consequences of a failure, it may not always be clear when enough has been done and when to stop.

For verification and validation, there are current approaches that apply to fault-tolerant systems. It was argued, however, that there is value in the increased use of formal methods. Formal methods are tools that can provide advantages such as specifying the system to understand its expected behavior, and then using the formal specification as a means against which to test.

It is safe to say that formal methods have had a checkered history in their application to systems engineering. More often than not, they have not been used, but where they have, there is recognized value in their application. Further exposition of the use of formal methods would be a workshop in itself!

### 4.1.3 Interoperability

*“We need to return to fundamentals.”*  
- Workshop participant

The term *interoperability* has a number of definitions; for our purposes, we consider interoperability as the ability of systems to exchange and use information. Interoperability assumes a position of central importance in the context of *system of systems*. In fact, in the system of systems context, the issues are broader than ever before. They are not simply technical issues, but impact systems, processes, procedures, organizations, and missions.

There are trends and pressures that make achieving interoperability difficult. On the one hand, current trends include decreasing budgets, uniqueness of fielded systems, and shorter cycles associated with system upgrades. On the other hand, there is increased pressure (better, faster,

cheaper!) in acquisition (e.g., DoD 5000.2), the use of COTS products, joint programs, and rapid technology insertion. An integrated systems engineering discipline must be able to account for the above trends and pressures.

An example where interoperability is an important issue is in the integration of C4 (Command, Control, Communications, and Computers) and weapons systems. Weapons systems are traditionally real-time in nature, while C4 systems are traditionally non-real-time. The integration of C4 systems and weapons systems has increasing performance (including functional) expectations from the user community. Many of the preceding comments apply to this general class of problems: lack of a process to apply in development and test, lack of management understanding, and lack of expertise in program offices to develop useful acquisition documents (such as a request for proposal). Hence, the issues that were identified in a local context now propagate to the system of systems context, and there are new concerns that must be addressed.

Clearly, one of the concerns is legacy system integration in a larger context; large-scale systems will not be built from scratch. However, legacy systems have their own problems when integrated in a larger context. For example, legacy systems may have never been designed to integrate with other systems. The fact that legacy systems may be based on different technologies, and different implementations, can make integration of legacy systems all the more difficult.

The concerns for system of systems and interoperability considerations are broader in scope than in platform-centric systems. In addition to technical consideration such as engineering process and architecture development, organizational issues (including politics) play a larger role.

Some of the key issues related to engineering, integration, and management processes include a need for the application of a new, unified, top-down approach that addresses

- requirements analysis
- functional analysis and allocation of requirements to systems
- integration and synthesis of systems

Each of the above activities must be performed in an integrated management context that considers all of the elements that constitute the system of systems. Current trends are toward a bottom-up system-integrated approach, in contrast to the application of first-principle, top-down approaches.

Architectural considerations are pervasive when considering a system of systems. Some of the concerns addressed the need for an integrated information flow architecture, allocation of

functional and performance requirements to architectural components, and the ability to deal with architectural mismatches when systems are integrated to form a larger system.

Just as there was concern expressed about the lack of an integrated discipline related to real-time systems engineering, that same concern applies in the larger, system of systems context. That is, the perceived lack of a disciplined process for the integration of timing predictability and fault-tolerant characteristics in a system context is magnified when we consider the larger system of systems context.

Another topic of relevance was the development and test process for systems that must interoperate. The development process needs to account for distributed configuration management. It must also support cooperative distributed engineering and integration test. There has been success in the use of early integration testing through the use of distributed, wide-area integration facilities. System testing in the context of a distributed engineering plant appears to hold promise. Here, one can integrate the system of systems in a developmental context without experiencing the costs associated with operational deployment.

## **4.2 Additional Considerations**

### **4.2.1 Standards**

Standards are valuable because they represent the codification of a state of the practice in some domain. They also provide a specification that can serve as the basis for COTS products. It is the pipeline of standards to COTS products that can serve as the foundation for technology insertion.

However, it was noted on multiple occasions during the workshop that there are missing standards for the domain of real-time systems. Examples noted were the need for standards to ensure predictable behavior, fault tolerance, and interoperability. Another area where standards do not exist is the remote management of a system. The lack of standards implies a lack of conforming COTS implementations. Thus, developers must each create interfaces and implementations of their own choice, often working independently. There is a lack of reuse on a large scale because of fragmented development efforts.

In addition, there are no standards that describe processes for the acquisition of real-time systems. It was noted that there is not a specified, repeatable practice of real-time systems engineering. In such a case, it is clear that standards cannot exist. Yet it is also recognized that such process-oriented standards would add value to the domain. For example, a standard that defined an overall process for the engineering of real-time systems would be of clear value.

A necessary condition for the development of appropriate standards is the existence of a community that understands the domain and that has sufficient expertise and resources to pursue the standards-development efforts. This topic is further explored under the discussion of Community Considerations in Section 5.

### **4.2.2 COTS Products**

As we noted earlier, there is an increased emphasis on the use of COTS products. It is hoped that the use of COTS products will allow organizations to take advantage of a competitive market and to develop and insert technology faster. The increased use of COTS products in a top-down engineering model does not work well: the analogy provided was that one goes to a store, sees what is available, buys selected items, and then assembles those items. This is a major change from a traditional top-down model and adds complexity to the acquisition process that is needed to meet the goals of rapid technology insertion.

COTS products are just one example of system components that are provided by others. Implementation details may not be available to the component user. However, for such components to be used effectively and predictably, their specifications must be enriched to provide relevant information that can be validated. The following issues are of particular concern:

- enrichment of interface specifications by adding relevant models of behavior and other expected component properties
- robust black-box testing of components to ensure compliance with specifications and reliability, especially in systems where fault tolerance and safety are of importance
- understanding system-level performance implications of component performance

The above are just some of the issues that must be faced in the use of COTS products. Clearly, COTS products can affect system performance and fault tolerance. In addition, COTS products can have large-scale effects in integration, especially in the system of systems context. It is not clear that the relevant risks to the use of COTS products have been identified. Of greater concern, perhaps, is the lack of a disciplined process for acquisition that treats COTS products as first-class components.

### **4.2.3 Modeling**

During the workshop, it was recognized that there are various opportunities where modeling can help in the acquisition process for real-time systems. Of particular interest was the development and use of models for the operational system. Some of the ideas expressed included the following:

- models of COTS products and their performance characteristics

- analyzing how sensitive various system properties are to changes (e.g., how well does a particular system architecture accommodate change in certain dimensions, such as input load)
- models to experiment with the system of systems context

It was pointed out that the modeling needs to be at the appropriate level for a particular purpose. For example, models used early in the acquisition may differ dramatically from those used to obtain quantitative performance measures later in the process. Other types of models can be applied. For example, formal specification, benchmarks, and simulation have value in the development of the operational system.

#### 4.2.4 Verification and Validation

In the domain of real-time systems engineering, verification and validation (V&V) are important because many of the systems are mission critical in nature. In addition to V&V activities, certification also is an important concern. Traditionally, verification and validation have consisted of a limited degree of design analysis and exhaustive testing of designs that exhibit a high degree of determinism in terms of both functional and timing behavior. However, such methods do not scale well, and integration test costs dominate development costs. This leads to issues of

- demonstrating the value of front-end engineering (i.e., early and continuous analysis of systems in terms of behavior, timing, performance, and reliability)
- understanding the benefits and limitations of various V&V methods in the context of real-time systems; where and when they can be applied appropriately
- understanding the technical and social aspects of performance specification and benchmarking
- scaling and adapting the certification process to rapidly changing large-scale systems—combining and leveraging build–test and design–analyze approaches
- role of self-testing, especially for COTS products, in a system context
- relation between a “build a little, test a little” approach and V&V methods

The use of modeling and analysis of system architectures and their properties, such as behavioral model checking, schedulability analysis, and reliability analysis was recognized as having value for the V&V effort.

#### 4.2.5 Evolution

Evolution in systems and their deployment environment is inevitable. System capabilities must be added, and software technologies used in the implementation must be upgraded or replaced repeatedly over the life of the system. The following issues are of particular concern:



- engineering tradeoffs in rapidly changing technology: what, when, and how often to upgrade
- cost of accommodating change and portability in terms of performance
- prototyping and benchmarking of key technologies to understand impact on performance
- understanding technology trends in terms of resource capacity and understanding system trends in terms of resource demands to predict and plan for technology replacement
- ability to perform fault-free upgrades to a system
- market analysis for the assessment of standards and associated COTS products that may be of relevance

As one might expect, in a system of systems context, the above issues are amplified. In addition, acquisition personnel must now address programmatic integration and synchronization among the relevant programs and the systems for which they are individually responsible.



---

## 5 Community Considerations

*“A rising tide lifts all boats.”*

- Workshop participant

One of the goals of the workshop was to initiate a community of interest for the real-time systems engineering domain. Support for this idea was expressed by several attendees who indicated that interacting with people from other programs, who often faced similar challenges, made their attendance at the workshop very valuable.

### 5.1 Knowledge Sharing

The most frequent comment was that a community could share interests and experiences based on lessons learned. One example cited was experience in the use of COTS products in a real-time, fault-tolerant system. Drawing on the broader community would benefit other programs and projects. Possible sources for information could be organizations, as well as lessons learned from independent technical assessments (sometimes referred to as “red teams.”) There was a request that the lessons learned be presented in a simple manner, perhaps in a “cook-book” form. It was also noted that lessons learned should be focused on good news as well as bad news. A related idea was that the community needs a repository for best practices for real-time systems engineering.

It was also recognized that there are many problems that are common across different organizations. Some of the suggested topics that are worthy of a community approach include

- missing standards: It was recognized that while standards are important, there are standards that do not exist. Some suggested areas include standards for fault tolerance and middleware.
- horizontal integration: Although this term is receiving increased attention, it was not clear what it really means. In particular, what are the technical and management issues that must be addressed, especially for integration across different domains?
- middleware: The use of middleware is becoming more common especially to achieve a specific purpose. However, it is also necessary to achieve a solution whose scope is from the application interface to the distributed system infrastructure. Hence, how does one effectively deal with localization of middleware effects?

## 5.2 Toward a Practice of Real-Time Systems Engineering

An established engineering practice has several key characteristics:

- a community that uses a common terminology
- a set of proven best practices
- training and education

Common terminology is important to a community in order for its members to communicate effectively. Attendees felt that a common terminology does not currently exist. For example, the terms *open* and *modular* are sometimes used interchangeably, but they are also used in very different ways. In addition to developing common terminology, the attendees felt that communication of system concepts and experiences would be valuable to the community.

There are generally accepted industry practices that are applied in an acquisition. Some examples of these include risk management, formal inspections, and configuration management. However, there do not appear to be sufficient best practices in the real-time systems engineering domain.

A third aspect for an established practice is the ability to provide training and education. Several times in this report the need for education has been noted. This includes educating management and technical staffs at various levels.

## 5.3 Consortium

Finally, the attendees suggested that a consortium be established in the area of real-time systems engineering. This consortium should be open to members from government, industry, and the academic community. In principle, the consortium should be broad in its scope. It should also investigate new approaches to systems engineering that may result in leaner and more versatile approaches to acquisition. It was felt that the consortium should focus on topics that are non-competitive. That is, the work performed by consortium members should not provide a competitive advantage to one member over other members.

Possible areas of work that were suggested for a consortium included the following:

- Work with Air Force Institute of Technology (AFIT), Defense Systems Management College (DSMC), and Naval Postgraduate School (NPS) to improve real-time systems engineering education programs.
- Work to establish a curriculum in real-time systems engineering for undergraduate institutions.

- Work within the scope of the professional engineering community to establish licensing programs.
- Work with senior acquisition executives to initiate congressional action to modify acquisition laws to allow for agile acquisition practices.

Finally, it was noted that a consortium would allow members to develop common, consensus-based practices in relevant domains. A consensus-based framework for real-time systems engineering was recognized as being very valuable.



---

## 6 Summary

This report has presented the results of presentations and discussions at the Real-Time Systems Engineering Workshop, held in conjunction with the SEI Symposium in September 2000. Many issues were raised throughout the workshop. The following list summarizes what we believe were the most important issues raised during the workshop:

- Although there is a recognized practice of systems engineering, and there are practices associated with predictability and fault tolerance, there is a lack of an integrated discipline for real-time systems engineering.
- There is a pressing need to educate program managers and their staffs about fundamental acquisition issues in the real-time systems engineering domain.
- The problems associated with achieving an interoperable system of systems loom large on the horizon.
- There is a need to establish a community of interest for dealing with real-time systems engineering.

The attendees agreed there is a need for continuing the interaction and work initiated at this workshop.





---

## Appendix: Attendees

<b>Name</b>	<b>Company</b>
Jeff Calcaterra	NRO
Terry Dailey	SEI
Frank DeBritz	Lockheed Martin
Tom DeLuca	NAVSEA PEO-TSC
Rick Dotson	Bechtel Bettis
Doug Elgin	Teledyne Brown Engineering
Peter Feiler	SEI
John Foreman	SEI
Dave Gluch	SEI
John Goodenough	SEI
Henry Gruner	American Systems Corp.
Joe Gwinn	Raytheon
Barbara Langworthy	Mercury Computer Systems
Harris Liebergot	NIST/ATP
Bill Locke	Bechtel Bettis
Ted Marz	SEI
Robin McCollum	Bechtel Bettis
Brian McNamara	NAVSEA 08K
Craig Meyers	SEI
Barry Miller	Anteon Corp
Gus Neitzel	NRO
Mike Nifontoff	NAVSEA 08K
Steve Palmquist	SEI
Tuan Phan	PEO-TSC PMS461
Daniel Plakosh	SEI
Tom Roberts	NAVSEA 08K
Linda Roush	NAVAIR Warfare Center, Weapons Division, China Lake

<b>Name</b>	<b>Company</b>
John R. Short	Naval Undersea Warfare Center
Jim Smith	SEI
Robert Stoddard	Motorola
Chris Walter	WW Technology Group
Oliver Watterson	Naval Undersea Warfare Center
Chuck Weinstock	SEI
Duard Williams	CSX Transportation
John Wood, LCDR	US Coast Guard

# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

7. AGENCY USE ONLY (leave blank)		8. REPORT DATE August 2001	9. REPORT TYPE AND DATES COVERED Final
10. TITLE AND SUBTITLE Proceedings of the Real-Time Systems Engineering Workshop		11. FUNDING NUMBERS F19628-00-C-0003	
12. AUTHOR(S) B. Craig Meyers, Peter H. Feiler, Ted Marz		14. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2001-SR-022	
13. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		16. SPONSORING/MONITORING AGENCY REPORT NUMBER	
15. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		16. SPONSORING/MONITORING AGENCY REPORT NUMBER	
17. SUPPLEMENTARY NOTES			
12.a DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12.b DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words)  This report presents the results of a workshop on the topic of real-time systems engineering. The workshop was held as part of the SEI Symposium in Washington, DC, during September 2000. The objective of the workshop was to identify key issues and obtain feedback from attendees concerning real-time systems engineering. Two broad keynote presentations were presented, followed by topical presentations concerning predictability, fault tolerance, and interoperability. This report summarizes the workshop in terms of foundation, management, and technical topics, and it contains a discussion related to developing a community of interest for real-time systems engineering.			
14. SUBJECT TERMS system engineering, real-time systems, fault tolerance, interoperability		15. NUMBER OF PAGES 26	16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL