1997

A Comparative Study and Estimation of the Life-Cycle Cost Impact of Application of Real-Time Non-Intrusive (RTNI) Monitoring Technology to Real-Time Embedded Systems

Michael D. Lewis

AFIT/GSS/LAS/97D-2

A COMPARATIVE STUDY AND ESTIMATION OF THE LIFE-CYCLE COST IMPACT OF APPLICATION OF REAL-TIME NON-INTRUSIVE (RTNI) MONITORING TECHNOLOGY TO REAL-TIME EMBEDDED SYSTEMS

THESIS

Michael D. Lewis, Captain, USAF

AFIT/GSS/LAS/97D-2

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

AFIT/GSS/LAS/97D-2

A COMPARATIVE STUDY AND ESTIMATION OF THE LIFE-CYCLE COST IMPACT OF APPLICATION OF REAL-TIME NON-INTRUSIVE (RTNI) MONITORING TECHNOLOGY TO REAL-TIME EMBEDDED SYSTEMS

THESIS

Presented to the Faculty

of the Graduate School of Logistics and Acquisition Management

of the Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Software Systems Management

Michael D. Lewis, B.S.

Captain, USAF

December 1997

Acknowledgments

I wish to thank my thesis advisor, Mr. Dan Ferens, for all of his help. I am also appreciative of my reader, Major Mark Kanko, for all of his assistance in lending some of his software testing expertise to my paper. I would also like to recognize Doctor Kim Campbell for her help with the Nominal Group Technique.

I am grateful to Richard Fryer, Martin Freed, Lieutenant Colonel Stephen Atkins, and Don Stearns for all of their help with this thesis. Also, I am very indebted to all of the participants in the expert judgment panel for their invaluable contributions. Although my project was not one of their priorities, they did provide timely and invaluable contributions to this study.

Most of all, I wish to thank my fiancée, Adele, for all of her support and understanding. I am sure that it must have seemed like there were times that she was second to my work, but she will always be the most important part of my life.

Michael D. Lewis

Table of Contents

Page

Page

List of Tables

Abstract

The use of real-time non-intrusive (RTNI) monitoring has had an impact on life-cycle costs of existing programs through a reduction in debug time. Other areas in which RTNI monitoring can provide potential benefits to future programs are through the use of increased dynamic testing and the sharing of testing time among more engineers. There are a number of areas in which software life-cycle costs are impacted by various cost drivers. To determine which areas were affected by the use of RTNI monitoring, a panel of expert users of RTNI monitoring was created using a form of the Nominal Group Technique (NGT) methodology to achieve a consensus from a group of experts. The group concluded that the above areas were most important across their programs. However, simply using RTNI monitoring may not have a major impact on future programs, unless it is accompanied by the commitment from management to successfully integrate it into the test programs of those future programs.

A COMPARATIVE STUDY AND ESTIMATION OF THE LIFE-CYCLE COST IMPACT OF APPLICATION OF REAL-TIME NON-INTRUSIVE (RTNI) MONITORING TECHNOLOGY TO REAL-TIME EMBEDDED SYSTEMS

I. Introduction

General Issue

Software has played an increasingly more important role in all military systems. Much of the avionics in today's military aircraft use embedded systems, systems that use computer processors and software that are embedded within a larger system (Glass, 1980: 265). Real-time embedded systems are typically required to be small, light, and inexpensive, and as a result, to control costs, designers often transfer the complexity of the system from hardware to software (Glass, 1980: 265). The reason for this is that multiple copies of software cost no more than one copy, while the cost of multiple copies of hardware changes by a factor of the number of copies (Glass, 1980: 266).

The problem with using embedded systems is that they contain a mix of hardware and software that is very difficult to test, as they are often dependent on various data feeds that have timing constraints imposed by real-world processes (Plattner, 1984: 756). The integration and testing of these systems provides challenges as all of these feeds must be serviced in real-time without affecting the basic operation of the software.

As a result, a new technology has been developed that is referred to as Real-Time Non-Intrusive (RTNI) Monitoring. This technology focuses on sending real-time signals

through the embedded systems and monitoring the results at several locations across the processors to pinpoint specific problem areas. Currently, hardware problems can be identified in an embedded system with the use of hardware-specific tools, such as an oscilloscope or a frequency generator, while software problems can be identified through the use of current techniques such as the static line-by-line walk-through of the code or dynamic debugging tools that require analysis of output. RTNI monitoring allows testers to simultaneously test for hardware and software problems without affecting the normal operations of an embedded system.

The strength of RTNI monitoring lies in its ability to detect software defects using monitors that have been attached to the system at several points. RTNI monitoring can detect bugs during run-time, and the tester does not have to examine output and track errors to the source – with RTNI monitoring, the defects are discovered, even if they do not affect the output (Daich et al., 1994: 8).

In 1990, the Defense Systems Management College expected that software costs would rise at rate proportionately greater than computer hardware costs (DSMC Report, 1990). This trend was supported by the Department of Defense Inspector General in a February 1991 audit (DOD IG Audit, 1991).

A 1993 report by the General Accounting Office (GAO) stated that the Department of Defense "expects post-deployment software support costs, typically 70 percent of life-cycle costs, to be less when disciplined, measurable software development processes are employed" (GAO Report, 1993: 4). The report states that the Department of Defense "estimates that 14 percent of these [software support] costs are attributable to

correcting software errors missed during testing and earlier development phases" (GAO Report, 1993: 5). The GAO report further added that:

> Up-front improvements in the quality of software development processes and more effective software test and evaluation may play a significant role in controlling these [software support] costs, which are incurred largely after systems have been fielded. (GAO Report, 1993: 11)

As a result, an up-front improvement to the test and evaluation of embedded systems, such as RTNI monitoring, may have a significant role in reducing overall life-cycle costs. Through the use of better software testing tools, such as RTNI monitoring, during development, more errors can be detected and corrected in development, which could result in a net-gain of 9.8 percent of total life-cycle costs (14 percent of 70 percent of life-cycle costs). RTNI monitoring is one of the tools that can be employed to better detect errors earlier and improve the test and evaluation of embedded systems.

Following the GAO Report in May 1994, the Office of the Secretary of Defense issued a memorandum called "Software Maturity Criteria for Dedicated Operational Test and Evaluation (OT&E) of Software-Intensive Systems". In this memorandum, the conditions stated include:

> System functionality to be operationally tested and evaluated must be available prior to the start of OT&E and must have been developmentally tested. In particular, the system features that are required to support specific requirements and the system interfaces that are required to interoperate with external systems must be certified to be functional, preferably in an operationally realistic environment (real users, data, procedures, etc.) against operational requirements. (Frame, 1994)

To comply with this requirement, software developers and testers can take advantage of RTNI monitoring to complete this testing for real-time embedded system. The issue is whether or not the use of RTNI monitoring can save money during this testing.

Specific Problem Statement

RTNI monitoring can provide many capabilities to developers and maintainers that currently cannot be provided by any other means. The cost of adding RTNI monitoring technology to existing or future systems can be great, but the potential payoffs from the use of RTNI monitoring technology can save time and money in the long run. Additionally, RTNI monitoring could increase system reliability.

No definitive proof exists to show that RTNI monitoring can save money. This report will investigate the monetary and time savings benefits of systems that currently use RTNI monitoring and determine if the Air Force is justified in advocating its use in future weapons systems as a means of testing the software in real-time embedded systems.

Research Objectives

The purpose of this research is to provide Department of Defense system program offices with an analysis of the potential life-cycle cost benefits of the use of RTNI monitoring. This research will also attempt to determine the life-cycle cost savings associated with RTNI monitoring technology.

Scope of the Research

This research will address the use of RTNI monitoring technology in the development and maintenance of several weapon system programs. The programs will be from the Department of Defense and the National Aeronautics and Space Administration (NASA), and will be limited to only those organizations that have experience in using RTNI monitoring technology.

While the impact of RTNI monitoring will not be available in specific dollar amounts, this study will determine if the use of RTNI monitoring has an overall positive (cost-effective) impact on life-cycle costs or a negative (cost-ineffective) impact on life-cycle costs. Several cost drivers will be identified, and the results given for each cost driver. This study will be limited to determining if there is a life-cycle impact and to what extent life-cycle cost drivers can be affected.

II. Literature Review

Chapter Overview

This literature review investigates five areas: software life-cycle cost drivers, types of software testing, real-time non-intrusive monitoring, the role in which real-time non-intrusive monitoring plays in software testing, and previous studies of the effect of real-time non-intrusive monitoring on life-cycle costs.

Software Life-Cycle Cost Drivers

The goal of this study is to determine if it is economically feasible to use RTNI monitoring in future Air Force systems. In the book, *Testing in Software Development*, the editors state that "a design is feasible if the life-cycle benefit of the system exceeds the life-cycle costs" (Ould and Unwin, 1986: 65). Since total life-cycle costs are not known until a system has reached the end of its life-cycle, cost estimation must be used to predict the life-cycle costs in the interim.

The factors that contribute to the total cost are known as "cost drivers" (Royer, 1993: 188). These factors are used in cost estimation models to determine an estimate for the total life-cycle costs. Various models have been developed, to estimate development and support costs and to give managers an estimation of the overall life-cycle costs. There is no one simple formula with which a total cost can be determined from all cost drivers input into it; each model must be calibrated to each organization's way of working

(Royer, 1993: 194). As a result, the specific cost drivers have different weights in different models for different organizations.

In some cases, it is not feasible to determine cost, but effort can be measured. Cost and effort are directly related. Boehm notes that his COCOMO cost model does not compute costs in dollars, since there is no clear definition as to what an organization considers labor costs, but that cost can easily be derived from effort for any organization (Boehm, 1981: 61). Brooks notes that cost varies as the product of the number of persons working on the project and the number of months, which is used as the unit of measure of effort (per person per month) (Brooks, 1995: 16). Thus, if overall effort is impacted, overall cost is impacted as well.

The impact that RTNI monitoring has on a project's life-cycle cost is primarily in testing costs. Since RTNI monitoring is a method of testing, the impact that RTNI monitoring has on life-cycle costs will be mostly through the impact that RTNI monitoring has on total testing costs.

Royer's Views on Testing Costs

Thomas Royer examines testing cost and testing cost estimation in his book, *Software Testing Management: Life on the Critical Path.* He suggests that a "test cost driver" is "any measurable quantity whose value can be shown to have a positive coefficient of correlation with testing costs and that has some apparent casual connection with them" (Royer, 188). He then mentions that the size of a system and the specification and design of a system correlate with testing costs, thus making them cost drivers.

Royer suggests the size and complexity metrics that should be used to measure these cost drivers. For size, some metrics that can be used are lines of code, number of modules, and number of large components and programs. However, he notes that due to language differences, there is no relationship between lines of code and testing costs (Royer, 1993:190). There is a relationship, however, between number of modules and testing costs, but he cautions that it is not a linear relationship (Royer, 1993:191). When there are more modules in a system, testing costs will be greater, because there will be more unit testing that needs to be completed (Royer, 1993:191). However, the number of modules does not correlate to the cost of system testing; system testing is related to the number of system functions (Royer, 1993:191). A large software component, such as a Computer Software Component (CSC) as described in the work breakdown structure (WBS) of a program, is the complete software needed to implement a subsystem specification (STSC, 1996:12-30). Using a count of large components is not recommended, he adds, because a mistake in the estimation of number of large components by a small number results in a large error in the overall cost estimate (Royer, 1993:191). This error occurs, because in larger programs, there may be fewer components tracked to a level at which functionality of each component can be assessed.

For system design, Royer suggests measuring the number of interfaces, the number of requirements, and function points. Royer notes that the number of interfaces, both internal and external, has a direct effect on the complexity of the software, which in turn impacts testing time; i.e., higher complexity results in a longer testing process (Royer, 1993: 191-192). Royer notes that since test cases must be developed in a one-to-

one correspondence with software and interface specifications and requirements, the number of requirements has the greatest influence of testing costs (Royer, 1993:192). He cautions, though, that all requirements are not of equal difficulty or risk (Royer, 1993:192). Function points are not a good measure for real-time systems, because the parameters used to calculate function points do not consider attributes of real-time systems (Royer, 1993:194).

Finally, Royer states that a good rule of thumb for measuring testing costs is to assume that the total testing cost, which will include unit, system, integration, and qualification testing, is between 40 and 50 percent of the total development cost (Royer, 1993:194).

Beizer's Views on Testing Costs

Beizer notes that historically, managers have mistakenly based their life-cycle costs and testing costs on the number of lines of code. He references a study by Weinwurm, Zagorski, and Nelson that showed that costs were more influenced by such items as "number of meetings attended, number of items in database, the relative state of hardware development, documentation requirements, and other factors" than lines of code, and thus, size should not be the only consideration of cost estimating (Beizer, 1984:277).

Another cost driver is schedule. Cost and schedule are both directly related to effort. Thus, if we can impact schedule, we can impact cost. Beizer reports on a method employed by Hitachi Software, Inc., that uses the fact that the time to complete testing is

dependent on the rate of defect detection (Beizer, 1984:299-302). If we can increase the defect detection rate, we can increase the likelihood that more defects will be found quicker and testing will be completed earlier. Likewise, Beizer notes that the time to complete testing is also dependent on the rate at which test cases are completed (Beizer, 1984:296-297). Thus, if we can increase the rate at which defects are detected and the rate at which test cases are completed, the test schedule will be reduced, which would reduce testing costs. Also, if the average test case completion rate increases for a system, the result should be lower testing costs.

Humphrey's Views on Costs of Quality

Watts Humphrey presents his idea of "costs of quality" in his book, *A Discipline for Software Engineering* (Humphrey, 1995:281). Edward Yourdon suggests that the cost of quality (COQ) is one of the methods by which quality management can be achieved as part of the Quality Management and Quantitative Process Management Key Process Areas of the Level 4 Capability Maturity Model (CMM) (Yourdon, 1996:118). Watts Humphrey states that costs of quality are comprised of three areas: failure costs, appraisal costs, and prevention costs (Humphrey, 1995:281). Failure costs are the costs incurred in diagnosing a failure, repairing the fault, and re-testing the software. Appraisal costs are the costs incurred in evaluating software quality, including code reviews and inspections. Prevention costs are the costs incurred when identifying defects and the action taken to prevent them from becoming failures.

Brooks' Views of Maintenance Testing Costs

Brooks states that during system maintenance, the fixing of a defect can sometimes introduce another defect (Brooks, 1995:122). The chance of this occurring, he adds, is between 20 and 50 percent. Because of this fact, he notes that maintenance testing requires regression testing, where the entire set of test cases used for development must be performed to ensure that no new defects have been introduced. Regression testing can be very costly.

Brooks advocates designing programs with less defects to lower maintenance costs, which would reduce life-cycle costs (Brooks, 1995:122). Likewise, if regression testing can be reduced or the time to perform it reduced, the result would be a decrease in overall maintenance testing costs, which in turn would reduce life-cycle costs.

Musa, Iannino, and Okumoto's Views of Reliability Costs

In their book, *Software Reliability: Measurement, Prediction, Application*, Musa, Iannino, and Okumoto describe a "failure intensity function" as "the rate of change of the mean value function or the number of failures per unit time" (Musa et al, 1987:11). They state that increasing the failure intensity function results in both a decrease in schedule and a decrease in cost (Musa et al., 1987:26-27).

They further state that to optimize life-cycle costs, a failure intensity objective could be set to affect these costs (Musa et al., 1987:196-197). They state that with higher failure intensity objectives, the testing costs during development decrease, while operational costs increase. Thus, if the reliability is changed to allow for more failures

per hour, the testing costs will decrease. The degree to which total costs are affected is determined by the amount of trade-off gained through system testing costs and operational costs.

Software Technology Support Center (STSC) Test Selection Considerations

According to the STSC's August 1994 Software Test Technologies Report, there are several factors that must be considered when selecting new test technologies. First, when new tools are added to a process, test strategies must be reworked to allow for the new testing to be considered throughout the life-cycle (Daich et al., 1994:34). Test plans may change, and the addition of new test technologies and methodologies needs to be properly researched and integrated into the overall process (Daich et al., 1994:34). They note that "when any development phase is upgraded with new technologies, the testing process must be reconsidered since all phases involve some form of testing" (Daich et al., 1994:34).

Another factor that must be considered is that new tools will require additional training for the personnel that will use them. They state, "along with the added sophistication of new technology, come the added costs in time and money for the training necessary to use this technology" (Daich et al., 1994:37). They suggest trying to use an automated tool that can be used for a variety of projects, amongst which the training costs can be divided (Daich et al., 1994:37).

Thus, integration into the overall process and personnel training are two additional cost drivers that will have an impact on the life-cycle costs.

Types of Software Testing

There are two basic, high level forms of testing – static and dynamic. There are also four phases of testing in the life-cycle of most software projects – unit, integration, system, and qualification.

Static testing is defined by the STSC as "evaluation of software without executing it using automated (tool assisted) and manual mechanisms such as desk-checking, inspections, reviews, and walkthroughs" (Daich et al., 1994:8). The STSC report also states that static analysis alone cannot find faults that occur as a result of the operating environment (Daich et al., 1994:8). Another definition of static testing is provided by Richard DeMillo et. al. in their book, *Software Testing and Evaluation*, as "the requirements and design documents and the code are analyzed, either manually or automatically, without actually executing the code" (DeMillo et al., 1987:27).

Dynamic testing is defined by the STSC as "executing a piece of software to determine if the software functions as expected" (Daich et al., 1994:8). This involves running the software in a special environment with test data or in the actual environment with actual data (Daich et al., 1994:8). Dynamic tests are only as effective as the data used during testing (Daich et al., 1994:8). Usually, defects found during dynamic analysis are detected only when they affect the output (Daich et al., 1994:8). Another definition of dynamic testing is provided by Martyn Ould and Charles Unwin in the book, *Testing in Software Development*, in which they state that dynamic testing "differs from static testing in that verification is done by executing that deliverable [software] rather than by comparing one specification with another" (Ould and Unwin, 1986:13).

Debugging is a two-part process in which software is tested to determine the exact nature of a defect and then repaired to fix the error (Myers, 1979:130). Beizer defines the purpose of debugging as "to find the error or misconception that led to the program's failure and to define the program changes that correct the error" (Beizer, 1984:5). Testing and debugging usually account for 50% to 80% of the costs to develop the first working version of a software package (Beizer, 1984:1).

Unit testing is defined by Royer as testing "to ensure that the software unit – which we have intentionally not defined – satisfies its functional specifications" (Royer, 1993:132). He notes that there is some disagreement as to the definition of a "unit", as often the terms "module" and "unit" are used interchangeably. Royer adds that a unit consists of all of the procedures and functions needed to implement a solution to a specific problem in a single entity (Royer, 1993:132). Ould and Unwin define unit testing as "testing that a module correctly implements its design and is ready to be integrated into the subsystem or full system" (Ould and Unwin, 1986:11). The STSC notes that unit testing often occurs by the developer in isolation (Daich et al., 1994:8). Often for unit testing, stubs, drivers, and test data are developed to test these modules in isolation.

Integration testing is a step up from unit testing. During integration, several modules are combined and tested, primarily in the interfaces between modules. The STSC defines integration testing as testing "performed to determine how the individual modules making up a subsystem component perform together as an integrated unit" (STSC, 1996:14-59). Integration testing may also include the testing of the integration of

software and hardware (Daich et al., 1994:8). Ould and Unwin define integration testing as "testing that the components of the system correctly combine to form a complete system of subsystems" (Ould and Unwin, 1986:10).

System testing is the testing of the complete hardware and software system to ensure that the system meets the specified requirements (Daich et al., 1994:8). System testing is usually performed after integration testing. System testing is performed using the complete set of test cases, which are developed from the system specification. Ould and Unwin note that system testing often occurs with integration testing (Ould and Unwin, 1986:10).

Qualification testing, also known as acceptance testing, is testing performed on the system to verify that the system meets the needs of the user. The STSC notes that generally, a subset of test cases is used to demonstrate key functionality to the users at the installed site (Daich et al., 1994:8). Ould and Unwin define qualification testing as "verifying that the delivered computer system meets the user requirements as expressed in the System Specification" (Ould and Unwin, 1986:10).

Real-Time Non-Intrusive Monitoring

Real-time processes are defined by Bernhard Plattner as "processes whose correctness depends on their behavior with respect to time (i.e., time in the sense of real-world time)" (Plattner, 1984:756). Most real-time processes are used to either control or analyze real-world processes (Plattner, 1984:756) It is important to note that he states "the pace of execution of a real-time process is therefore not determined by internal

criteria, e.g., the execution speed of the underlying processor, but by the timing constraints imposed by the real-world process" (Plattner, 1984:756). Thus, a real-time process uses real-world time as a parameter, and is dependent on real-world time as much as it is dependent on other input values.

An intrusive system is one in which the software has been modified to allow the monitoring of internal data elements (Daich et al., 1994:8). A non-intrusive system is one in which special hardware monitors are attached to the system to monitor the internal data elements (Daich et al., 1994:8), and the software remains unmodified from its operational state (Fryer et al., 1995:1).

In their paper "Real Time Non-Intrusive Monitoring and Embedded Systems Simulation", Fryer, Vansteenkiste, and Van Campenhout note that there are two key concepts for RTNI monitoring:

> i) The timing of the system processes and events must be entirely insensitive to the 'on' or 'off' status of the hardware instrumentation. No time perturbation is allowed – the instrumentation must operate throughout the full speed gamut of the production system. Following the notions of (Kishon, Hudak, and Consel 1991), the monitoring must be "consistent – i.e. it is not possible for a monitor to change program behavior."
>
> ii) No changes to the operational code are allowed to facilitate the instrumentation. (Fryer et al., 1994:3)

The first concept is that of "real-time" processing, while the second concept is that of "non-intrusive".

In the paper, "The 'Instrumentation Memory': A Measuring Feature for Simulation of Real Time Embedded Systems", Fryer, Vansteenkiste, and Van

Campenhout state that there are two requirements for RTNI instrumentation of software that apply to the above two concepts. The first is:

> The instrumentation mechanisms must be transparent to the behavior of the software. Transparency has typically been deemed adequate if the flow of addresses in program execution is identical and if the time relationships of all software detectable events are equivalent (i.e. interrupts, sequence of procedures run, time spent in a process, etc.). (Fryer et al., 1995:1)

The second requirement for RTNI instrumentation is "no changes to the operational code are admitted to support instrumentation unless they are also part of the operational code" (Fryer et al., 1995:1).

Role of RTNI Monitoring in Software Testing

The role of RTNI monitoring in software testing is that of dynamic analysis. The goal of RTNI monitoring is to provide an analysis of the software as it operates in the actual hardware, in the real-time sequence of events, and without modification to allow for the monitoring. Unlike other dynamic analyses, RTNI monitoring does not compare an output with an input for its error detection, but rather errors are detected as they occur (Daich et al., 1994:8). Errors are detected as they occur, because the hardware attached to the system monitors the states of internal data elements.

RTNI monitoring can be used for testing in any phase. However, Robert Glass has suggested distinct test phases for real-time systems in his paper "Real-Time: The 'Lost World' of Software Debugging" (Glass, 1980:266).

The first phase Glass suggests is "Host-Host". In this phase, the testing is done only on the host computer. The purpose of this phase is to eliminate programming errors

from code. He notes that usually this testing occurs as unit testing of "small program components", as full system testing usually does not occur on the host computer because of the complexity of simulating a real-time environment on the host computer (Glass, 1980:267).

The second phase of real-time testing is "Target" (Glass, 1980:267). The testing of the software is executed on the target computer. Typically, the only tools available for a tester are a recorder of output for "post-mortem review" and the real-time environment simulator which allows the tester to review the input parameters (Glass, 1980:267). The use of RTNI monitoring allows testers to monitor the execution at several points and detect errors as they occur, rather than after a test case has failed.

Glass notes that there are different "flavors" of target testing (Glass, 1980:268). The first is the software unit testing that can occur when only the initial target hardware is available. The second is the use of an environment simulator with the target, with which integration testing can be performed. The third is the removal of the environment simulator and the full system testing in the actual operating environment.

Previous Studies

This section will briefly describe two previous studies of the life-cycle cost impact of RTNI monitoring by ITCN, a developer of RTNI monitoring technology.

The first study by ITCN focused on the F-18 A/B Software Support Activity (SSA) at the Naval Air Warfare Center at China Lake in July 1990. The study, which hasn't been independently verified, showed that the total life-cycle benefit of using RTNI

monitoring was $2,586,182 (ITCN 1, 1990:1). Also, the study claimed that "50% of Software Trouble Reports (STRs) resolved faster" with RTNI monitoring (ITCN 1, 1990:1). The study also reported that RTNI monitoring reduced flight testing delays, which also resulted in a savings of "millions of dollars" (ITCN 1, 1990:1). One assumption this report makes, though, is that the use of RTNI monitoring will not only lower the time to detect errors, but also lower the time to correct them. For example, it notes that it took four engineers one week to isolate a bug, and then nineteen engineers two weeks to fix the problem (ITCN 1, 1990:3). However, with RTNI monitoring, they state the error "would have been found in minutes, permitting a fix in hours" (ITCN 1, 1990:3). The report provides no basis for the assumption that the fix could occur in hours versus weeks with RTNI monitoring.

The second study by ITCN focused on the systems integration phase of the VHSIC 1750 insertion research and development (R&D) project for the F-111 at the Sacramento Air Logistics Center at McClellan AFB in August 1990. One result of this study was that the systems integration phase was reduced from 17 months to 12 months (ITCN 2, 1990:1). Another was that $200,000 in net project savings occurred during the first year (ITCN 2, 1990:2). This study was also not independently verified.

III. Methodology

Chapter Overview

This chapter details the steps needed to collect and analyze the necessary data to support the hypothesis that the use of RTNI monitoring technology will result in cost savings over the life-cycle of a program. Since no financial data exists with which to compare the savings already realized with the use of RTNI monitoring in existing programs, a structured interview was constructed from which a general idea of the impact of the use of RTNI monitoring technology could be determined. The results of this interview were used to theoretically determine what potential impact the use of RTNI monitoring could have on similar programs.

Initial Investigation

Since the study aims to determine the life-cycle cost impact of the use of RTNI monitoring, the first step is to estimate life-cycle costs. Barry Boehm suggests several methods in his book, *Software Engineering Economics* (Boehm, 1981:329).

The first method of cost estimation examined was the algorithmic model. This was quickly ruled out for use in this project for two reasons. First, as Boehm suggests, since these models are calibrated to previous projects, it is not known with certainty if these projects are representative of future projects using new techniques, such as the examination of the impact of RTNI monitoring (Boehm, 1981:333). Second, Boehm also

points out that the model will be inaccurate if poor sizing inputs and inaccurate cost driver ratings are used.

The next method of cost estimation examined was estimation by analogy. As Boehm states, "estimation by analogy involves reasoning by analogy with one or more completed projects to relate their actual costs to an estimate of the cost of a similar new project" (Boehm, 1981:336). The initial methodology for this study was to examine two similar block changes at one of the Air Logistic Centers (ALC), with one block developed using RTNI monitoring and one block that was developed without the use of RTNI monitoring. Initial searches for data found that the ALCs do not track costs to a level at which determinations could be made about the use of specific tools. Also, the differences in block change upgrades were too significant to attribute any variations in spending to one specific attribute. As Boehm suggests, a problem with the use of estimation by analogy is that "it is not clear to what degree the previous project is actually representative of the constraints, techniques, personnel, and functions to be performed by the software on the new project" (Boehm, 1981:336).

The third method of cost estimation examined was expert judgment. This method uses the past experience of experts who, based on their experience and understanding, can project an estimate of cost. As Boehm notes, a strength of expert judgment is the ability of the expert to "factor in the differences between past project experiences and the new techniques" (Boehm, 1981:333). Boehm also points out that a weakness of expert judgment is that the results may be influenced by the bias of the experts surveyed. To eliminate bias, Boehm suggests obtaining estimates from several experts. As a result, a

structured interview of expert judgment was conceptualized to determine the general impact that RTNI monitoring had on various programs.

Interview Instrument

The participants were determined to be existing users of RTNI technology. These users were determined through the use of "snowball sampling". Snowball sampling as described by Donald Cooper and C. William Emory in their book, *Business Research Methods* (Cooper and Emory, 1995:230), is identifying a group that has a particular characteristic, who in turn, identify others who also possess the additional characteristics. As a result, a sample set is constructed with which all members of a sample set possess the same characteristic. In this case, the characteristic was users of RTNI technology.

The initial list came from customer listings provided by companies that market RTNI test equipment. These customers then additionally provided the names of other users of similar equipment, as well as providing other names of additional sources of RTNI test equipment. Since snowball sampling was used to identify the users of RTNI testing, there is no way to determine the actual population of all RTNI users.

Initially a survey was conceived to be given to the users of RTNI testing. However, it soon became apparent that the small number of current users of RTNI test equipment would not allow for enough data from which accurate conclusions could be drawn. Thus, the survey evolved into a structured interview.

Nominal Group Technique

The initial design for the structured interview was the use of the nominal group technique (NGT). This technique was chosen, because it controls the interview to avoid the potential for one individual to dominate the group discussions, and the fact that the responses from NGT can be converted to numerical data, allowing for quantitative and qualitative data (Zimmerman and Muraski, 1995:108).

The NGT requires all participants to meet in one conference room. Because the users of RTNI monitoring are located at several locations across the country, it is impossible to get them all together in one room to discuss the problem. Thus, this methodology used a modified NGT using phone and electronic mail interviews.

The first stage of the NGT is to define the question under study (Zimmerman and Muraski, 1995:108). The question has to be defined narrowly enough to limit the responses, but openly enough to allow for diverse responses (Zimmerman and Muraski, 1995:108-9). NGT sessions generally use only one question (Zimmerman and Muraski, 1995:108). In this study, the question was: "In what areas do you see the use of real-time non-intrusive monitoring having an impact on life-cycle costs, whether positive or negative?"

The second stage of the NGT is to select participants (Zimmerman and Muraski, 1995:109). According to Zimmerman and Muraski, 6 to 12 knowledgeable participants should be selected to represent the subject under study (Zimmerman and Muraski, 1995:109). The initial searches for knowledgeable users of RTNI monitoring turned up around 13 users, of which 6 consented to participate in the study. They are representative

of the Air Force, the Navy, and NASA, and use RTNI monitoring in the development and support phases of the life-cycle. Each participant represents a different program, so as to allow diverse opinions, while each system can be represented on an equal basis. They have been told of the area of the study, but not the research question itself, as the question could bias responses before the study begins (Zimmerman and Muraski, 1995:109).

The third stage of the NGT is to begin the session (Zimmerman and Muraski, 1995:111). In most cases, the researcher gathers the participants in a room and addresses them about the study. In the case of this study, an electronic mail message (Appendix A) was sent to tell the participants about the study and how the process will be conducted. In a subsequent electronic mail message (Appendix B), all participants were provided with the research question. Each participant was to take a few minutes to list a number of responses to the question. They were then to respond via electronic mail with the responses to the administrator. It was important that the participants only communicate with the administrator and not with one another (Zimmerman and Muraski, 1995:112). Each participant was directed to reply directly to the administrator within 48 hours of receipt of the question, and they were cautioned not to respond to any other recipients of the message. By setting the time limit at 48 hours, the participants were allowed to answer the questions at their convenience, yet still allow the minimal brainstorming time used in the NGT (Zimmerman and Muraski, 1995:112).

The fourth stage of the NGT is to list nominations of responses (Zimmerman and Muraski, 1995:112). In most cases, each participant takes turns nominating responses from their list. The nominations are listed on large panels of newsprint paper, and are

written in the participant's wording. However, in this study, the nominations were compiled by the administrator in a document that was a collection of all of the individual experts' nominations. The responses were listed in a random order, so that each participant's responses were not grouped together. This process continued until all responses were nominated by the participants (Zimmerman and Muraski, 1995:112). This was accomplished by collecting all of the responses and compiling them in the single document mentioned above.

The fifth stage of the NGT is to discuss the items, clarify items, and eliminate duplicates (Zimmerman and Muraski, 1995:112). This was accomplished by sending the nomination list to all participants. They then replied with an electronic mail to all participants as to whether or not they understood all of the items and how duplicates should be listed. At this point, the participants were also able to make additional comments and discuss any of the issues presented in the list. As a result, everyone in the group was a recipient of each electronic mail response. The group was given one week to make comments on the list of nominations. The administrator then collected all of the responses and clarified questions and eliminated duplicates as per the participants' wishes.

The sixth stage of the NGT is to select individual items (Zimmerman and Muraski, 1995:113). Each participant was electronically mailed the revised list of response items. Each participant listed the five items most important to him or her.

The seventh stage of the NGT is to rank the selected items (Zimmerman and Muraski, 1995:113). Each participant ranked his or her list of selected items. An

"outside-in" technique was used (Debold, 1997:WWWeb). The "outside-in" technique begins with ranking the most important item of the five with a "5", and then the least important item with a "1". The second stage of the "outside-in" technique is ranking the second most important with a "4", the second least important with a "2", and the remaining item with a "3". The participants then electronically mailed their responses to the administrator.

Item Analysis

The responses were compiled and the ratings were added. Statistical analysis was performed on each item, which included the percentage of the participants that selected that response, the minimum, the maximum, and the average value of each item.

The results from the NGT are important, because they contain two major elements (Zimmerman and Muraski, 1995:114). The first major data element is that the items that the group felt were the most important are shown. The second data element is the weights that participants assigned to each item. Zimmerman and Muraski state that "the weights assigned give an importance value to the items" (Zimmerman and Muraski, 1995:114). Thus, not only can the data be analyzed to determine which items were selected by the participants, but also how the participants rated the items in importance to one another. As a result, the results showed the cost drivers and areas that the experts perceived as having an impact by RTNI monitoring on life-cycle costs. Each of these areas also has a ranking, so that it can be determined to how important each area is to total life-cycle costs.

IV. Findings

Chapter Overview

This chapter summarizes the results of the expert judgment of the life-cycle cost impact of the use of real-time non-intrusive (RTNI) monitoring by government system program offices from the Department of Defense and the National Aeronautics and Space Administration (NASA). The data are provided for those results obtained in each stage in the Nominal Group Technique (NGT), which was used as the instrument to obtain an expert opinion of users of current RTNI monitoring technology. The data is also shown with statistical analysis to quantify the expert opinions provided during the NGT.

Nominal Group Technique Results

The results from each stage of the Nominal Group Technique are listed by stage in the following sections. The actual outputs of each section can be seen in the appendices to this report. In the appendices, the items are shown as they were provided to or received from the participants. The only cases of editing in these sections are in areas where the response contains information that might compromise the anonymity of the respondent or where a real-time non-intrusive monitoring instrument is referred to by name. This report does not make reference to the names of specific tools, so as to not show a bias towards any particular RTNI monitoring system.

Stage One: Definition of Question

In this study, the question was developed to limit responses to areas in which real-time non-intrusive monitoring has had impacts in life-cycle costs. The question was designed to allow the experts to respond with not only the potential benefits of real-time non-intrusive monitoring, but any drawbacks that real-time non-intrusive monitoring causes as well. Thus, the question defined in this study was: "In what areas do you see the use of real-time non-intrusive monitoring having an impact on life-cycle costs, whether positive (decreasing costs) or negative (increasing costs)?" Providing a definition of what is meant by positive and negative life-cycle costs (i.e., decreasing versus increasing costs) removed ambiguity from the question.

The question was developed and presented to the participants in the "response worksheet" form described by Zimmerman and Muraski, which provides a background to the problem before the presentation of the question (Zimmerman and Muraski, 1995:109). The research question presented to the respondents is shown in Appendix B.

Stage Two: Selection of Participants

Initial searches for participants through the Department of Defense found six programs that currently use real-time non-intrusive monitoring. NASA also uses real-time non-intrusive monitoring in the development of their deep-space probes. These programs were contacted to determine if they wanted representation in this study. Of these programs, five defense programs and one NASA program agreed to provide representation.

The characteristics of the programs were mixed. Three of the programs were Air Force, two of the programs were Navy, and one was NASA. Two of the programs used real-time non-intrusive monitoring in the development phase of the life-cycle, while four used it during the support phase.

One person from each program was named to represent their program as the point of contact. Each of the six programs received equal representation in the study. All of the participants were expert users of real-time non-intrusive monitoring in current programs. Each point of contact was allowed to collaborate with their program office on generating the responses in the study. The responses were true representations of their program, rather than solely personal opinion.

Each participant was provided an initial notification, which explained how the study would be conducted and a brief overview of the process with which they would be providing input. The initial notification to respondents can be seen in Appendix A.

Stage Three: Beginning the Session

The participants were provided the research question shown in Appendix B. They were given 48 hours to review the question and list answers. At the end of the time, they were to mail their responses to the test administrator.

Several respondents were unable to complete the responses within the 48 hour period due to scheduling conflicts. However, all respondents had provided input within one week from the receipt of the study question. The initial responses from participants can be seen in Appendix C.

Stage Four: List Nominations

The responses from the respondents were listed by similar areas. Those areas were:

- testing costs
- complexity
- software size
- reliability
- maintenance costs
- hardware procurement costs
- training costs

Listing in this fashion showed the areas in which there was some duplication and or ambiguity. This listing allowed the participants to view all responses that apply to a particular area and decide upon a specific idea that encompasses the complete idea trying to be conveyed. The ideas were listed in no particular order and were arranged to show no apparent importance or ranking of the items. The items were listed with letters, to show that there was no ranking of ideas at this point, which numerical listing can sometimes imply. The items were also listed in the respondents' own words.

The compiled responses were then provided to the participants to be used as items for discussion. The items are listed in Appendix D.

Stage Five: Discussion

Each participant was provided a copy of the items for discussion as listed in Appendix D. To allow for anonymity by the participants, it was decided that all participants would reply with their input to the administrator, who in turn would pass it along to all other members of the study. The administrator would edit the responses to remove any identifying information from the participant's response, so as to allow non-attribution to remain in effect.

The time for discussion was one week. During this time, only a few participants provided additional information. The results of the discussion appear in Appendix E. As with the listing of initial responses, the items appear in the words of the participant.

There are a few reasons for the lack of response during this phase. Some members were away from work during this time, so they could not actively participate. Another reason is that some members felt no additional discussion of items was necessary.

As administrator, I provided some additional areas that could be addressed during this stage, as well as provided some guidance on items that could be combined to eliminate duplicates.

Stage Six: Select Individual Items

Following the lack of response during the discussion, and in particular the failure to eliminate duplicates, some of the wording was changed and some of the ideas were combined to allow for easier interpretation of the responses during this final phase. Since

the final two phases involve selecting and ranking the most important ideas, each section should have only one idea. In that case, the idea is ranked according to other ideas, and not ranked according to its particular wording.

The final list of ideas included:

- Reduction of debug time
- Sharing of testing time among engineers
- Reduction of regression testing time
- Additional testing data that was unavailable before
- Increased amount of dynamic testing
- Reduction in test case development time
- Increase in testing reliability
- Reduction in time needed for integration
- Increase in training costs
- Increase in overall system reliability
- Increase in hardware procurement costs
- Reduction in maintenance costs

Each of the above areas is detailed with an explanation in the next section of this report, entitled "Life Cycle Impact by Area". The final list as sent to the participants is shown in Appendix F.

Stage Seven: Ranking Selected Items

In the final phase of the nominal group technique, the participants rank order their list of ideas. The final rankings by the participants is listed in Appendix G.

Life-Cycle Impact by Area

During the nominal group technique, the participants listed several ideas which can be seen in the appendices. Most importantly, the participants listed several areas in which it could be seen that RTNI monitoring has an impact on life-cycle costs. Of the twelve areas identified by the users, ten of these areas were positive (decreasing cost) areas in which there are potential cost savings, and two of them were negative (increasing cost) areas in which there are potential additional expenses.

The first positive area identified by the respondents was the area of debug time. The participants noted that before RTNI monitoring, the only way to find defects in the code was to perform a step-by-step analysis of the code, which would allow the tester to find the area in which the code was in error. The errors are frequently timing errors that are not apparent until the code is executed during real-time. Testers could not examine the internal variables of the code during these dynamic tests, so they were unable to determine the cause of an error, but rather only conclude that an error had occurred. According to the participants, since the advent of RTNI monitoring, developers can now see the internal settings of the code and debug the code as it is actually executing. Another idea in this area, according to the study group, is that by finding the defects

earlier, any follow-on tests or corrections do not have to wait for the results, so that the overall debug and correction process is much quicker.

A second positive area identified by the study participants is the idea of sharing time among engineers. Since RTNI monitoring requires two computer systems, one which executes the software and one which monitors it, the engineers can execute the code on the target system, and then view the results on a separate system, according to the group. This frees up the target system for additional testing by other personnel, which could decrease overall testing time or allow more testing in a given time period.

A third positive area identified by the study participants is reduced regression testing time. According to the group, since the memory of the target system is visible to RTNI monitoring, automated tests can be developed to take advantage of this visibility. These automated tests can be used to test only those parts of the code that have changed since the last time the test was performed.

A fourth positive area in which RTNI monitoring can affect life-cycle costs is by providing additional testing data that was not available before the advent of RTNI monitoring. The group contributed that examination of the code by engineers in the RTNI environment could allow engineers to discover important information about the behavior of the software. This information about the behavior could be applied during code development, which could lead to additional savings in life-cycle costs.

A fifth positive area identified by the study participants is the area of increased amount of dynamic testing. According to the study group, engineers can ensure that the address calls and jumps are going to their proper address by capturing the number of time

an address is accessed. Before RTNI monitoring, this required static analysis. After RTNI monitoring, it is easier to check for defects during code execution, which leads to more dynamic testing.

A sixth positive area is the reduced time of the test case development. According to the group, less time is needed to find the cause of a defect when using RTNI monitoring, since new tests can be developed quickly to narrow the focus of an investigation.

A seventh positive area identified by the members of the study is an increase in the reliability of testing. According to the group, RTNI monitoring allows for the recreation of a problem in the actual real-time environment, so tests are performed in an actual real-time environment, rather than in a simulated environment with different timing.

An eighth positive area is the time saved in integration. According to the group, RTNI monitoring allows the various different models of complex systems to be viewed at various points within the system, instead of strictly at the input and output levels.

A ninth positive area is an increase in overall system reliability. The participants identified that since more code can be tested in less time, overall system reliability can be increased.

A tenth positive area is the reduction of maintenance costs. The participants identified that the maintenance cost attributable to rework can be reduced. Also, they added that RTNI monitoring can aid during software patching and the synchronization of certain functions.

One negative area is an increase in the training costs. The group contributed that although training will be required for all personnel using RTNI monitoring equipment, there should be long term savings, since training is a one-time expenditure, unless there are additional upgrades to RTNI testing tools.

A second negative area is an increase in the costs of procuring RTNI monitoring hardware. According to the group, there are not only the costs of procuring RTNI monitoring devices, but also the media storage for collecting the testing data.

Analysis of Results

The rankings of the participants provide two important data elements. The first data element provided is frequency. The participants selected the items that they felt were the most important. The second data element is the ranking. By ranking the five selected ideas with one another, each participant assigned a weight to each idea, with a "5" being the most important and a "1" being the least important. Table 4-1 shows the respondents' rankings of all of the items.

Table 4-2 shows the number of responses each item received. The table lists the items ordered by frequency. This is important, since the items with a higher percentage are those items that were chosen by more members of the group. The percentage denotes the percentage of the group choosing that particular item as one of the five most important items. The frequency denotes the number of participants that chose that particular item. It is important to note that the item "debug time is reduced" was selected as important by all members of the study. It is also important to note that the ideas of

"debug time is reduced", "increased amount of dynamic testing", and "testing reliability is increased" were chosen by over half of the participants. The items with higher frequency would have more applicability to more programs than items that were only chosen by one participant. The items chosen by one participant might be specific items that apply only to that particular program.

Table 4-1. Rankings of All Items By Participants*.

	Respondent					
Item	**1**	**2**	**3**	**4**	**5**	**6**
Debug time is reduced	4	5	4	5	5	5
Sharing of testing time among engineers	5			3		2
Reduced regression testing time						
Additional testing data not available before		4				
Increased amount of dynamic testing		3	1	4	3	
Test case development time reduced	2		5	2		
Testing reliability is increased		2		1	2	4
Time saved in integration	3					
Training costs increased		1				
Overall system reliability increased			2		4	3
Hardware procurement costs increased						1
Maintenance costs reduced	1		3		1	

* Legend: 5 = Most important ... 1 = Least important

Table 4-3 shows the statistical data obtained for each item, with the items arranged by average ranking. The items with the higher averages were determined to be the most important to the group. It is important to note that some of these items were ranked high by a single participant, which gives them a high overall ranking, while the item may not be applicable to all programs. For example, the item "additional testing data not available before" has an average weight of 4, giving it the second highest average weight, yet it was only selected by one participant. The maximum and minimum values are presented to show the range of weights assigned by the participants for each

item. The ordering of data by both frequency and average weight is provided in Table 4-4.

Table 4-2. Results Ordered by Frequency of Items Selected by Participants.

Item	Percentage	Frequency*
Debug time is reduced	100	6
Increased amount of dynamic testing	66.67	4
Testing reliability is increased	66.67	4
Sharing of testing time among engineers	50	3
Test case development time reduced	50	3
Overall system reliability increased	50	3
Maintenance costs reduced	50	3
Additional testing data not available before	16.67	1
Time saved in integration	16.67	1
Training costs increased	16.67	1
Hardware procurement costs increased	16.67	1
Reduced regression testing time	0	0

* - Total number of participants is n=6

Table 4-3. Results Ordered by Mean with Minimum and Maximum Values.

Item	Average (Mean)	Minimum	Maximum
Debug time is reduced	4.667	4	5
Additional testing data not available before	4	4	4
Sharing of testing time among engineers	3.333	2	5
Test case development time reduced	3	2	5
Overall system reliability increased	3	2	4
Time saved in integration	3	3	3
Increased amount of dynamic testing	2.75	1	4
Testing reliability is increased	2.25	1	4
Maintenance costs reduced	1.667	1	3
Training costs increased	1	1	1
Hardware procurement costs increased	1	1	1
Reduced regression testing time	0	0	0

Table 4-4 shows the data arranged by considering both of the previous data elements, frequency and weight, to provide an overall total weight. The total weight is

the sum of all rankings of each item by all of the participants. Thus, it is the combination of both the average ranking of the item as well as the percentage of participants that chose the item. The total weight can also be calculated by multiplying the average by the percentage, and then by six, which is the total number of participants in the test. The items that are higher on the list are those that were judged to be most important by a majority of the group.

Table 4-4. Results Ordered by Total Weight (Rank and Percentage).

Item	Average	Percentage	Total Weight
Debug time is reduced	4.667	100	28
Increased amount of dynamic testing	2.75	66.67	11
Sharing of testing time among engineers	3.333	50	10
Testing reliability is increased	2.25	66.67	9
Overall system reliability increased	3	50	9
Test case development time reduced	3	50	9
Maintenance costs reduced	1.667	50	5
Additional testing data not available before	4	16.67	4
Time saved in integration	3	16.67	3
Training costs increased	1	16.67	1
Hardware procurement costs increased	1	16.67	1
Reduced regression testing time	0	0	0

Summary

According to the expert opinion of the group, the biggest benefit of the use of real-time non-intrusive monitoring is that debug time is reduced. This finding is significant, because not only did every member of the group consider this one of the most important areas in their program's experience with RTNI monitoring, but also, every

member ranked it as either the most important or second most important area in their program.

Two additional areas were important to over half of the group. The first area that was important as a potential benefit is the increased amount of dynamic testing that can be performed compared to the amount of other testing. Another area that was important to a majority of the programs was an increase in the reliability of the testing. The quality of data collected in the testing was improved, resulting in a higher reliability of the testing to actually determine the causes of defects in the software.

There were several areas that had above average values of importance. In addition to reduced debug time, other time savers that were deemed to be important were the sharing of testing time among engineers, the reduced time needed for test case development, and the time saved during integration. The increased amount of dynamic testing and the increase in system reliability were also judged to be important. Finally, the additional testing data that is available to developers from RTNI monitoring was judged to be important.

Overall, for the selected programs, the biggest benefits of RTNI monitoring are the reduction in debug time, the increased amount of dynamic testing, and the sharing of time amongst engineers. All of these factors can result in shorter testing periods. These areas were ranked among the most important items to a majority of the group, which in turn, implies that the application of RTNI monitoring to future programs is most likely to have the largest impact in these areas.

V. Conclusions and Recommendations

Chapter Overview

In this chapter, the conclusions supported by the research are presented. Additionally, recommendations for future research are presented. These recommendations include not only ways in which this study could be improved for use in the future, but also ways in which future programs considering the use of real-time non-intrusive (RTNI) monitoring technology could determine its applicability to their programs.

Conclusions

The use of RTNI monitoring on programs that require the development and support of software-intensive embedded systems has definitely had a major impact on life-cycle costs. The major area in which this type of testing has had the largest impact on current programs is the reduction of debug time. RTNI monitoring provides a capability that cannot be provided through the use of other tools.

The participants in this study have identified various areas in which real-time embedded systems provide challenges to the test engineers. Some of these areas are a "scaling problem in a fixed point calculation", "jump addresses and call addresses", and general timing problems between systems. According to the expert users of RTNI monitoring, before RTNI monitoring, an engineer would have to spend long periods of time examining the code line by line to determine where the error was located. Using

RTNI monitoring, an engineer can see the exact spot where an address might be incorrect or a calculation fails during real-time testing.

Using static analysis to find these errors was often time consuming; the error would become apparent during dynamic testing, but it would require static analysis of the code to determine the exact location of the defect. Additionally, other tests often cannot be performed until all previous tests were complete and the problems are resolved, because of the critical nature of the embedded systems.

Using RTNI monitoring, an engineer can quickly identify the exact location where a failure occurs. This is accomplished by allowing the tester to view the execution of the software on a separate system. This view of the software execution allows the tester to obtain the data that was previous unavailable to the testers of real-time embedded systems, such as specific memory addresses of locations being accessed by the software.

Another major area in which the use of RTNI monitoring can have a strong impact on life-cycle costs is through an increased amount of dynamic testing. During a fixed testing period, more dynamic testing can be performed, in which more defects could be detected.

A final major area in which the use of RTNI monitoring can have a strong impact on life-cycle costs is allowing the sharing of testing time among engineers. Engineers can execute tests on the target system, then review the results on a separate system. By reviewing the results on a separate system, it opens the target system for additional tests by others. During a fixed testing period, more tests could be performed, since different groups could test their code at various times and review the results later.

According to Frank Wellman, in his book *Software Costing*, there are factors that influence whether a perceived benefit from the use of a system is actually achieved (Wellman, 1992:134). These factors include:

1. a change in market conditions that might result in an increase or decrease in the rate of return of expected benefit
2. a change in operational conditions, ..., leading to a change in the use of the system and a change in the rate of return
3. a failure on the part of management to ensure that the system is effective in use and that the maximum corporate benefit is derived from the information handling capabilities of the software. (Wellman, 1992:135)

He notes that the first two factors are often outside of managerial control, but that the third factor can be influenced by management and is often the reason that "systems never achieve their intended potential in operational use" (Wellman, 1992:135).

Thus, although the expert users of RTNI monitoring have identified that the use of RTNI monitoring has had an impact in their programs, the potential benefits to future users will vary, depending on how RTNI monitoring is implemented by those program managers. Each program considering the use of RTNI monitoring should conduct a cost-benefit analysis to determine if the potential benefits provided will outweigh the costs of acquiring and implementing RTNI monitoring tools into their test programs. Procedures for conducting cost-benefit analyses for software systems can be found in books such as *Software Costing* by Frank Wellman (1992) and *Managing the System Life Cycle* by Edward Yourdon (1982).

Recommendations

A detailed analysis of the current savings from RTNI monitoring could not be accomplished, because there was no data available with a detailed cost breakdown as to how problems were resolved using specific tools and techniques. As a result, there was no means with which it could be determined to ascertain the role that RTNI monitoring has had in impacting life-cycle costs except through the use of the expert users who conducted the software testing. Future studies could track this data over a period of time, and analyze the potential savings to a system. This would provide actual quantifiable examples of the specific impact on life-cycle costs directly attributable to RTNI monitoring.

Another recommendation is that programs considering the use of RTNI monitoring could evaluate its applicability to their program by experimenting with various test methods and determining which methods work the best. This could be accomplished by "seeding" a program with various types of defects and determining how well RTNI monitoring compares with other methods in detecting the defects. Such an experiment could also be used as a follow-on study to determine which defects are best detected with RTNI monitoring.

Summary

The use of RTNI monitoring on future programs will definitely impact the total life-cycle costs. The major area of life-cycle impact is the benefit of reduced debug time. Additional areas that can reduce testing time that are improved by the use of RTNI

monitoring are increased amounts of dynamic testing and the sharing of testing time among engineers. The use of RTNI monitoring has the strong potential for reducing a program's overall testing time, as can be shown by its use on existing programs. However, these cost savings can only be realized with the proper implementation and commitment by management. Simply procuring a new software testing tool itself will not have an impact on a software program, but rather the complete integration of RTNI monitoring into the overall testing program and methodology could result in long-term savings over the program's life-cycle.

Appendix A: Initial Notification to Participants

Dear Participant:

Real-time non-intrusive (RTNI) monitoring has been used in a few programs throughout the Department of Defense to test software on real-time embedded systems. However, it is often very expensive to add RTNI capabilities to avionics computers. Program offices are reluctant to fund RTNI monitoring enhancements because of the expense involved. As part of my requirements for completion of my graduate program, I am completing a thesis study that will analyze the life-cycle cost impact of RTNI monitoring.

Since there are so few programs that currently use RTNI monitoring, it is necessary to use a panel of experts to determine what impact, if any, the use of this technology will have on life-cycle costs. This study will be conducted using a modified form of the Nominal Group Technique (NGT) expert judgment interview. This form of the NGT will take place via e-mail, and will consist of seven participants. All responses will be anonymous, and there will be a non-attribution policy. This study will enable us to determine the most important issues to the group by collecting ideas and then achieving a consensus among the group. The procedure for the study will be as follows:

1. A specific question will be issued to all participants via e-mail.

2. All participants will respond via e-mail to myself with responses to the question. The responses should be clear and brief, yet no classified or other controlled data should be provided as part of the response. The responses should not be sent to all members of the group, but only the administrator. Only 5 to 10 minutes should be used

to list responses to the questions. Since all of the participants are from different organizations, feel free to discuss the question with other members of your organization, but please do not discuss them with the other participants of the study.

3. The entire list of responses from all participants will be compiled and sent to the entire group. All participants can submit an e-mail to the group with any comments that they have about the nominations. Any e-mail responses to any of the items should be carbon copied to all other members of the study. Participation in the discussion is highly recommended. The discussion will continue for a week.

4. A final list of nominated items will be e-mailed to each participant. Each participant will then respond to myself with a list of the five most important items to them and their organization. The five items will also be rank ordered by number from 1 to 5, with 5 being the most important item and 1 being the least important item.

This study is important to determine the impact that the use of RTNI monitoring has on the total life-cycle costs of a program. This study is being conducted by the Air Force Institute of Technology (AFIT) through the Graduate School of Logistics and Acquisition Management. Any questions, comments, explanations, or suggestions are welcomed and will be considered in the final report. Please address any questions to Captain Michael Lewis, AFIT/LAA at DSN 785-7777 x2139 (military) or (937)255-7777 x2139 (civilian) or at "mlewis@afit.af.mil" (e-mail). Additional questions can be answered by the academic advisor of this thesis, Professor Dan Ferens, AFIT/LAS at DSN 785-1210 (military) or (937)255-1210 (commercial) or at "dferens@afit.af.mil".

Thank you for your time and consideration in participating in this important study. If you know of anyone else who would like to participate in this study, feel free to have them contact myself or Professor Ferens.

Appendix B: The Research Question as Provided to the Participants

Dear Participant:

There are several cost drivers that can affect the life-cycle costs of a software program. Some examples of these drivers are reliability, training, and software size and complexity. The use of a new tool, such as real-time non-intrusive (RTNI) monitoring, can cause changes in several of these areas, which can have an impact on total life-cycle costs. Identifying all areas in which RTNI monitoring can affect life-cycle costs is the first step in identifying the life-cycle cost impact.

For the next several minutes, please think about the following question. Then, please briefly list your answers. Please e-mail your list of answers by Monday, July 21 to "mlewis@afit.af.mil".

Question: In what areas do you see the use of real-time non-intrusive monitoring having an impact on life-cycle costs, whether positive (decreasing costs) or negative (increasing costs)?

Thank you for your time and cooperation in this study.

Appendix C: Initial Responses from Participants

Respondent 1

Positive Impact on Life-Cycle Costs:

Debug time will be reduced because code execution would be done in real time instead of stepping through one instruction at a time.

Engineers may be able to execute code on the target system and then examine the results on a separate system at their leisure allowing more engineers to share the target system.

Examination of code execution will probably allow engineers to discover important information about the behavior of the software and hardware which may result in some cost savings as well.

Negative impact on life cycle costs:

Additional investment in hardware since logic analyzers aren't free. Use of a processor for which a personality module is available off the shelf can reduce some of the potential costs.

Training may be required but there should be overall savings.

Additional media for storage of data collected from logic analyzer until it can be analyzed.

Respondent 2

Nothing to contribute at this point.

Respondent 3

The Programmable Signal Data Processor (PSDP) contains two separate processors that communicate through shared RAM. There are test jacks for each processor giving the [RTNI monitoring instrument] the ability to "emulate" the UVPROM and to have control of the processor (i.e. start, halt, view/set registers, view/set instruction counter). The software can be loaded and run from the RAM allowing easy access to all variable storage and program lines.

1) Design/Debug time is reduced. Using the [RTNI monitoring instrument], the debugging process for this can be done in the target LRU instead of using a simulator.

This allows for early detection of timing problems between the two processors and between the processors and the I/O.

2) Testing time is reduced. Tests can be automated since the entire LRUs memory is "visible". There will be additional cost on the front end for the development of the initial set of automated test; however, for all follow-on changes, only those portions of the test will get modified. The automated testing, if carefully thought-out at the beginning, will allow the maintainers to only test those sections of the OFP that have changed. Unaffected sections need not be tested.

3) Reliability is increased due to the visibility that the developer has into the code.

4) The Maintenance cost attributable to rework can be reduced.

Respondent 4

First off, I don't know if I agree with your terminology of "the use of real-time non-intrusive monitoring." If I am not mistaken, perhaps you meant to say "the use of non-real-time monitoring device on real-time system." I don't mean to be picky, but without having the clear understanding of the subject definition, information can be distorted.

In my opinion, the real-time non-intrusive monitoring device is most needed during Embedded Computer System(ECS) development/testing/integration with other ECS. While the ECS is running in real-time, the non-real time monitoring device that is connected to the either back-plane or at the edge of the CPUs, monitors the status of the CPU functions in a non-intrusive way.

There could be many other aspects to the use of this during all phases of ECS development cycle including even after the development phase, maintenance of ECS software cycle.

A few sub-categories in ECS phases where the device can be used very effectively would be:

1. CPU through-put timing check,
2. CPU watch-dog monitor(WDM) monitoring during boot-up/testing of any modules,
3. as an aiding device during software patching,
4. checking of synchronization of certain functions,
5. can be a great asset during analysis of ECS Built-in-test software development/testing.

Respondent 5

Question: Identify where RTNI has a positive or negative effect on life cycle cost. Please take into account the following areas: reliability, training, software size, software complexity.

Answer: Executive Summary - The primary benefit to using RTNI tools like the [RTNI monitoring instrument] product is to be able to look at the system, without effecting it's timing performance, and be able to look at the software execution in a system manner. By system manner I am referring to looking at the execution of the software from a top-level view of the data and the actions taken open the data to generate the desired results.

In a very complex system like the [major weapon system] which runs complex fixed point algorithms for navigation, ballistic etc... It is very hard to comprehend the execution of these algorithms at a low level. Even the software engineers need tools to help them see through the complexity of the system to the core of the algorithm that may be executing on the system. For example: we used the [RTNI monitoring instrument] to find numerous problems like a problem we had with the sidewinder algorithm. In this case it has a simple scaling problem in a fixed point calculation. But mixed into all the details of what is executing it would normally take ~1 month to find this problem. The [RTNI monitoring instrument] graphically showed us the sidewinder algorithm executing and we were able to narrow in on the problem in a few hours.

Reliability: All complex embedded systems have bugs in them, they just do not cause a great deal of damage. If they do people see them and try to find them. The [RTNI monitoring instrument] allows you the ability to truly validate the algorithms. Not the hunt and pick method that most testers take to the problem.

When major problems occur, you have to be able to debug the system without impacting the timing. It is hard enough to recreate a problem let allow recreate a problem in a different timing environment. Trying to create complex problems can easily consume a month of your time. Most of the fighter aircraft programs Validation and Verification activities are almost brought to a stand still when a major safety related problem occurs. These programs use on ($100000 - $200000 a day) this is a great expenditure when you are hating on the critical problem. Not to mention stressful when 100 - 300 people can do any work on the OFP until you have found the problem.

Software Size: I do not have a comparison to looking at software size. With or without the [RTNI monitoring instrument] allot of code will be written. Having an RTNI system is vital in a large embedded system such as [aircraft] mission computer operational flight programs.

Complexity: As discussed before in a large embedded system the software engineers are unable to fully comprehend the whole system. Thus an RTNI system needs to give them a high more system engineering look into the problem.

With the growing complexity and safety criticality of a Ground Proximity Warning System, the RTNI unit [RTNI monitoring instrument] was the only way to validate the correctness of the GPWS algorithm. We ran various true models flights against the OFP and graphed the results with the [RTNI monitoring instrument] system.

Areas such as sensor fusion require a great deal of algorithmic intensity which is not easily viewable by the test community. These areas are were the [RTNI monitoring instrument] system became invaluable.

Training: With the [RTNI monitoring instrument] system a lesser able software engineer now has a fighting chance to integrate debug and analyze there design with the system full up and running.

Respondent 6

My interpretation of the term "non-intrusive monitoring" as indicated in your E-mail is the peek/poke capability implemented in our flight software testbed FSDS (SUN based, non-real time) and FSTB (1750 based, real-time). These platforms provide the feature to allow users to retrieve/set memory contents at locations corresponding to any user specified variables. The contents can be sampled at a user specified frequency (max. every RTI - 125msec.) and stored the data onto a disk for analysis. The flight software are executed in parallel while these activities are being processed. The flight software does not have to be instrumented to specify the variables for monitoring. It is performed by a separate task other than the flight task.

Appendix D: Items for Discussion by Participants

Dear Participant:

Thank you for your timely response in the first step of this project. Your input has been extremely valuable.

Based on all of the initial lists of items from all of the participants, I have compiled the lists into various different areas that can be impacted by implementing RTNI testing. In this second stage of the three, we will now try and narrow our focus on existing items and identify additional areas that RTNI can impact. Also, each participant can provide some form of discussion of the items as to whether they agree or disagree with them.

To provide anonymity for all discussion responses, each participant will send their comments and items to me, and I will strip out all identification and in turn, pass it on to all members of the group. The group consists of 6 people, working on projects for the Air Force, the Navy, and NASA.

Additionally, I will provide some questions for discussion on other areas that might need to be considered. You can submit new items to this list as well. The purpose of this second stage is to finalize a list of items that everyone will vote on as to what the biggest impacts on life-cycle costs are. If I have accidentally changed the meaning of your answer while compiling them, please let me know. Also, any discussion you can provide will be valued. Of particular importance, we need to eliminate the duplication of items.

The discussion of items will occur between Friday, July 25 and Thursday, July 31. Thank you all for submitting your initial responses. The third stage of the this process will occur on Friday, August 1. Thank you all for your participation and cooperation in this study.

The items are broken down into the following areas:

Testing Costs

A. Debug time will be reduced because code execution would be done in real time instead of stepping through one instruction at a time.

B. Engineers may be able to execute code on the target system and then examine the results on a separate system at their leisure allowing more engineers to share the target system.

C. Design/Debug time is reduced. Using the [RTNI monitoring instrument], the debugging process for this can be done in the target LRU instead of using a simulator. This allows for early detection of timing problems between the two processors and between the processors and the I/O.

D. Testing time is reduced. Tests can be automated since the entire LRUs memory is "visible". There will be additional cost on the front end for the development of the initial set of automated test; however, for all follow-on changes, only those portions of the test will get modified. The automated testing, if carefully thought-out at the beginning, will

allow the maintainers to only test those sections of the OFP that have changed. Unaffected sections need not be tested.

E. In my opinion, the real-time non-intrusive monitoring device is most needed during Embedded Computer System(ECS) development/testing/integration with other ECS. While the ECS is running in real-time, the non-real time monitoring device that is connected to the either back-plane or at the edge of the CPUs, monitors the status of the CPU functions in a non-intrusive way. There could be many other aspects to the use of this during all phases of ECS development cycle including even after the development phase, maintenance of ECS software cycle. A few sub-categories in ECS phases where the device can be used very effectively would be:

a. CPU through-put timing check,

b. CPU watch-dog monitor(WDM) monitoring during boot-up/testing of any modules,

c. as an aiding device during software patching,

d. checking of synchronization of certain functions,

e. can be a great asset during analysis of ECS Built-in-test software development/testing.

F. All complex embedded systems have bugs in them, they just do not cause a great deal of damage. If they do people see them and try to find them. The [RTNI monitoring instrument] allows you the ability to truly validate the algorithms. Not the hunt and pick method that most testers take to the problem. When major problems occur, you have to be

able to debug the system without impacting the timing. It is hard enough to recreate a problem let allow recreate a problem in a different timing environment. Trying to create complex problems can easily consume a month of your time. Most of the fighter aircraft programs Validation and Verification activities are almost brought to a stand still when a major safety related problem occurs. These programs use on ($100000 - $200000 a day) this is a great expenditure when you are hating on the critical problem. Not to mention stressful when 100 - 300 people can do any work on the OFP until you have found the problem.

G. Examination of code execution will probably allow engineers to discover important information about the behavior of the software and hardware which may result in some cost savings as well.

Additional Considerations for Testing:

- Has the amount of static (code walk-through, non-execution code analysis) testing changed in respect to the amount of dynamic (execution testing) performed during the same time period since the use of RTNI?
- Has the average defect detection rate increased or decreased since the implementation of RTNI? That is, can you find more defects (bugs) in a time period using RTNI than you could with other methods?
- Regression testing is defined as the amount of testing that needs to be performed to ensure that software modifications do not introduce additional defects into the system.

Has the use of RTNI had any impact on the amount of regression testing that needs to be performed?

Complexity

H. As discussed before in a large embedded system the software engineers are unable to fully comprehend the whole system. Thus an RTNI system needs to give them a high more system engineering look into the problem. With the growing complexity and safety criticality of a Ground Proximity Warning System, the RTNI unit [RTNI monitoring instrument] was the only way to validate the correctness of the GPWS algorithm. We ran various true models flights against the OFP and graphed the results with the [RTNI monitoring instrument] system. Areas such as sensor fusion require a great deal of algorithmic intensity which is not easily viewable by the test community. These areas are where the [RTNI monitoring instrument] system became invaluable.

Software Size

I. I do not have a comparison to looking at software size. With or without the [RTNI monitoring instrument] allot of code will be written. Having an RTNI system is vital in a large embedded system such as [aircraft] mission computer operational flight programs.

Reliability

J. Reliability is increased due to the visibility that the developer has into the code.

Additional Consideration for Reliability

- What is the impact on reliability?

- Have the program's reliability goals changed since the use of RTNI (that is, can you now set a higher reliability goal, say 90% vs. 85%, since RTNI has been used)?

Maintenance Costs

K. The Maintenance cost attributable to rework can be reduced.

Hardware Procurement Costs

L. Additional investment in hardware since logic analyzers aren't free. Use of a processor for which a personality module is available off the shelf can reduce some of the potential costs.

M. Additional media for storage of data collected from logic analyzer until it can be analyzed.

Training Costs

N. Training may be required but there should be overall savings.

O. With the [RTNI monitoring instrument] system a lesser able software engineer now has a fighting chance to integrate debug and analyze there design with the system full up and running.

Additional Considerations for Training Costs

- Are training costs for RTNI a one-time, fixed cost during the life-cycle, or are they continuous throughout the life-cycle?

Appendix E: Discussion Results

Respondent 3:

Testing:

The amount of dynamic testing has increased. Using [RTNI monitoring equipment], it is easier to check/follow code execution.

We can capture the "hits" on any address or range of addresses in the code and determine where it was called from. We can also check jump addresses and call addresses to ensure that they are going to the proper location at the proper time. To do this manually is labor intensive and error prone.

Using RTNI, there is less time required to determine the exact cause of a defect. New tests can be created quickly which will narrow the focus of the investigation. This approach is most useful when dealing with multiple processors that communicate using shared memory instead of interrupts.

Reliability:

Reliability is increased since more code can be tested in less time.

We do not presently set reliability goals for the software. We try to get the software functioning properly in the time allotted for the update. Any major problems are fixed; however, minor problems may not be.

Training Costs:

Training costs should be a one time expenditure for each engineer unless the RTNI goes through a major upgrade/change.

Administrator:

Dear Participant:

I propose that items A and C be combined to eliminate duplicate items (which will be important for when we rank order the five most important items). Also, several participants noted during our phone conversations that RTNI monitoring provides them a capability that cannot be provided through any other

means. Is this a correct assessment?

Thank you for your time and participation. The final voting procedure will begin on this Friday, 1 August.

Respondent 6

As a footnote, let me mention that we have been using both non-realtime and realtime approach to system testing and integration. Non-realtime approach is used throughout for initial functional integration and later functional performance testing, while the realtime testing is used to flush out timing and data integrity (mutual exclusion) kind of problems continually, especially in the mid-phase of the development lifecycle.

The realtime testing is also indispensable when it comes to worst case load testing on the system, when it is very easy to severely skew the timeline and stress the system to expose weak points in the design. The sooner this gets done, the cheaper it is to fix. This may also expose the actual hardware quirks which were unknown at the time of the initial and detailed design phases.

During the realtime testing it is essential to have maximum data visibility - both from engineering data and system (OS etc.) data; for maximum analysis and debug effectiveness.

Respondent 5

Hardware procurement does save 160K versus 250K+ but not big in overall costs.

Appendix F: Final List of Ideas

Dear Participant:

The following list contains the areas in which the study group has identified real-time non-intrusive (RTNI) monitoring as having an effect on life-cycle costs. I have taken the liberty of eliminating duplicates by combining the similar ideas, and I have changed the wording of some of the ideas to provide more clarity.

In this final stage, you are to choose the five most important ideas to your program. At this point, simply choose the five ideas that you think had the biggest impact on your program's life-cycle costs.

Next, you need to rank order the five ideas according to their importance to your program. An "outside-in" technique should be used. You should rank the most important item of the five with a "5", and then the least important item with a "1". Then, you rank the second most important with a "4", the second least important with a "2", and the remaining item with a "3". All participants will then electronically mail their responses to the administrator (myself).

The responses need only consist of the five letters that correspond to the ideas in their ranked order from 5 to 1, and any additional comments you may have. All responses provided during this phase will be seen only by myself, and will not be sent to the entire group.

It is important that I have this information as soon as possible. Thus, please send me your choices as soon as possible, but no later than 4 August. Thank you for your time and cooperation during this study.

The ideas are as follows:

A. Debug time is reduced – testing performed in real-time instead of step-by-step analysis; instead of using simulator, timing problems can be found earlier, which means that other engineers do not have to wait as long for the results (wait at a stand still until the problem is found)

B. Sharing of testing time among engineers – engineers can execute code on the target system and then view the results on a separate system, freeing the target system for additional testing

C. Reduced regression testing time – allows for automated tests since memory is "visible"; automated tests allow for testing only those parts of the code that have changed

D. Additional testing data not available before – the examination of the code execution allows engineers to discover important information about the behavior of the software and hardware, which could lead to developments that save costs

E. Increased amount of dynamic testing – with RTNI, it is easier to check and follow code execution, leading to more dynamic testing than before; by capturing the "hits" on

any address, engineers can ensure that call addresses and jump addresses are going to there proper place; before, this required static analysis with mixed results

F. Test case development time reduced – using RTNI, less time is required to determine a cause of a defect; new tests can be created quickly to narrow the focus of the investigation

G. Testing reliability is increased – RTNI allows an engineer to truly validate an algorithm rather than using "hunt and pick" methods; RTNI allows for the recreation of the problem in the real-time environment, rather than in a simulated environment with different timing

H. Time saved in integration – software engineers can provide a higher systems engineering approach to a problem; RTNI is valuable because the various models of complex systems can be viewed at various points within the system

I. Training costs – training will be required, but in the long term, there should be savings; this will be a one-time expenditure unless there are additional upgrades to the RTNI testing tools

J. Overall system reliability – overall system reliability will be increased, since more code can be tested in less time

K. Hardware procurement costs – there are costs incurred in the procurement of RTNI monitoring equipment; also, there will be costs incurred in procuring additional media storage for the data collected during testing

L. Maintenance costs – the maintenance cost attributable to rework can be reduced; RTNI can aid during software patching and during the checking of synchronization of certain functions

Appendix G. Final Ranking by the Participants

Respondent 1

5 Sharing of testing time among engineers
4 Debug time is reduced
3 Time saved in integration
2 Test case development time reduced
1 Reduced maintenance costs

Respondent 2

5 Debug time is reduced
4 Additional testing data not available before
3 Increased amount of dynamic testing
2 Testing reliability is increased
1 Increased training costs

Respondent 3

5 Test case development time reduced
4 Debug time is reduced
3 Reduced maintenance costs
2 Increased overall system reliability
1 Increased amount of dynamic testing

Respondent 4

5 Debug time is reduced
4 Increased amount of dynamic testing
3 Sharing of testing time among engineers
2 Test case development time reduced
1 Testing reliability is increased

Respondent 5

5 Debug time is reduced
4 Increased overall system reliability
3 Increased amount of dynamic testing
2 Testing reliability is increased
1 Reduced maintenance costs

Respondent 6

5 Debug time is reduced
4 Testing reliability is increased
3 Increased overall system reliability
2 Sharing of testing time among engineers
1 Hardware procurement costs increased

Bibliography

Beizer, Boris. Software Systems Testing and Quality Assurance. New York: Van Nostrand Reinhold, 1984.

Boehm, Barry. Software Engineering Economics. New York: Prentice-Hall, Inc., 1981.

Brooks, Frederick P., Jr. The Mythical Man-Month. Reading MA: Addison-Wesley, 1995.

Cooper, Donald R. and C. William Emory. Business Research Methods. Chicago: Irwin, 1995.

Daich, Gregory T., Gordon Price, Bryce Ragland, and Mark Dawood. Software Test Technologies Report. Software Technology Support Center, Ogden Air Logistics Center (AFMC), Hill AFB UT, August 1994.

Debold, Bob. "Nominal Group Technique," WWWeb: (http://www.radix.net/~ash2jam/TQM/nominal.htm), June 28, 1997.

Defense Systems Management College. Mission Critical Computer Resources Management Guide. Washington DC, 1990.

DeMillo, Richard A., et al. Software Testing and Evaluation. Menlo Park CA: Benjamin/Cummings Publishing Co., 1987.

Department of Defense Inspector General. Management of the Software Technology for Adaptable, Reliable Systems Program. DOD IG Audit 91-059. Washington: Department of Defense, February 1991.

Frame, L.H., for the Director, Office of Secretary of Defense. Memorandum: Software Maturity Criteria for Dedicated Operational Test and Evaluation of Software-Intensive Systems. Washington DC. 31 May 1994

Fryer, R. E., G. Vansteenkiste, and J. Van Campenhout. "Real Time Non-Intrusive Monitoring and Embedded System Simulation," Special Proceedings on Embedded Systems of the Society for Computer Simulation Summer Simulation Conference (July 1994).

----. "The 'Instrumentation Memory': A Measuring Feature for Simulation of Real Time Embedded Software," Proceedings of the 1995 Society for Computer Simulation European Multiconference (Prague, Czech Republic) 1995.

Fryer, Richard. "Why Instrument Software in Embedded Systems?" EE Virtual Conference (On-Line) for Embedded Systems (www.eg3.com, March 1997).

Glass, Robert L. "Real-Time: The 'Lost World' of Software Debugging and Testing," Communications of the ACM, Volume 23 Number 5: 264-271 (May 1980).

Government Accounting Office. Test and Evaluation: DOD Has Been Slow in Improving Testing of Software-Intensive Systems. GAO-NSIAD-93-198. Washington: Government Accounting Office, 1993.

Hetzel, Bill. The Complete Guide to Software Testing: Second Edition. New York: John Wiley and Sons, 1988.

Humphrey, Watts. A Discipline for Software Engineering. Reading MA: Addison-Wesley, 1995.

ITCN. C-TAC RTNI Case Study #1. Centerville OH: ITCN Incorporated (1990).

----. C-TAC RTNI Case Study #2. Centerville OH: ITCN Incorporated (1990).

Kishon, A., P. Hudak, and C. Consel. "Monitoring Semantics: A Formal Framework for Specifying, Implementing, and Reasoning about Execution Monitors," Proceedings of the ACM Sigplan '91 Conference on Programming Language Design and Implementation. ACM, New York NY, 1991.

Musa, John D., Anthony Iannino, and Kazuhira Okumoto. Software Reliability: Measurement, Prediction, Application. New York: McGraw-Hill, 1987.

Myers, Glenford J. The Art of Software Testing. New York: John Wiley and Sons, 1979.

Ould, Martyn A. and Charles Unwin. Testing in Software Development. Cambridge, U.K.: University Press, 1986.

Plattner, Bernhard. "Real-Time Execution Monitoring," IEEE Transactions on Software Engineering, Volume SE-10 Number 6: 756-764 (November 1984).

Royer, Thomas C. Software Testing Management: Life on the Critical Path. New Jersey: Prentice-Hall, Inc., 1993.

Software Technology Support Center. Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems (Version 2.0): Volumes 1 and 2. Software Technology Support Center, Ogden Air Logistics Center (AFMC), Hill AFB UT, June 1996.

Wellman, Frank. Software Costing: An Objective Approach to Estimating and Controlling the Cost of Computer Software. New York: Prentice Hall, 1992.

Yourdon, Edward. Managing the System Life Cycle: A Software Development Methodology Overview. New York: Yourdon Press, 1982.

----. Rise and Resurrection of the American Programmer. Upper Saddle River NJ: Prentice-Hall, Inc., 1996.

Zimmerman, Donald E. and Michel Lynn Muraski. The Elements of Information Gathering: A Guide for Technical Communicators, Scientists, and Engineers. Phoenix: Oryx Press, 1995.

Vita

Captain Michael D. Lewis [redacted] graduated from Papillion-LaVista High School in Papillion, Nebraska, in 1988. He received an Air Force ROTC scholarship and attended Embry-Riddle Aeronautical University in Prescott, Arizona, graduating with a Bachelor of Science in Electrical Engineering in December 1992. Upon graduation, he was commissioned as a reserve officer in the U.S. Air Force. He began active duty service at Rome Laboratory, Griffiss AFB, New York. He was lead engineer for the Advanced Planning System (APS) program, and later, lead engineer for the Force Level Execution (FLEX) program. He was transferred to Wright-Patterson AFB in May 1996 to attend the Air Force Institute of Technology in the Software Systems Management Program.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	~~December~~ 1997	Master's Thesis

4. TITLE AND SUBTITLE
A COMPARATIVE STUDY AND ESTIMATION OF THE LIFE-CYCLE COST IMPACT OF APPLICATION OF REAL-TIME NON-INTRUSIVE (RTNI) MONITORING TECHNOLOGY TO REAL-TIME EMBEDDED SYSTEMS

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael D. Lewis, Captain, USAF

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
2750 P Street
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSS/LAS/97D-2

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ASC/ENAS
BLDG. 560 AREA B
WPAFB, OH 45433

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 Words*)
The use of real-time non-intrusive (RTNI) monitoring has had an impact on life-cycle costs of existing programs through a reduction in debug time. Other areas in which RTNI monitoring can provide potential benefits to future programs are through the use of increased dynamic testing and the sharing of testing time among more engineers. There are a number of areas in which software life-cycle costs are impacted by various cost drivers. To determine which areas were affected by the use of RTNI monitoring, a panel of expert users of RTNI monitoring was created using a form of the Nominal Group Technique (NGT) methodology to achieve a consensus from a group of experts. The group concluded that the above areas were most important across their programs. However, simply using RTNI monitoring may not have a major impact on future programs, unless it is accompanied by the commitment from management to successfully integrate it into the test programs of those future programs.

14. Subject Terms	15. NUMBER OF PAGES
Real Time, Non Intrusive, Software Engineering, Test Management, Test Equipment	82
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

AFIT Control Number AFIT/GSS/LAS/97D-2

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. **Please return completed questionnaire** to: AIR FORCE INSTITUTE OF TECHNOLOGY/LAC, 2950 P STREET, WRIGHT-PATTERSON AFB OH 45433-7765. Your response is **important.** Thank you.

1. Did this research contribute to a current research project? a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?
a. Yes b. No

3. **Please estimate** what this research would have cost in terms of manpower and dollars if it had been accomplished under contract or if it had been done in-house.

Man Years_____ $_____

4. Whether or not you were able to establish an equivalent value for this research (in Question 3), what is your estimate of its significance?

a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments (Please feel free to use a separate sheet for more detailed answers and include it with this form):

_____ _____
Name and Grade Organization

_____ _____
Position or Title Address

FOLD AT THE DASHES SHOWN ABOVE AND SEAL WITH TAPE OR WAFER SEALS. DO NOT STAPLE.