Technical Report
CMU/SEI-87-TR-036
ESD-TR-87-199

Prototype Real-Time Monitor: Requirements

Richard D’Ippolito
Kenneth Lee
Charles Plinta
Michael Rissman
Roger Van Scoy

November 1987

Technical Report

CMU/SEI-87-TR-36

ESD-TR-87-199

November 1987

Prototype Real-Time Monitor: Requirements

Richard D'Ippolito

Kenneth Lee

Charles Plinta

Michael Rissman

Roger Van Scoy

Dissemination of Ada Software Engineering

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office
ESD/XRS
Hanscom AFB, MA 01731

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Karl H. Shingler SIGNATURE ON FILE
SEI Joint Program Office

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

Prototype Real-Time Monitor: Requirements

Abstract. The requirements imposed by flight simulators and good software engineering practice on Ada®[1] systems force software engineers to seek new solutions to the problem of monitoring executing software. This report examines some of these requirements and, based on these requirements, defines a subset for implementation as a prototype real-time monitor (RTM).

Preface

Intended Audience

This document is intended to be read and used by systems engineering professionals to design and produce a real-time monitor package that meets all of the requirements stated in this report.

Associated Documents

- *Prototype Real-Time Monitor: Executive Summary* [Van Scoy 87a]
- *Prototype Real-Time Monitor: Design* [Van Scoy 87b]
- *Prototype Real-Time Monitor: User's Manual* [Van Scoy 87c]
- *Prototype Real-Time Monitor: Ada Code* [Van Scoy 87d]

Context of Report

The prototype RTM in this report was built to address two specific technical questions raised by the Ada Simulator Validation Program (ASVP) contractors:

1. How can user tools find, access, and display data hidden in the bodies of Ada applications?
2. How can user tools be layered on top of Ada applications?

The prototype is documented by this report because the ASVP contractors needed a monitor tool, but did not have the contract resources to develop one. The prototype RTM is intended to be a simple tool that is easily rehosted and extended. It is not intended to be an example of what a well-documented system should include. Since it was a prototyping effort, no standard documentation or development methods were applied. Also, we did not attempt to solve all the traditional "monitor" problems.

[1]Ada is a registered trademark of the U.S. Government (Ada Joint Program Office).

1. User's Problem Statement

There is a large class of computer programs that run in real time and are used to control mechanical apparatus. These programs are generally characterized by some sort of high-speed signal transfer interface to transmit and obtain signals from the real world apparatus that is being controlled. It is the nature of these programs that they must be capable of reacting to and providing the real world stimuli without interruption or pause, or data may be lost. This could result in errors in the control of, or the possible destruction of, the real world apparatus.

One of the problems associated with this specific class of computer programs is obtaining accurate information about the internal states of the controlling algorithms. There are two reasons to require this information: The first is to obtain enough information to be able to evaluate the performance of the process controller. Sometimes this investigative process is more effectively carried out in real time while the system is active. Other times, a stored record of pertinent internal state data is required. The second reason is to be able to do dynamic adjustment of the process-control parameters. This is difficult because any attempt to add the extra workload of having the program report status to an external observer may introduce delays in the control algorithms, which may have the undesired effect of changing the state during the measurement.

Ada programs used for real-time simulation are a particular class of computer programs of interest. Monitoring these programs is difficult because well-engineered Ada programs enforce the general principle of information hiding: that is, modules within the program generally have access to state information about other modules only on a need-to-know basis. This makes it very difficult to introduce into the system a monitoring device that is capable of making visible (to an external observer) state information that is well hidden due to the lack of a universal repository of such information. To further compound the problem, the real-time nature of the program virtually assures that the internal states are constantly changing except for a few systems whose real-world response is extremely slow relative to the computer cycle time.

What is needed, then, is a software-state monitoring method that takes the dynamic nature of the system into account while producing state information for external display. The monitor, while extracting the information, must not introduce undue changes in the nature of the system being monitored, and it must produce the information in a timely enough fashion to be still meaningful and useful to the observer. It is important for the observer to be able to select the categories and particulars of the state information and process-controlling parameters to be monitored and for the observer to be able to maintain running histories of the information for further off-line study of the system. It is also important to be able to make timely changes in the process-controlling parameters to adjust the processes effectively.

2. Requirements Analysis

To obtain a clearer understanding of the needs and wants of the user, it is necessary to elaborate on the problem definition provided by the user. This is an appropriate step in that it allows the designer to clarify key points in the user's problem statement. Once the designer has presented an interpretation of the problem, a dialogue with the user can begin, and points of contention and misunderstanding can be resolved in an interactive process. To achieve these ends, the following paragraphs extract important phrases from the user's problem statement starting on page 3, denote them with bold and italicized text, and add additional detail based on the experience and understanding the designer brings to the problem.

To begin with, the user has stated that a ***software-state monitoring method*** is needed in this problem domain. A requirement of such a monitor is that it extract state information and update process-control parameters at a periodic rate. This periodic rate will be under user control so that the monitor can fit into any desired timing pattern.

Since both ***state information*** and ***process-control parameters*** are embodied by Ada variables, all Ada data types except the following shall be accessible via the monitor:

- **access** type variables[2]
- **task** type variables

Finally, given the critical nature of some ***state information*** and ***process-control parameters***, the user will have the ability to protect information (from modification via the monitor) by designating it as read-only or unprotect it by designating it as read/write (which is the default status of all information).

The ***external display*** will be a video display terminal (VDT) dedicated to monitoring use during the active participation of the monitor.

When the user requests that the monitor ***must not introduce undue changes in the nature of the system being monitored***, this dictates that the active participation of the monitor must not adversely affect the timing of the monitored system.

To be able to ***produce the information in a timely enough fashion for it to be still meaningful and useful*** is critical to the usability of the monitor. Failure to achieve this is a failure of the monitor as a whole. To achieve this, the monitor is obliged to extract as much information as possible from the monitored program during its period of active execution. If the monitor cannot process all the user's commands in a given time period, it must identify to the user those actions that were not processed.

A key function from the user standpoint is the ability to ***select the categories and particulars of the state information and the process-control parameters***. Under the heading of ***categories***, the user will be able to:

[2]The objects accessed are monitored; access values (i.e., addresses) are not.

- Select the state information for display.
- Select the process-control parameters for display.
- Select the state information for history recording.
- Select the process-control parameters for history recording.

Under the heading of ***particulars***, the user will be able to:

- Format the display.
- Format the state information and process-control parameters. Specifically, the user will be able to define (for each monitored item):
    - units (such as meters, feet, etc.)
    - numeric representation (binary, octal, decimal, hexadecimal, scientific notation, engineering notation, floating point, fixed point)
    - rate at which the display is updated
    - rate at which the history is updated

The ability to ***maintain running histories*** requires that the monitor be able to save ***state information*** and ***process-control parameters*** to a permanent storage device and to provide a mechanism for retrieving that history at a later time.

Finally, the ability to ***make timely changes in the process-controlling parameters*** requires that the user be able to select which items to adjust and that the application of any given set of adjustments be under the user’s control and occur at a known, predictable time and in a deterministic manner.

3. Designer's Problem Statement

The problem statement presented here contains four sections:

1. **Definitions of Terms**: terms whose meanings are specific to the context of this report.
2. **User's Needs**: essentially a recasting of the user's problem statement in technical terms, with no added content.
3. **Designer's Technical Program-Related Additions**: statements that come from the designer's body of expert knowledge and consist of additions of technical content related to the problem.
4. **External Considerations**: considerations either of a technical or non-technical, but not problem-related, nature.

The statements in this chapter represent the top-level framework for the solution and are approved by the user.

3.1. Definitions of Terms

The following terms have special meaning within the context of this document:

Flight simulator: The hardware and software packages exclusive of the *monitor*, used to implement and control the simulated aircraft.

Monitor: The software package (also referred to as the RTM) to be used on the flight simulator computer to obtain, store, and display system-state information to the user.

Process-controlling parameters: Internally stored coefficients that are used in the control loop equations to determine the feedback amounts and system outputs of the flight simulator.

State information: The values of internally generated and stored numbers that represent the flight simulator inputs, outputs, or intermediate results of calculations used to determine them.

User: The person operating and directing the monitor to obtain, present, and modify the flight simulator state information and process-controlling parameters. The user is not the instructor or trainee who use the flight simulator software.

3.2. User's Needs

1. The monitor will be capable of extracting all flight simulator information that is internally represented as Ada variables and provide within the monitor a flexible means to display that information on a video display terminal.
2. The monitor will be integrated into the flight simulator software so that its use will not degrade the normal operations of the flight simulator.
3. The monitor will have user-selectable flight simulator information extraction and display rates.
4. The monitor will have user-selectable display formats for the flight simulator information, with display storage and recall.

5. The monitor will have user-modifiable values for the process-controlling parameters. These modifications will become effective only in a predictable and deterministic manner.

3.3. Designer’s Technical Problem-Related Additions

6. The monitor will be written in the Ada programming language.
7. The monitor will be compatible with and function with the flight simulators of the two ASVP contractors (i.e., Boeing and Burtek). This means that the monitor must be capable of being compiled, loaded, and executed with both systems.
8. The monitor will prohibit the Ada ***access***[3] and ***task*** type variables from being accessible to the monitor. We will also prohibit the Ada object ***constant*** from modification, as it cannot be changed after initialization.

3.4. External Considerations

9. The monitor will use good software engineering principles and produce a well-documented, extensible, modular, and maintainable system.[4]

[3]The objects accessed are monitored; access values (i.e., addresses) are not.

[4]That is, as long as these requirements do not conflict with the prototype nature of the development.

4. Real-Time Monitor Requirements

This chapter sets forth the requirements for a real-time monitor package (the monitor) that will be integrated into a hardware/software system used as an aircraft trainer/flight simulator. This package will be used to provide system state information to the user for the purposes of monitoring, recording, and adjusting the performance of the flight simulator software. While the contents of this document are believed to be logically consistent and sufficient to describe the required operation of the monitor, no guarantee of completeness is made.

The top-level requirements for the system are derived in Chapter 3, Designer's Problem Statement. The system is composed of the three major system objects: the ***user***, the ***monitor***, and the ***flight simulator***. There are three possible interfaces for information exchange among the three objects. The interface between the user and the monitor is characterized by the *User's Manual* [Van Scoy 87c]. The interface between the monitor and the flight simulator is characterized by the *System Interface Manual*.[5] The third interface is between the user and the flight simulator and is not a part of this problem.

4.1. User Interface Requirements

The requirements for the monitor were extracted during the composition of the *User's Manual* and the *System Interface Manual*. A reading of the user's needs (Section 3.2, paragraphs 3, 4, and 5) yields the following user-controlled actions:

R1. Activate the monitoring process.

R2. Deactivate the monitoring process.

R3. Select the information to be extracted for display.

R4. Select the rate that information is extracted.

R5. Select the format of the displayed information in advance.

R6. Select the format of the displayed information on demand.

R7. Initiate the display of extracted information.

R8. Terminate the display of extracted information.

R9. Cause a display to be permanently stored.

R10. Cause a stored display to be recalled for viewing.

R11. Cause new values to be entered for process-controlling parameters.

A reading of the designer-imposed additions in Section 3.3, paragraph 7, gives the following additional user-controlled actions:

R12. Transfer the monitor software package, including libraries, to the target hardware and compile it there.

[5]The *System Interface Manual* is the Ada package specification of the *Rtm_core*.

R13. Provide and connect any special monitor hardware to the target hardware.

R14. Load the monitor package together with the flight simulator package on the target hardware.

R15. Initiate the execution of the combined system.

4.2. System Interface Requirements

A reading of the user's needs in Section 3.2, paragraphs 1 and 2, yields the following system-controlled actions:

R16. Extract the value of flight simulator Ada variables at the monitor's request.

R17. Service the monitor's requests to translate information for the monitor's VDT.

R18. Provide the necessary computing and memory resources to the monitor from spare capacity.

R19. Do not degrade the normal operational performance of the flight simulator.

A reading of the designer-imposed additions in Section 3.3, paragraphs 6 and 8, give the following additional monitor-controlled actions:

R20. Refuse the request to extract flight simulator Ada variables if they are of **task** type.

R21. Refuse the request to modify flight simulator Ada objects if they are of **constant** type.

R22. Extract objects referenced when requested to extract Ada **access** type objects.

4.3. General Requirements

A reading of the external considerations in Section 3.4, paragraph 9, and the designer imposed additions in Section 3.3, paragraph 6, gives an extensive set of general requirements. Some of these apply to the production of the code and documents; others apply to the design and implementation of the system.

R23. Handle imperfect inputs in a reasonable, graceful manner.

R24. Implement the monitor in Ada.

R25. Develop a well-engineered monitor.

R26. Develop an extensible monitor.

R27. Develop a modular monitor.

R28. Develop a maintainable monitor.

R29. Develop a well-documented monitor.

5. Prototype Requirements

Given the context of the real-time monitor described in the the *Real-Time Monitor: Executive Summary* [Van Scoy 87a], certain restrictions were imposed on the requirements presented in Chapter 4. These restrictions were imposed to limit the size of the effort (because of time and staffing constraints) and to restrict the scope of the task to the technically challenging areas. While these restrictions limit the functionality of the prototype developed, the requirements implemented still represent a meaningful and useful subset.

To develop a useful prototype, the requirements were divided into four classes:

- fully implemented by the prototype
- partially implemented by the prototype
- fully implemented by the user of the prototype
- not implemented

Each requirement class is delineated below.

5.1. Fully Implemented Requirements

These requirements are fully implemented by the prototype RTM. They were chosen because they represent a useful subset of the full system functionality and form the basis upon which extensions can be built.

R1. Activate the monitoring process.

R2. Deactivate the monitoring process.

R4. Select the rate at which information is extracted.

R7. Initiate the display of extracted information.

R8. Terminate the display of extracted information.

R16. Extract the value of flight simulator Ada variables at the monitor’s request.

R17. Service the monitor’s requests to translate information for the monitor’s VDT.

R20. Refuse the request to monitor flight simulator Ada variables if they are of **task** type.

R21. Refuse the request to modify flight simulator Ada objects if they are of **constant** type.

R22. Extract objects referenced when requested to extract Ada **access** type objects.

R23. Handle imperfect inputs in a reasonable, graceful manner.

R24. Implement the monitor in Ada.

R25. Develop a well-engineered monitor.

R26. Develop an extensible monitor.

R27. Develop a modular monitor.

R28. Develop a maintainable monitor.

R29. Develop a well documented monitor.

5.2. Partially Implemented Requirements

These requirements contribute to the baseline functionality of the monitor, but are restricted to limit the scope of the implementation (i.e., items of the "bell and whistle" variety were deliberately excised from the requirements). The requirements are listed, with the restrictions shown in *italics*.

R3. Select the information to be extracted for display: *not all the variables in the flight simulator are available, as noted elsewhere, but additional restrictions depend on the sophistication of the address generation scheme used by the implementation (see the Prototype Real-Time Monitor: Design [Van Scoy 87b] for additional information)*.

R5. Select the format of the displayed information in advance: *decimal output for integers and scientific notation for floats; no other notations (octal, binary, etc.)*.

R6. Select the format of the displayed information on demand: *see restrictions for previous requirement.*

R11. Cause new values to be entered for process-controlling parameters: *only variables which are accessible can be modified, see above restrictions on accessibility, and then only using values that are compatible with the Ada type of the variable*.

R12. Transfer the monitor software package, including libraries, to the target hardware and compile it there: *system dependencies were removed wherever possible and isolated when necessary.*

5.3. User-Implemented Requirements

These requirements are imposed on the user because they represent system level needs about which the monitor has no knowledge:

R12. Transfer the monitor software package, including libraries, to the target hardware and compile it there.

R14. Load the monitor package together with the flight simulator package on the target hardware.

R15. Initiate the execution of the combined system.

R18. Provide the necessary computing and memory resources from spare capacity to the monitor.

R19. Do not degrade the normal operational performance of the flight simulator:

5.4. Unimplemented Requirements

These requirements are not implemented because they are extensions to the baseline requirements established above.

R9. Cause a display to be permanently stored.

R10. Cause a stored display to be recalled for viewing.

R13. Provide and connect any special monitor hardware to the target hardware.

References

[Van Scoy 87a]	Van Scoy, R. *Prototype Real-Time Monitor: Executive Summary*. Technical Report CMU/SEI-87-TR-35, Software Engineering Institute, November, 1987.
[Van Scoy 87b]	Van Scoy, R., C. Plinta, T. Coddington, R. D'Ippolito, K. Lee, and M. Rissman. *Prototype Real-Time Monitor: Design*. Technical Report CMU/SEI-87-TR-38, Software Engineering Institute, November, 1987.
[Van Scoy 87c]	Van Scoy, R., C. Plinta, T. Coddington, R. D'Ippolito, K. Lee, and M. Rissman. *Prototype Real-Time Monitor: User's Manual*. Technical Report SEI-CMU/SEI-87-TR-37, Software Engineering Institute, November, 1987.
[Van Scoy 87d]	Van Scoy, R. *Prototype Real-Time Monitor: Ada Code*. Technical Report CMU/SEI-87-TR-39, Software Engineering Institute, November, 1987.

Table of Contents