Requirements-based Monitors for Real-Time Systems

Dennis K. Peters
Electrical and Computer Engineering
Faculty of Engineering and Applied Science
Memorial University of Newfoundland
dpeters@engr.mun.ca
http://www.engr.mun.ca/~dpeters/

November 15, 2000

Abstract

Before designing safety- or mission-critical real-time systems, a specification of the required behaviour of the system should be produced and reviewed by domain experts. After the system has been implemented, it should be thoroughly tested to ensure that it behaves correctly. This is best done using a monitor, a system that observes the behaviour of a target system and reports if that behaviour is consistent with the requirements. Such a monitor can be used both as an oracle during testing and as a supervisor during operation. Monitors should be based on the documented requirements of the system.

This paper discusses design of monitors for real-time systems, and examines the conditions under which a monitor will produce false reports. We describe the conclusions that can be drawn when using a monitor to observe system behaviour.

1 Introduction

Computer systems are increasingly being used in situations where their correct behaviour is essential for the safety of people, equipment, the environment and businesses. In many cases there are *real-time* requirements on the behaviour of these systems—failure to satisfy timing constraints is as costly as responding incorrectly.

When designing such safety- or mission-critical real-time systems, good engineering practice dictates that a clear, precise and unambiguous specification of the required behaviour of the system be produced and reviewed for correctness by experts in the domain of application of the system.

After the system has been implemented, it should be tested to ensure that its behaviour satisfies the requirements. In safety-critical applications the system should be monitored by an independent safety system to ensure continued correct behaviour. To achieve these goals there must be a means of quickly determining if the observed behaviour is acceptable or not; this can be quite difficult for complex real-time systems. A practical approach to analysing the behaviour of a real-time system is to use a *monitor*: a system that observes and analyses the behaviour of another system (the *target system*). Such a monitor could be used either as an 'oracle'[6] during system testing, or as a 'supervisor'[4] to detect and report system failure during operation.

This paper examines the relationship between the target system and the monitor, in particular with respect to the means by which the monitor observes the system behaviour, and the impact of this on the usefulness of the monitor. It gives some necessary conditions for monitor feasibility.

This work focuses on monitors for computer-based systems that are intended to observe and/or control some quantities external to the computer. Such quantities are often best represented by continuous, rather than discrete, valued functions. In particular, the requirements for any real-time systems will include time, which we model as a continuous variable.

Section 2 briefly presents the "Four Variable Model", which relates system and software requirements in terms of the behaviour of the input and output devices. Section 3 discusses possible monitor configurations. Section 4 discusses the impact of realistic monitor input devices on the conclusions that can be drawn from using the monitor, and gives some necessary conditions that must be satisfied in order for a monitor to be useful. The final section draws some conclusions and suggests future work.

2 The Four Variable Requirements Model

When specifying system and software requirements it is important to distinguish quantities that are in the environment, i.e., external to the system, from those that are internal to the system or artifacts of the description of either the requirements or the design. The "Four Variable Model" [1, 5], defines *environmental quantities* as those that are independent of the chosen solution and are apparent to the "customer"; they are the best quantities to use when describing the requirements for the system. These quantities will include such things as physical properties (e.g., temperature, pressure, location of objects), values or images displayed on output display devices, settings of input switches, dials etc., and states of controlled devices. Such quantities can be modelled by functions of time.

The environmental quantities of interest can be classified into two (not necessarily disjoint) sets: the *controlled* quantities—those that the system may be required to change the value of, and the *monitored* quantities—those that should affect the behaviour of the system. A *monitored state function*, denoted $\underline{m}^t$, is a vector function describing the values of the monitored quantities for some period. Similarly, a *controlled state function*, denoted $\underline{c}^t$, describes the values of the controlled quantities.

The environmental quantities cannot usually be directly observed or manipulated by the system software, but must be measured or controlled by some devices (e.g., sensors, actuators, relays, buttons), which communicate with the software through the computer's input or output registers, represented by program variables. The *input* quantities are those program variables that are available to the software and provide information about the monitored quantities. An *input state function* denoted $\underline{i}^t$ represents the values of the input quantities. Similarly, the *output* quantities are those program variables through which the software can change the value of the controlled quantities. An *output state function*, denoted $\underline{o}^t$, represents the values of the output quantities.

Using these four sets of quantities, the system requirements and design, and software requirements can be expressed in five relations, as follows.

System Requirements, REQ: Characterizes the required behaviours of the system—it contains only those values of $(\underline{m}^t, \underline{c}^t)$ that are acceptable behaviours of the system. Note that, since implementations will invariably introduce some amount of unpredictable delay, or inaccuracy in the measurement, calculation, or output of values, **REQ** will not be functional for real systems, i.e., there will be more than one acceptable $\underline{c}^t$ for a given $\underline{m}^t$.

Environmental Constraints, NAT: Characterizes the possible values of the environmental state function—it contains all values of $(\underline{m}^t, \underline{c}^t)$ that are possible in the environment. This describes constraints imposed by physical laws of nature independent of the system to be built.

Input Relation, IN: Characterizes the behaviour of the input devices—the possible values of $\underline{i}^t$ for any instance of $\underline{m}^t$.

Output Relation, OUT: Characterizes the behaviour of the output devices—the possible values of $\underline{c}^t$ for any instance of $\underline{o}^t$.

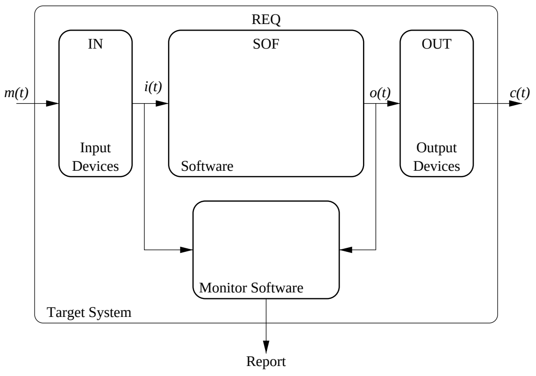

Figure 1: Software Monitor

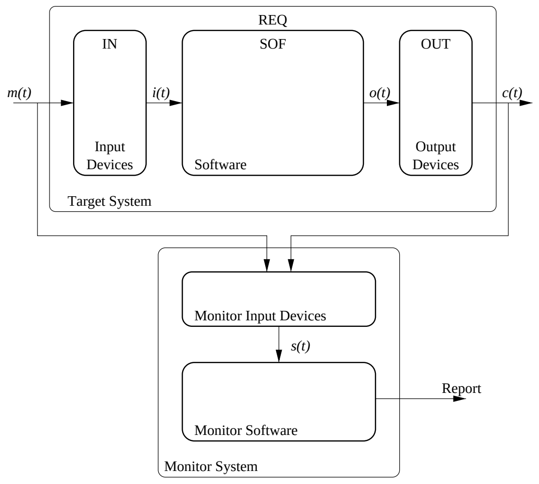

Figure 2: System Monitor

Software Requirements, SOFREQ: Characterizes the set of acceptable behaviours of the software—those pairs, $\left(\underline{i}^t, \underline{o}^t\right)$, such that any possible environmental state function, with respect to the given input and output devices and environmental constraints, are acceptable. This is fully determined by **REQ**, **IN**, **OUT** and **NAT**.

3 Monitors for Real-Time Systems

Testing a real-time system typically involves running the target system in a test environment, observing its behaviour and comparing it to that required by its specification. A *monitor* is a system that observes the behaviour of a system and determines if it is consistent with a given specification. That is, an ideal monitor reports the value of $\mathrm{REQ}\left(\underline{m}^t, \underline{c}^t\right)$.

3.1 Monitor Configuration

In this work, the monitor is assumed to consist of some software running on a computer system. The monitor software cannot, in general, observe the environmental state function, $\left(\underline{m}^t, \underline{c}^t\right)$, directly, but must do so through some input devices that communicate the values of the environmental quantities to input registers known as the *monitor software inputs.* A *monitor input state function* is a function, $\underline{s}^t$, representing the value of the monitor software inputs for the periods of monitor operation. The behaviour of the monitor input devices is characterized by the monitor input relation, $\mathbf{IN_{mon}}$. An environmental state function–input state function pair is in the monitor input relation, $\left(\left(\underline{m}^t, \underline{c}^t\right), \underline{s}^t\right) \in \mathbf{IN_{mon}}$, if and only if $\underline{s}^t$ is a possible monitor input state function for the environmental state function represented by $\left(\underline{m}^t, \underline{c}^t\right)$. Since the monitor must observe all acceptable behaviours, it is required that $domain(\mathbf{IN_{mon}}) \supseteq \mathbf{REQ} \cap \mathbf{NAT}$.

The design of the monitor will determine, for each monitored or controlled quantity, whether it is observed independently of the target system (i.e., using different devices) or observed directly from the target system software. This results in two basic monitor configurations, in addition to the obvious mixtures of these approaches:

Software Monitor A *software monitor* is a monitor that directly observes the target system software input and output variables, i.e., $\underline{s}^t = \left(\underline{i}^t, \underline{o}^t\right)$, as illustrated in Figure 1.

System Monitor A *system monitor* is a monitor that observes $\left(\underline{m}^t, \underline{c}^t\right)$ using its own input devices as illustrated in Figure 2.

4 Practical Monitors

Practical monitors are likely to be implemented using either general- or special-purpose digital computers. This technology implies certain characteristics of the monitor input relation, and monitor behaviour, which influence the conclusions that can be drawn from the monitor output. This section discusses these characteristics, and states some conditions which must hold in order for the monitor to produce meaningful results.

4.1 Observation Errors

The choice of devices and/or software used by the monitor to observe the environmental quantities is a major design decision with respect to the monitor system. Assuming that the monitor is a discrete-time system, there are two basic approaches to observing behaviour:

- Sample (i.e., observe the instantaneous value of) the relevant quantities at intervals.
- Modify the behaviour of the target system, and/or the systems that interact with it, to have them notify the monitor system of the values of relevant quantities ($\underline{m}^t, \underline{c}^t, \underline{i}^t$ or $\underline{o}^t$) as they read or change them. Such notification is assumed to include a *timestamp* indicating the time at which the reported value was observed by the target system.

4.1.1 Discrete Time

Regardless of whether sampling or notification is used, time can only be measured at discrete points: if sampling is used then the sampling period determines the smallest relevant clock increment, whereas if notification is used it is determined by the precision of the notification timestamp. If we assume that using the notification approach the monitor receives notifications for all relevant changes, then this approach is not significantly different from the sampling approach in which the sampling period is the precision of the notification timestamp and uninteresting samples discarded. Thus, the results from sampling theory (e.g., see [3]) can be applied here to show that, for infinite duration signals (behaviours), it is sufficient to sample at twice the maximum frequency of change in the environmental quantities. However, the monitor is typically concerned with what has happened between the most recent two samples, and so the discrete clock will introduce some error in the perceived time of events, which is referred to as the *time error*. For real-time systems, errors in measuring time are particularly important.

Consider the behaviours illustrated in Figure 3, in which the values of m and c represent that a condition of, respectively, a monitored and controlled quantity is either *false* (low) or *true* (high). Similarly, the values of s_m and s_c represent the values as they appear to the monitor software and the shaded regions represent the image of these changes under $\mathbf{IN_{mon}}^{-1}$. Let δ_{mon} represent the monitor sampling interval (i.e., $m_i - m_{i-1}$) and d be the elapsed time between the change in m and c, as illustrated. Assuming that the change in c is a correct target system response to the change in m, consider the two cases illustrated.

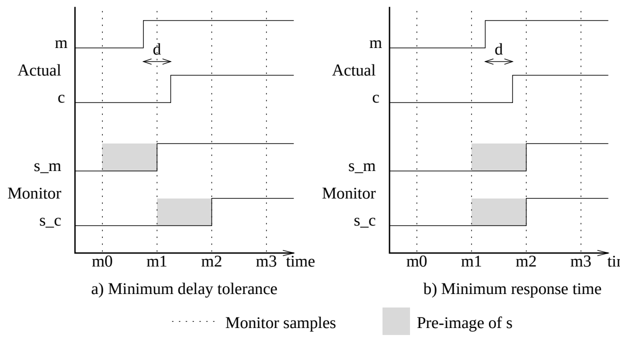

Figure 3: Time Accuracy

a) The monitor sees distinct changes. The monitor can determine only that $0 < d < 2\delta_{mon}$. This behaviour will be rejected (considered unacceptable) if the specified maximum delay for that change is less than $2\delta_{mon}$. This results in Condition 1, below.

Condition 1 *The maximum time error introduced by the monitor input devices must be less than* $\frac{1}{2}min(\mathbf{Delay})$, *where* **Delay** *is the set of maximum delay tolerances for the dependent*[1] *quantities given in the System Requirements Document (SRD).*

b) The monitor sees simultaneous changes. Here the monitor can determine that $-\delta_{mon} < d < \delta_{mon}$ (i.e., c could change before m); hence this behaviour will be rejected if c is only permitted to change following m. The implication is that δ_{mon} must be less than the minimum response time of the target system. This constraint can be weakened, however, by noting that, in order for the target system to have responded to the change in m, it must have observed its value between the changes in m and c, so this case can be avoided by ensuring that the monitor samples in that interval as well. Thus we have Condition 2, below, which can be satisfied by ensuring that sampling by the target and monitor systems is synchronized to within the minimum target system response time. If event notification from the target system is used, the monitor and target systems are assured to be synchronized.

Condition 2 *The maximum difference between the time error in the target system and the time error in the monitor system for the same event must be less than the minimum time in which the target system might respond to that event.*

A monitor system that does not satisfy Condition 1 will be infeasible. A system that does not satisfy Condition 2 may give false negative results for target systems responding too quickly.

4.1.2 Quantization and Measurement Error

As with time, other values observed by the monitor software must be of finite precision, so **Real** valued environmental quantities must be quantized, such that, for example, discrete value v_i represents all continuous values, x, such that $l_i < x \leq h_i$. Whereas time is continuously increasing, so we know something about the error, other quantities do not necessarily have this property. As illustrated in Figure 4, if the quantization is perfect, i.e., $h_i = l_{i+1}$, the worst case error is half the quantization step size, $h_i - l_i$, and no non-determinism is introduced. Practical devices will exhibit some measurement error in addition to quantization, so the actual error will be larger, and $\mathbf{IN_{mon}}$ will be non-functional.

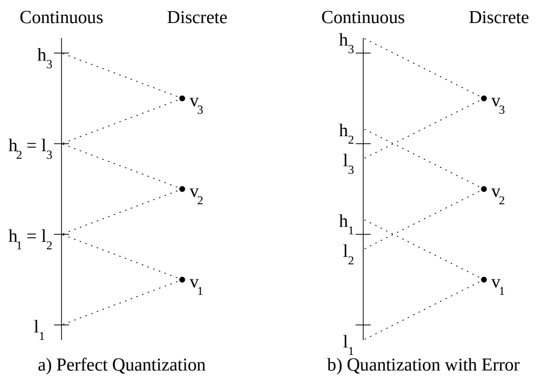

Figure 4: Quantization and Error

For a monitor to be feasible, there must be some monitor input state functions, $\underline{s}^t$, for which all images under $\mathbf{IN_{mon}}^{-1}$ are acceptable. Because of the variety of ways that quantities may be used in the SRD, we cannot state generally applicable conditions on $\mathbf{IN_{mon}}$ that will ensure that a monitor is feasible. Condition 3 is a necessary, but not sufficient condition for feasibility.

Condition 3 *The maximum error in observing a particular controlled quantity must be less than the difference between the maximum and minimum values of that quantity permitted by* **REQ**.

4.2 Non-determinism

As mentioned in Section 2, practical requirements documents will be non-functional to allow for unpredictable delays or errors in calculation or measurement. In particular, if the target

[1]A quantity c is dependent on m if the value of c may be required to change as a result of a change in the value of m.

system is to be implemented using a discrete-time system, then, for some small time, r, **REQ** must allow events that occur within r of each other to be treated as either a single event (i.e., simultaneous) or distinct events (i.e., non-simultaneous). The time r is known as the *time resolution* for the target system. The monitor system must take this non-determinism into account when evaluating behaviour.

Consider the behaviour illustrated in Figure 5, and the target time resolution as indicated. The requirements must allow the changes in C1 and C2 to be treated as either simultaneous or not in both cases illustrated. Assuming that the monitor system samples at the indicated times, it will observe the changes either simultaneously or not, but can certainly tell that they occurred within $2\delta_{mon}$ of each other. If δ_{mon} is less than half the time resolution required for the target, which is required to satisfy Condition 1, then in both cases all images of $\underline{s}^t$ under $\mathbf{IN_{mon}}^{-1}$ allow the changes to be interpreted as happening in either order or simultaneously, so the monitor accepts a behaviour in which the target system interprets them in either way. The monitor software must take this non-determinism into account.

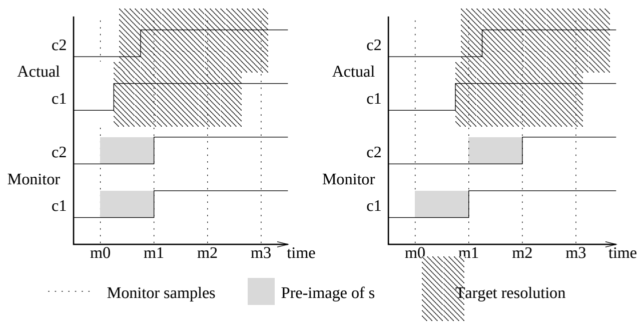

Figure 5: Event Resolution

In the case of the software monitor configuration, as illustrated in Figure 1, the monitor software and the target system software are assured to see the same values (i.e., $\underline{s}^t = (\underline{i}^t, \underline{o}^t)$), so the monitor implementation can require deterministic behaviour.

4.3 Response Time

Clearly the delay introduced by the monitor input devices will impose a lower limit on the monitor response time—the maximum time between a failure occurring and the monitor reporting it—since a monitor cannot report a failure before it is evident in $\underline{s}^t$. The choice of input devices can also affect the amount of processing required by the monitor software, which will also affect response time, although less predictably so. For example, input devices may be available that can directly report the value of relevant conditions (e.g., sensors to detect if a robot has touched a wall) whereas a different choice of input devices would require that the monitor software perform some, possibly expensive, calculations (e.g., search a list of wall locations to determine if the robot is touching any).

5 Conclusions

This paper presents some necessary conditions for a monitor for a real-time system to be feasible and useful. These conditions can be used to help determine if a particular monitor design is sufficient for the target system.

Monitors, such as described in this work, are well suited to automated testing of systems, where they function as an oracle, reporting if the behaviour is acceptable or not. This application offers significant improvement over non-automated testing since test cases can be evaluated quickly and errors in behaviour are quickly and reliably detected.

In a similar way, monitors can be used as supervisors to observe the behaviour of the target system in operation and report failures as they occur. Such a supervisor could be used as a redundant safety system to initiate corrective or preventative action when a failure is detected.

5.1 Future Work

We have validated this work using a few software monitors that were automatically generated from system requirements documentation.[2] Further study, using different target systems and using the system monitor configuration discussed in Section 3.1 would undoubtedly lead to new insight.

Further work is also needed to enhance techniques for specifying the behaviour of input and output devices, and to develop analysis techniques that will permit designers to easily determine if a particular set of monitor input devices is sufficient for the monitoring task at hand.

References

[1] D. L. Parnas and J. Madey. Functional documentation for computer systems. *Science of Computer Programming*, 25(1):41–61, Oct. 1995.

[2] D. K. Peters. *Deriving Real-Time Monitors from System Requirements Documentation.* PhD thesis, McMaster University, Hamilton ON, Jan. 2000.

[3] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing Principles, Algorithms and Applications.* Maxwell Macmillan, second edition, 1992.

[4] D. Simser and R. E. Seviora. Supervision of real-time systems using optimistic path prediction and rollbacks. In *Proc. Int'l Symp. Software Reliability Eng. (ISSRE)*, pages 340–349, Oct. 1996.

[5] A. J. van Schouwen, D. L. Parnas, and J. Madey. Documentation of requirements for computer systems. In *Proc. Int'l Symp. Requirements Eng. (RE '93)*, pages 198–207. IEEE, Jan. 1993.

[6] E. J. Weyuker. On testing non-testable programs. *The Computer Journal*, 25(4):465–470, 1982.