

Draft

Safety Checklist for Four-Variable Requirements Methods

Guy Helmer
Iowa State University
Department of Computer Science
Ames, IA 50011

January 22, 1998

Abstract

This paper organizes safety criteria, as given by Leveson, Jaffe, Heimdahl, Melhart, and Lutz, as a safety checklist for use on four-variable requirements models for real-time process-control systems. Special attention is given to the Software Productivity Consortium CoRE and Naval Research Laboratory SCR requirements specifications built on the four-variable model. The criteria are presented in English text and may be applied during the requirements engineering phase of software development to reduce safety-related software errors.

1 Introduction

This document organizes and applies the safety criteria defined by Software Requirements Analysis for Real-Time Process-Control Systems [3], Targeting Safety-Related Errors During Software Requirements Analysis [5] and Leveson's book Safeware [4] to four-variable models such as that used by Software Cost Reduction (SCR) [2] and the Software Productivity Consortium's CoRE [1].

1.1 Four-Variable Model

The four-variable model was defined by Parnas and Madey [6] to organize software design documentation in a way that is analogous to the design methods used by professional engineers in fields like electrical or chemical engineering. Design validation can only be done if the design is precise enough to use systematic analysis, but software design documentation has typically consisted of narratives that preclude precise analysis. Unfortunately, evaluation of a software design is then usually done after the coding stage, and corrections to the design are much more difficult and expensive than had corrections been made in the design stage.

The four-variable model documents a system by describing, in a mathematical language, the function of the system, where function means a mapping between two sets of elements, the domain and range, such that every element of the domain is mapped to exactly one element in the range. This contrasts with typical software design documentation that describes how the system works.

These four-variable methods state the requirements of a system in terms of the input, monitored, controlled, and output variables. More thoroughly stated, with help from Steve Tockey and Parnas' article [6]:

⁰This work was supported by Rockwell Collins Commercial Avionics, Cedar Rapids, Iowa.

Input The hardware input to the computer’s input registers provided by sensors, switches, and such.
For example, an input from a thermometer may be a 16-bit unsigned quantity directly from the input hardware.

Monitored Environmental quantities from inputs that are expressed in terms of the system domain vocabulary, i.e. what the user wants the system to measure.
Extending the thermometer example, the monitored variable **Temperature** may represent the degrees Fahrenheit reported by the thermometer.

Controlled Environmental quantities for outputs that are expressed in terms of the system domain vocabulary, i.e. the values that a system is expected to control.
For example, a controlled variable may specify a number of degrees of deflection up or down for an elevator on an airplane.

Output The computer’s output registers that provide values in terms that the hardware understands.
For example, a particular integer or floating-point value output to an actuator may set an airplane’s elevator to a particular angle.

Operations on the four-variables are specified as statements specifying the mapping from one variable to another:

IN Maps monitored variables to input variables; IN describes the behavior of the input devices.
For example, an IN relation may map an altitude in feet onto a particular integer or floating-point value from an input device.

REQ Maps controlled variables to monitored variables in environmental terms; REQ describes the expected behavior of the system.
For example, a flight-guidance system in altitude-hold mode may respond to a deviation of an observed altitude (monitored variable) from desired altitude (an internal state variable) by specifying a change in the elevator angle (controlled variable).

OUT Maps output variables to controlled variables; OUT describes the behavior of the output devices.
For example, when the flight-guidance system mentioned in REQ specifies a new elevator angle, the OUT statement maps the integer or floating-point value for the elevator hardware onto the elevator angle given by a controlled variable.

NAT Maps monitored variables to controlled variables; NAT defines the environmental context of the system that is being controlled.

The interactions in the four-variable model are as shown in figure 1.

As a final note on the background of the four-variable model, Parnas’ article [6] gives the mathematical foundation for the four-variable software engineering method as well as a list of required documents for a complete set of documentation that would fully define a computer system. Parnas notes that an early version of this model was used in 1977 for the A-7E aircraft, so this model has been used successfully for a number of years. The SCR [2] and CoRE [1] methodologies build on the Parnas and Madey foundation by providing either software tools or requirements development guidelines to develop a requirements model of a system in the four-variable format.

1.2 Safety Criteria Background

The criteria given here should be easily applied to any of the methodologies that build on the four-variable model, such as SCR and CoRE among others. The criteria are given in plain English, and are meant to be applied during the requirements phase of a software development project. If formal specifications are required, one may use the original predicate calculus given in the “Software Requirements” [3] article.

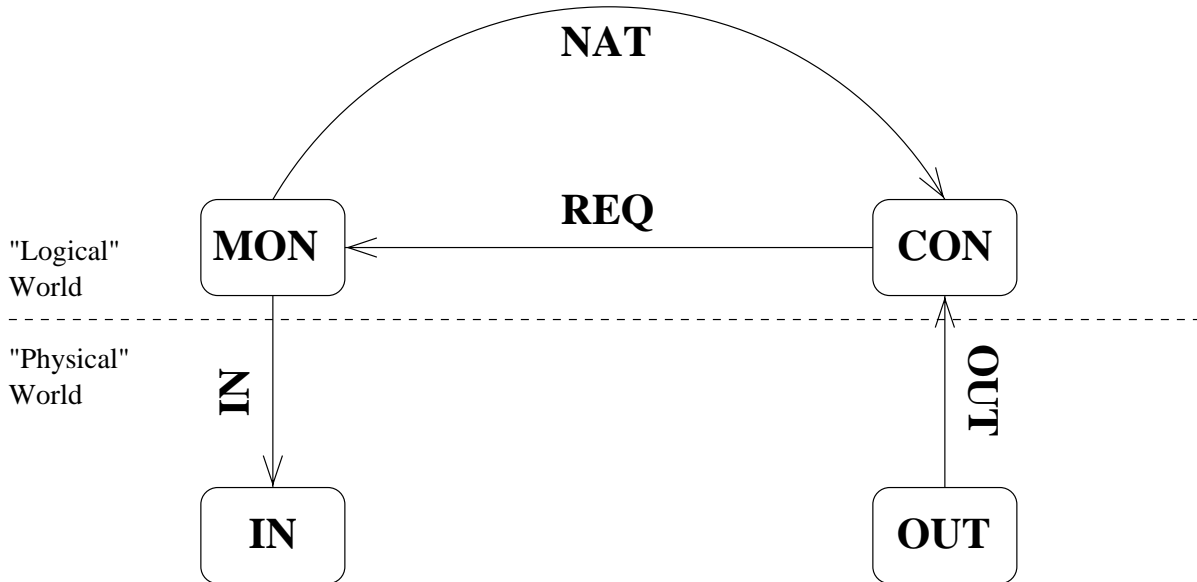


Figure 1: Graphical Depiction of Four-Variable Model

“Targeting Safety-Related Errors” [5] provides a similar English-text list of criteria in the form of a checklist to be applied to requirements for spacecraft systems. Similar to what is stated in Lutz’s article, the list of requirements given here may form a checklist to be used during development of requirements without needing to build a formal model, or the checklist may be used in conjunction with formal analysis. If these requirements are used as a checklist, they may be integrated into formal software engineering requirements review procedures as were Lutz’s “Safety Checklist.” [5]

Safeware [4] is a comprehensive discussion on safety and computers which goes far beyond the safety criteria. Chapter 15 expands on the list of criteria from “Software Requirements” [3] in a few places, and those additions are included in the lists below. John Rushby points out in “Formal Methods and their Role in the Certification of Critical Systems” [7] that the Jaffe and Leveson criteria are not exhaustive, but serve as a starting point that may be modified or extended as was done by Lutz [5].

1.3 Modes versus States

Since the criteria so often refer to states, the question arises as to whether states correspond to CoRE/SCR modes. The definition of states as used in “Software Requirements” [3] applies to a requirements state machine model using Mealy machines with logical of inputs and corresponding outputs on the transitions between states. This seems to match with the CoRE model’s use of mode machines to define the states of CoRE environmental variables [9, section 4.1.5]. Likewise, SCR defines modes whose transitions are triggered by events. So, for at least the CoRE and SCR requirements specifications, the criteria that apply to “states” can be thought of as applying to “modes”.

2 REQ

Criteria are grouped by categories as they are in Safeware [4]. Criteria marked with an asterisk (*) are satisfied by the CoRE and SCR implementations of the four-variable requirements model, as explained in section 5.

2.1 Startup and State Completeness

Leveson 15.4.2-1 The system and software must start in a safe mode. Interlocks should be initialized or checked to be operational at system startup, including startup after temporarily overriding interlocks.

Leveson 15.4.2-2 The internal software model of the process must be updated to reflect the actual process mode at initial startup and after temporary shutdown.

Leveson 15.4.2-3 (*) All system and local variables must be properly initialized upon startup, including clocks.

Leveson 15.4.2-5 Paths from fail-safe (partial or total shutdown) modes must be specified. The time in a safe but reduced-function mode should be minimized.

Leveson 15.4.2-6 Interlock failures should result in the halting of hazardous functions.

2.2 Input and Output Variable Completeness

Jaffe 4.1, Lutz 9 (*) Each monitored variable must be used.

Jaffe 6.1, 6.2, Lutz 9 Every mapping from controlled to monitored variables must have a behavior defined for every possible value of monitored variables.

Every value, both in-bounds and out-of-bounds, must be handled. If monitored variables are included that indicate out-of-bounds values of a monitored variable, then the above criteria (Jaffe 4.1) should satisfy this criterion.

Jaffe 6.3, Lutz 2 (*) Every state must handle timeouts (when input has not arrived within a specified interval or by a specified time).

Jaffe 6.4, Lutz 4 Transitions out of a mode must be deterministic based on the values of monitored variables.

Jaffe 6.5, Lutz 1 (*) Acceptable ranges of values must be specified for each monitored variable used to define the value or duration for some controlled variable.

Jaffe 6.9, Lutz 6 If a monitored variable indicates that a minimal arrival rate of input is not being met, the software should be able to query the system regarding inactivity over any distinct communication path.

2.3 Output Specification Completeness

Jaffe 7.1, Lutz 8 When a monitored variable indicates the input arrival rate exceeds input capacity, abnormal action (such as graceful degradation) may be required.

When output capacity is limited, the system must take special action when output capacity would be exceeded.

Jaffe 7.2, Lutz 5 Values of monitored variables must be bounded in time (each value has a data validity factor or age limit) when used to specify values of controlled variables.

Leveson 15.4.5-8 Incomplete hazardous action sequences (transactions) should have a finite time specified after which the software should be required to cancel the sequence automatically and inform the operator.

Leveson 15.4.5-9 Revocation of a partially completed action sequence may require:

1. Specification of multiple times and conditions under which varying automatic cancellation or postponement actions are taken without operator confirmation

2. Specification of operator warnings to be issued in the event of such revocation

Jaffe 7.3 When a controlled variable is to be set to indicate that an interval of time has passed without receiving an expected monitored input, a latency factor must be included in the time between the end of the timeout interval and the trigger of the controlled variable.

The value of the latency factor will be influenced by the frequency at which the system polls for inputs, how quickly it can react to interrupts, and how long it takes for the output to be generated; as a result, the latency factor can never be reduced to zero.

Leveson 15.4.5-11 Contingency action may need to be specified to handle events that occur within the latency period.

If action is taken based on the notion that a monitored input did not arrive within the latency period and it is then discovered that the monitored input did indeed arrive, corrective action may need to be taken.

Leveson 15.4.5-12 A latency factor must be specified for changeable human-computer interface data displays used for critical decision making. Appropriate contingency action must be specified for data affecting the display that arrives within the latency period.

If an operator enters an input just as the display changes, the system may need to confirm the operator's input.

Leveson 15.4.5-13 A hysteresis delay action must be specified for human-computer interface data to allow time for meaningful human interpretation. Requirements may also be needed that state what to do if data should have been changed during the hysteresis period.

Data which changes must remain constant long enough for meaningful human interpretation. An additional requirement may be needed if the situation arises where the data changes enough over the hysteresis delay so that the display would have changed but the operator made a decision based on the older data.

Jaffe 8.1, Lutz 11 If the desired response to an overload condition is performance degradation, the specified degradation should be graceful. Response time should not suddenly or erratically change.

Jaffe 8.2 If function shedding is used to manage overload conditions, a hysteresis delay and other checks must be included in the conditions required to return to normal processing load. Return to normal operation may be based on operator actions, reset messages from interfaces, and elapsed time. Temporal history of mode changes may be used to avoid entering a continual cycle between shedding and normal states.

2.4 Output to Trigger Event Relationships

Jaffe 8.3 Basic feedback loops, as defined by the process function F , must be included in the software requirements. That is, there should be a monitored variable that the software can use to detect the effect of a value of a controlled variable on the process control system. The requirements must include appropriate feedback on these monitored variables in order to detect internal or external failures or errors.

Jaffe 8.4 Every controlled variable of which a monitored variable measures the response must have associated with it:

1. A requirement to handle the normal response
2. Requirements to handle a response that is too late, too early, or has an unexpected value

Jaffe 8.5 Spontaneous receipt of a non-spontaneous monitored variable must be responded to (as an abnormal condition).

2.5 Specification of Transitions Between States

Jaffe 9.1, Lutz 14 All states must be reachable from the initial state, q_0 .

If a state is unreachable, either it is not necessary and may be removed, or the state should be reachable and the requirements must be adjusted.

Jaffe 9.2 Desired recurrent behavior must be part of at least one cycle (in the state machine).

Jaffe 9.3 Reversibility - If a controlled variable setting may be canceled, the state where the controlled variable is reset must be reachable from the state where the controlled variable was set.

Leveson 15.4.7-6 Preemption requirements must be specified for any multi-step transactions in conjunction with all other possible control activations.

For example, an action to recompute estimated time of arrival might be prohibited until an in-progress, manual navigation updated is completed or canceled.

Jaffe 9.4 There should be no paths to undesired hazardous states (unless necessary to perform actions of the system).

The system may enter hazardous states by transforms not initiated by the computer, such as software or equipment failures, human errors, environmental disturbances, or stress.

Leveson 15.5-2 Reachable hazardous states should be eliminated, or, if that is not possible (they are needed to achieve the goals of the system), their frequency and duration reduced.

Jaffe 9.5, Lutz 15 Every path from a hazardous state must lead to a safe state.

Even though a hazardous state may not be reached through the software, hazardous states may be entered due to non-software factors. The software must then provide paths to safe states from hazardous states.

Jaffe 9.6, Lutz 15 If hazardous state cannot reach a safe state, all paths from the hazardous state must lead to minimum risk states.

If a hazardous state is entered and a transition can not be made to a safe state, a transition should at least be made to a state with minimum risk.

Jaffe 9.7 Soft and hard-failure modes should be eliminated for all hazard-reducing outputs.

Leveson 15.4.7-8 Multiple paths should be provided for state changes that maintain or enhance safety. Multiple inputs or triggers should be required for paths from safe to hazardous states.

It should be possible to reach a safer state in more than one way so that a single hardware failure cannot prevent the operator from taking action to move the system to a safer state. On the other hand, multiple independent inputs should be required to move the system from a safe state to a hazardous state, such as a fighter pilot launching a missile (where the pilot has to arm and then fire the missile as separate actions).

A criteria that is not stated separately by Jaffe and Leveson [3] (though it is mentioned in the text after criteria 6.6) is: all timing bounds must not be tighter than the precision of the reference clock.

3 IN

IN relations define the correspondence between physical system inputs and the monitored variables over time, perhaps such as the translation of an analog electric signal from a device into its internal digital (binary) representation. Criteria marked with an asterisk (*) are satisfied by the CoRE and SCR implementations (see section 5).

3.1 State Completeness

Jaffe 4.1, 6.1, 6.2, Lutz 9, Leveson 15.4.3-1 (*) Each physical input should have an IN relation defining its relationship to monitored variables, and out-of-bounds values may be noted in associated monitored variables.

Jaffe 5.1, Lutz 10, Leveson 15.4.2-2 Input received before startup, after shutdown, or while off-line must be acknowledged or determined that it can be safely ignored.

Issues here involve resynchronization of the computer to the process and appropriately dealing with input that may have been missed while off-line. Also note that after startup, the system must be able to handle the condition where the initial input does not arrive within an expected time.

3.2 Input and Output Variable Completeness

Jaffe 6.3, Lutz 2 (*) Timeouts must be noted by IN statements.

Jaffe 6.5, Lutz 1 (*) Acceptable ranges of values must be specified for each input.

Jaffe 6.6, Lutz 5 (*) All inputs must be fully bounded in time (inputs may arrive too soon or too late), or the bounds must be implied by the specification.

This seems to indicate that queued values should be marked with a time stamp.

Jaffe 6.7, Lutz 2 If an action is to be taken when an input times out, the interval that describes the time period where the input is expected must have explicit lower and upper bounds in time ($t > t_{start} + t_{dur}$), where t_{start} and t_{dur} are calculable from observable events or expressed as absolute time.

Jaffe 6.8, Lutz 6, 7 An interrupt-signaled event that is at any time un-dominated requires a capacity assumption.

Interrupts may be disruptive to computation if not un-dominated; whether interrupt capacity is part of IN relations is not clear to the author.

Jaffe 6.9, Lutz 6 IN should be able to notify REQ when the minimum arrival rate is not being met.

Jaffe 7.1, Lutz 8 An input arrival rate exceeding input capacities must signal the system that abnormal action may be required (such as graceful degradation).

Jaffe 7.2, Lutz 5 (*) All inputs used in specifying output events must be bounded in time (each input has a data validity factor or age limit).

Similar to notes for criteria 6.3 and 6.6; inputs should be given a time stamp, and obsolete data should not be used to determine outputs.

Jaffe 8.3 (*) Basic feedback loops, as defined by the process function F , must be included in the software requirements. That is, there should be a monitored variable that the software can use to detect the effect of a value of a controlled variable on the process control system. The requirements must include appropriate feedback on these monitored variables in order to detect internal or external failures or errors.

Jaffe 8.5 - Spontaneous receipt of a non-spontaneous input must be detected.

4 OUT

OUT defines the correspondence between controlled variables and physical system outputs over time, e.g. such as the translation of a value of a controlled variable into an analog electric signal to drive a motor. Criteria marked with an asterisk (*) are satisfied by the CoRE and SCR implementations (see section 5).

4.1 Output Specification Completeness

Leveson 15.4.3-2 Legal output values that are never produced should be checked for potential specification incompleteness.

It is likely there is an omission from the specifications if there is a legal output that is never produced.

Leveson 15.4.5-1 Safety-critical outputs should be checked for reasonableness and for hazardous values and timing.

Jaffe 7.1, Lutz 8 The output capacity, if limited by behavioral or physical constraints, should be specified.

Jaffe 7.2, Lutz 5 Obsolete data should not be used to generate outputs.

Jaffe 8.3 (*) Basic feedback loops, as defined by the process function F , must be included in the software requirements. That is, there should be a monitored variable that the software can use to detect the effect of a value of a controlled variable on the process control system. The requirements must include appropriate feedback on these monitored variables in order to detect internal or external failures or errors.

5 Criteria Satisfied by Four-Variable Requirements Model

Some of the safety criteria are inherently satisfied in implementations of the four-variable requirements models CoRE and SCR. In particular, CoRE's specification for requirements includes tolerance (range of values), initiation delay and completion deadline (timing), and initial values for controlled variables.

5.1 REQ

Leveson 15.4.2-3 All system and local variables must be properly initialized upon startup, including clocks.

CoRE requires initial values and initiating events for all variables.

Jaffe 4.1, Lutz 9 Each monitored variable must be used.

CoRE requires that each variable has a relation defined on it.

Jaffe 6.3, Lutz 2 Every state must handle timeouts (when input has not arrived within a specified interval or by a specified time).

CoRE specifies REQ relation timing, which is required to be consistent with the delays allowed by the IN and OUT relations.

Jaffe 6.5, Lutz 1 Acceptable ranges of values must be specified for each monitored variable used to define the value or duration for some controlled variable.

In CoRE, tolerance is given for each monitored variable.

5.2 IN

Jaffe 4.1, 6.1, 6.2, Lutz 9 Each physical input should have an IN relation defining its relationship to monitored variables, and out-of-bounds values should be noted in other monitored variables.

The CoRE Guidebook [9] suggests in section 4.3.3 that in some cases additional monitored variables are created to denote undesired events.

Leveson 15.4.3-1 All inputs from the sensors should be transformed to a monitored variable.

CoRE requires that each variable has at least one relation defined on it.

Jaffe 6.5, Lutz 1 Acceptable ranges of values must be specified for each input.

CoRE requires that the tolerance for each variable be stated.

Jaffe 6.6, Lutz 5 All inputs must be fully bounded in time (inputs may arrive too soon or too late), or the bounds must be implied by the specification.
CoRE requires the delay for inputs be specified.

Jaffe 7.2, Lutz 5 All inputs used in specifying output events must be bounded in time (each input has a data validity factor or age limit).

Jaffe 8.3 Basic feedback loops, as defined by the process function F , must be included in the software requirements. That is, there should be a monitored variable that the software can use to detect the effect of a value of a controlled variable on the process control system. The requirements must include appropriate feedback on these monitored variables in order to detect internal or external failures or errors.

The inclusion of this requirement in REQ should lead to its satisfaction in IN and OUT due to the requirements for a monitored variable to detect the effect of a controlled variable on the process.

5.3 OUT

Jaffe 8.3 Basic feedback loops, as defined by the process function F , must be included in the software requirements. That is, there should be a monitored variable that the software can use to detect the effect of a value of a controlled variable on the process control system. The requirements must include appropriate feedback on these monitored variables in order to detect internal or external failures or errors.

The inclusion of this requirement in REQ should lead to its satisfaction in IN and OUT due to the requirements for a monitored variable to detect the effect of a controlled variable on the process.

6 Comments on and Enhancements for Four-Variable Model

It is assumed that an implementation of the four-variable model contains procedures to be used during development to assure that the operations on the variables (monitored to in, controlled to monitored, out to controlled) are complete in that no variable is unused.

Timing requirements in the four-variable model present a problem because of the three different timing domains (IN, REQ, OUT) involved between receipt of an input and generation of an output. CoRE specifically requires that the timing requirements given in REQ be consistent with the delays given in IN and OUT.

7 Acknowledgments

Thanks to Robyn Lutz for providing direction, support, and all of the supporting documentation on safety requirements.

Thanks to Steve Tockey of Collins Avionics for aiding my understanding of the four-variable requirements model and providing feedback on the development of this document, and to Rockwell Collins Avionics for funding this work.

References

- [1] Stuart Faulk, John Brackett, Paul Ward, and James Kirby, Jr. The Core method for real-time requirements. *IEEE Software*, pages 22–33, September 1992.
- [2] IEEE. *SCR*: A Toolset for Specifying and Analyzing Requirements*, 1995.

- [3] Matthew S. Jaffe, Nancy G. Leveson, Mats P.E. Heimdahl, and Bonnie E. Melhart. Software requirements analysis for real-time process-control systems. *IEEE Transactions on Software Engineering*, 17(3):241–257, March 1991.
- [4] Nancy G. Leveson. *Safeware*. Addison-Wesley Publishing Company, 1995.
- [5] Robyn R. Lutz. Targeting safety-related errors during software requirements analysis. *Journal of Systems and Software*, 34:223–230, 1996.
- [6] David Lorge Parnas and Jan Madey. Functional documents for computer systems. *Science of Computer Programming*, 25(1):41–61, October 1995.
- [7] John Rushby. Formal methods and their role in the certification of critical systems. Technical Report SRI-CSL-95-1, Computer Science Laboratory, SRI International, Menlo Park, CA, March 1995. Also available as NASA Contractor Report 4673, August 1995, and to be issued as part of the *FAA Digital Systems Validation Handbook* (the guide for aircraft certification). Reprinted in [8, pp. 1–42].
- [8] Roger Shaw, editor. *Safety and Reliability of Software Based Systems (Twelfth Annual CSR Workshop)*, Bruges, Belgium, September 1995. Springer.
- [9] Software Productivity Consortium Services Corporation, SPC Building, 2214 Rock Hill Road, Herndon, VA 22070. *Consortium Requirements Engineering Guidebook*, December 1993.