

Developing Safety-Critical Systems

Bill Petit

www.billpetit.com

This article was printed in the 2006 Railway Age C&S Buyers Guide.

Although processor-based systems have been safely used in the rail industry for over a quarter century, there has recently been a lot of discussion on what constitutes a safety-critical (or a vital) system, primarily because of the publication of the FRA rule on processor-based signal and train control systems. This article explores some of the background and steps necessary to develop such a system. Much of the information comes from texts such as “SAFWARE; System Safety and Computers” by Dr. Leveson and “Safety-Critical Computer Systems” by Storey as well as IEEE standards (IEEE 1483-2000) and AREMA Recommended Practices (Manual Part 17). Readers are strongly encouraged to review this material for a more in-depth understanding.

Processor-based safety systems have come under scrutiny across all industries over the past couple decades due to many well-publicized accidents resulting from their use. It's generally accepted that some of the key reasons for this is that systems being developed are exceedingly complex and tightly coupled. This leads to more error-prone designs, inability to adequately analyze failure possibilities, and more accidents, as tightly coupled systems don't allow failures to be contained and mitigated before they spread into other parts of the system.

General guidance for developing safety-critical processor-based systems consists of the following:

- Avoid complexity in the design. Simplicity allows a more complete understanding of the system operation minimizing the chances for error, and makes the designs easier to test. Drastically reducing the number of functions performed by the safety-critical system (and enforcing this reduction) also keeps function creep under control and allows future changes to be made and verified. A good criteria to keep in mind is “Just because you can doesn't mean you should”.
- Use deterministic design techniques. It is necessary for the processor-based system to operate the same way every time a critical function is performed. This is necessary to allow comprehensive testing of the system. If different timing or alternative paths through the logic due to multi-tasking, dynamic memory allocation, or other characteristics exist, all possible execution paths cannot be adequately tested.
- Separate safety-critical from non-safety-critical portions of the system. This allows simplicity to be maintained in the safety-critical portion of the system

so that it can be adequately tested and maintained. Since software exhibits weak-link behavior (where software in an unrelated portion of the system can adversely affect a safety-critical portion), this separation reduces the required analysis needed.

The last bullet above has been successfully applied to the rail industry for over a century. Protection systems (e.g. signal systems) are used as a relatively simple system to ensure the safety of operations systems (e.g. dispatch systems, meet/pass planners, pacing guidelines). Interestingly, this separation is known as “interlocks” in other industries so, once again, the railroad industry is leading the way.

Over the years, some specific guidance for the rail industry for safety-critical systems has also evolved. Other industries follow similar guidelines but refer to them with different terminology. These consist of:

- Apply the closed loop principle. This simply means that permissive actions can only be allowed when all conditions necessary for them to exist are proven to exist, and are continually proven to exist for the length of the allowed permissive action. This sounds simple enough but there have been incidents in the rail industry where permissive actions were allowed to exist only because there had been no external notification that conditions had changed (i.e. waiting for someone to tell you to stop rather than continually verifying that it is OK to proceed).
- Design according to the fail-safe principle. This is a design principle where all possible failures in a system are analyzed (either hardware, software or interface). Occurrence of any of these failures cannot allow the system to permit unsafe conditions to exist either by forcing the system to a safe state, or transferring control to a backup system to continue safe operation.

Now that we’ve covered some general principles, some specific tools and techniques for analyzing a system will be discussed. Note that these always start from the beginning of the development and address the system as a whole. It’s not practical (and most believe it’s not possible) to attempt to make a system safe after it has already been developed.

This discussion will be limited to analyzing the safety-critical design. These steps must be integrated with comprehensive quality management and configuration management to ensure that the system is correctly built, maintained and operated. Quality design attributes are not included in this list. It’s assumed that those steps required for quality design (proper functional requirements and operation, documentation, requirement tracking and Design Verification & Validation) are done properly. The following steps must be done for safety-critical systems development and are limited to that portion of the system development lifecycle. They are initially listed below and then described in additional detail later.

- Develop a system concept of operations

- Identify an initial set of hazards that the system will protect against.
- Identify the architecture that will be used to implement the system along with its critical assumptions and dependent factors.
- Add hazards resulting from architecture choice to list of hazards that must be mitigated.
- Maintain hazard log for all hazards identified throughout the system including identifying the processes for verifying that the hazards have been effectively mitigated.
- Assess each hazard for risk (probability and likelihood).
- Determine methods to be used for mitigating each hazard identified.
- Complete product design and test to verify functionality.
- Use Fault Tree Analysis techniques to analyze design from the top down in order to identify functional faults that could cause an unacceptable or undesirable hazard.
- Use Implementation Fault Trees to identify specific hardware and software components that could create a functional fault. Provide verification that each of these areas has been mitigated per your design strategy and architecture choice.
- Use Failure Mode and Effect Analysis techniques to analyze design from the bottom up to uncover failures that could create unacceptable or undesirable hazards.

A system concept of operations provides a good frame of reference for the entire project. Actual requirements need to be stated formally but this document should be written in an informal style allowing the user to gain an overall view of what the system is intended to accomplish.

Identifying, assessing and mitigating hazards is critical to the analysis of the system. Hazard Analysis techniques were first addressed in Military standards and both Mil-Std 882C and IEEE Standard 1474.1-2004 provide excellent descriptions of the various analyses. Hazard Analysis starts with the Preliminary Hazard List (PHL), which identifies the top-level hazards that the system is intended to address. For example, a signal system would have top-level hazards such as collision or derailment.

Starting from the PHL, a Preliminary Hazard Analysis (PHA) is begun. Fault tree techniques are used to identify lower level hazards contributing to the hazards identified previously. Based on functional requirements, application requirements, and design choices, these lower level hazards are identified using a combination of expert knowledge, previously identified hazards from other projects, fault tree analysis, brainstorming and previously generated checklists. Each hazard identified through this process must be tracked throughout the project and verified that it has been properly mitigated. This is typically done through a hazard log that tracks all the hazards, and references a specific location (e.g. a hazard sheet) where all the information regarding the hazard is maintained. These hazard sheets also include a risk analysis (to be discussed later) and identify techniques to be used for mitigating the hazard.

As noted before, the specific architecture chosen for the design contributes hazards to the PHA. IEEE 1483-2000 identifies some of the architectures used in safety-critical designs along with some of their critical assumptions and dependent factors. These assumptions and dependent factors must be addressed in the PHA. That standard identifies the following architectures as generally accepted, although recognizing additional architectures may be identified in the future. Most systems use a combination of the following architectures. Some representative critical assumptions and dependent factors are included here, additional information can be found in the IEEE standard.

- Intrinsic Fail-Safe Design. This is generally used for discrete mechanical or electrical components where the credible failure modes of all components can be directly analyzed to ensure no unsafe conditions are created as a result of failure. Most systems use this technique as some portion of their system design, particularly for Input/Output or comparison mechanisms. AREMA MP 17.3.3 identifies a minimum set of credible failure modes to be included in the analysis.
- Checked Redundancy. This technique uses 2 or more identical, independent hardware units executing identical software and performing identical functions. This is more generally known as Two out of Two (2oo2) or Two out of Three (2oo3) systems and is used in some European signal systems.
 - Critical Assumption. Independent hardware failures in each of the redundant units that produce the same erroneous unsafe effect will not occur simultaneously.
 - Dependent Factors.
 - Safety-Critical characteristics of the comparison mechanism.
 - Identification of all safety-critical functions and degree to which they are cross-checked (e.g. only cross-checking final result is not sufficient).
 - Frequency of checking process.
 - Elimination of common-mode factors.
 - Ability to place and maintain the system in a known safe state.
- N-version programming. This technique requires at least two software programs, executing together and performing identical functions. An independent team using independent tools must write each software program. They may or not run on independent hardware platforms.
- Diversity and Self-Checking. This technique generally operates on a single processor and requires that all critical functions be performed in diverse ways, using diverse software operations and/or diverse hardware channels, and that critical hardware be tested with self-checking routines. This is the most common technique used in North American signal systems.

- Dependent Factors
 - Identification of all safety-critical functions and verification that they are implemented diversely.
 - Ability of checking process to compare diverse results and provide permissive outputs only if they agree.
 - Comprehensiveness of the self-checking process.
- Numerical Assurance. This technique requires that the state of each safety-critical parameter be represented by a large encoded numerical value. Off-line data structures are defined such that real-time permissive results can only be calculated (by pseudo-randomly combining these values) when all the proper conditions for a permissive result are present.

In addition to the PHA, other analyses are used to identify hazards that must be added to the hazard log. For example the Sub-system hazard analysis (SSHA) looks at each subsystem or component within the system and identifies hazards associated with operating or failure modes. The System Hazard Analysis (SHA) examines all the subsystem interfaces and interfaces with other systems. (This is also known as an Interface Hazard Analysis). The Operating and Support Hazard Analysis (O&SHA) looks at the potential hazards of human-machine interfaces involved in the operation, testing and maintenance of the system.

As part of the hazard log, each hazard is assessed for risk. Hazards are classified according to their overall risk depending on their severity and probability. Severity ranges from Catastrophic through Negligible (4 categories total) and Probability ranges from Frequent through Improbable (5 categories total). AREMA MP 17.3.5 identifies guidelines for assigning hazards to each of these categories as well as overall risk. For example, “Frequent” implies that it will happen more than once every 1000 hours and “Improbable” implies that it will happen less than once every 10^9 hours. When each hazard is assigned a Severity and Probability Category, it is assigned an overall risk from “Unacceptable” to “Acceptable” depending upon its position in a matrix. As part of the analysis, justification for the Frequency assignment must be provided. It is important to note that these numbers are not simple reliability numbers, they include intermittent and design faults also. According to most standards, hazards that have risk ratings of “Unacceptable” or “Undesirable” must be mitigated (i.e. reduce the risk, which is generally done by decreasing the frequency of occurrence) through system and equipment design.

The majority of the above steps are completed prior to detailed product design. Hazards will continue to be added to the Hazard log throughout the design process as they are identified.

As noted earlier, attributes such as quality design, requirements tracking and design verification and validation are critical components of a safety-critical design. However, this article focuses on specific steps necessary for safety-critical aspects of the design in

addition to the quality aspects. Substantial information exists elsewhere to address these other areas.

As part of the safety V&V, it is necessary to provide verification that all of the identified unacceptable or undesirable hazards have been properly mitigated. In order to do this, all of the safety-critical functions necessary to implement the system (down to a very low level) must be identified. Functions that have to be implemented so that they are both performed and performed correctly must be implemented fail-safely (and are identified as vital functions in IEEE 1483). The fail-safe implementation means that you look at all the credible failures that could occur and make sure that occurrence of any one of them (or combination of failures in the event that the first failure is not self-evident) maintains the system in a safe state, either by forcing the system to a stop (or other safe state such as a less permissive signal) or by transferring control to a secondary system (e.g. redundant computer). Some people refer to this transfer to a redundant system as fail-operational, but it is important to note that the fail-safe design philosophy is still necessary to identify when system transfer is initiated.

Functional Fault Trees are a widely accepted tool for identifying the safety-critical functions from the top down. These trees start at the upper level hazards previously identified and branch down through the system, subsystem and interfaces to identify all of the functions that, if not performed correctly, could precipitate the upper level hazards. The functions at the base of the tree are the lowest level functions that can be defined and cannot be subdivided further.

Having identified these low level functions, a final fault tree is performed to identify the specific hardware and software components used to implement the function. These detailed hardware and software components are then verified as having been implemented fail-safely. These verification techniques are generally specific to the design techniques used.

The functional fault trees provide a top-down analysis of the system. A Failure Modes and Effect Analysis (FMEA) is also used to provide a bottom-up analysis of the system. Starting at the component level, all credible failures are analyzed to verify that they do not create any unsafe condition. AREMA Manual Part 17.3.3 provides guidance on credible component failure modes. As with all safety analyses described so far, secondary failures must be considered in combination with the initial failure if the initial failure is not self-revealing.

The major goals of the process described above for the development and verification of safety critical systems is to provide a systematic approach that.

- Minimizes errors of omission,
- minimizes errors of commission,
- Is complete and comprehensive,
- Is visible (allows others to understand what was and what was not done),
- Is consistent and unambiguous, and

- Results in a system with identifiable and acceptable risk.

This article provides a top-level review of some of the aspects of safety-critical design. Interested readers are strongly recommended to review the available standards and textbooks for a more in-depth understanding.