An Engineering Approach to Critical Software Certification

Robert E. Loesh  
*Software Engineering Sciences, Inc.*  
Covina, CA 91724  
rloesh@computer.org

Arthur B. Gosnell  
Thomas F. Benoist  
*Software Engineering Directorate*  
*U.S. Army Aviation and Missile Command*  
Redstone Arsenal, AL 35898  
bgosnell@sed.redstone.army.mil  
tbenoist@sed.redstone.army.mil

Richard M. Wyskida, Ph.D., PE  
*The University of Alabama in Huntsville*  
Huntsville, AL 35899  
wyskidar@email.uah.edu  
johannesj@email.uah.edu

Abstract

A significant number of computer-based systems currently exist that perform human-related critical functions (e.g., medical diagnosis/treatment support, weapon systems, banking financial systems, transportation systems). The Software Engineering Directorate (SED) of the United States Army Aviation and Missile Command (AMCOM) develops and maintains numerous critical systems that fall into this category. Prior to operational deployment of AMCOM aviation and missile systems, the SED has responsibility for the certification of these critical systems. This paper will describe a four pronged engineering-based analysis for determining a Flight Readiness Risk Index and a Certification for Flight Readiness recommendation.

Developers of critical systems have generally approached the certification for flight readiness by performing an independent audit of the development process and technical standards. While the independent audit is a good step towards determining operational readiness, a more rigorous analysis of the software and its execution is necessary to establish high confidence in the system deployment decision. The SED is developing a procedure utilizing an engineering-based approach which permits the determination of the operational worthiness of these systems. The engineering-based approach addresses four analyses that determine the following:

- **Software Reliability**: Measured by frequency of failure
- **Correctness**: Measured by error experience and test coverage
- **Safety**: Measured by resistance to hazards
- **Operational Usability**: Measured by operational scenario-driven automated testing, user testing, and/or operational user simulation results

It has been standard procedure to audit the development process and engineering standards used for the development of these systems as well as perform in-process oversight (i.e., Independent Verification and Validation (IV&V)) of the software. The purpose of this standard audit is to determine if the approved best practices and established...
engineering standards suitable for man-rated, critical systems were used during development. The objective of the IV&V is to ensure that the correct, safe, and most reliable software design is implemented and tested. However, the SED continually works to improve internal analyses to assist in elevating the level of confidence in system performance and in safe and reliable operations upon deployment. As a result of SED research and development of analyses methods, a new procedure is being developed for determining the software’s suitability for airworthiness that is based on the rigorous engineering approach described in this paper. The newly developed procedure utilizes a set of four complimentary engineering analyses which quantify the risks and leads to a recommendation concerning software airworthiness. The four engineering analyses are:

1. Software reliability analysis - determines the frequency of error for each of the Error Priorities 1 through 5 as defined in the DoD-STD-2167A [2] during CSCI qualification testing, system integration and testing, operational test and evaluation.
2. Correctness analysis - assess the error experience and code test coverage during the code and unit test (CUT) and/or CSC integration and testing.
3. Safety analysis - determines the software’s design to resist hardware and input anomalies (utilizing hazard analysis methods).
4. Operational usability analysis - determines whether the system/software performs correctly and reliably as commanded or requested (using automated simulation and/or user testing, and/or user operational scenario simulation).

In each of the four analyses, a Risk Index (RI) of High, Medium, or Low is determined based upon software measurements and a criteria established by the procedure for assigning the Risk Index. The initial recommended criteria is based upon heuristics, expert opinion, documented studies, reports, and policies. All of the recommended criteria may not be from controlled studies but represents the judgements of the staff developing the certification process. Each section of this paper will include a brief discussion addressing the rational for criteria selection together with supporting reference studies where appropriate.

Among the benefits of formalizing this procedure is to initialize a common set of criteria and the opportunity to analyze and evaluate the procedure results with system operational experience, leading to the refinement of the criteria and the set of analyses necessary to assess and recommend certification for flight readiness.

The criteria for recommending deployment and the software’s suitability for airworthiness are a composite of the results of the engineering analysis risk indices. After the data is collected and individually assessed for each analysis, the four risk indices provide a recommendation for certifying the software airworthiness. If the software is deemed high or medium risk, based upon the individual analysis, specific action to reach a lower RI and a more positive airworthiness recommendation is possible.

2. Software Reliability Analysis

The reliability analysis addresses the likelihood that the software will perform a reasonably useful and safe function to completion and focuses on the performance of the software at the integrated Computer Software Configuration Item (CSCI) level and beyond. For this procedure, software reliability is defined as the measurement of the frequency of errors per Test Series where the series of tests is one of the following:

Test Series-1: All of the software item qualification testing as specified in J-STD-016-1995 [3]. All safety requirements identified in the Software Requirements Specification (SRS) must be specifically tested.
Test Series-3: Operational Test and Evaluation (OT&E). Special emphasis shall be given to safety critical requirements [5].

The reliability data is similar to the often-used error density ratio except that the ratio is not errors per lines of code but rather errors per Test Series. The procedure requires, as a minimum, the execution and collection of error data from both the CSCI and System test activities. It is recommended that the operational level testing also be performed to provide additional data fidelity. From a safety and correctness viewpoint, the validation of the software to meet mission objectives, while executing in a deployed or simulated deployed mode, is significant [6]. The collection and recording of errors for each Test Series by Error Priority is evaluated after the Test Series is complete and then assessed to determine the RI.

Equation 1 is recommended to arrive at an RI per specific Error Priority:

\[
RI_j = L_{ij} + 2L_{ij} + 4L_{ij}
\]

where:

- \( L_{ij} \) is the number of Priority j (j = 1, 2, 3) errors found in Test Series i (i = 1, 2, 3).
- \( RI_j \) is the calculated Risk Index for a specific Error Priority j.
This basic information is normally represented in matrix format as shown in Table I. The entries in Table I represent the differential weighting applied depending upon the Test Series which is being performed. These weights significantly penalize errors discovered in Test Series performed well into the software development program.

Table I. Generalized format

<table>
<thead>
<tr>
<th>Test Series (i)</th>
<th>Error Priority (j)</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

The RI is determined for each of the three significant error priorities for the performed Test Series and then each of the three Risk Indices is assessed using the entries in Table II. The highest Risk Index is assigned when more than one criteria is satisfied.

Table II. Error priority risk indices

<table>
<thead>
<tr>
<th>Risk Index</th>
<th>Criteria</th>
</tr>
</thead>
</table>
| High       | $R_{i_j} > 6$  
               $R_{i_2} > 3$  
               $R_{i_3} > 6$  |
| Medium     | $2 \leq R_{i_j} \leq 3$  
               $3 \leq R_{i_2} \leq 6$  |
| Low        | $R_{i_j} < 2$  
               $R_{i_2} < 3$  |

Table III presents a hypothetical situation. The entries in ( ) represent the errors detected during the respective Test Series. The respective RI’s are calculated as the sum of the products for each error priority as shown in the RI row of Table III. Utilizing Table II with the RI value for Error Priority 1 (RI₁), a Risk Index of High results from Table II. For Error Priority 2 (RI₂), a Risk Index of Low results, while Error Priority 3 (RI₃) results in a Risk Index of Medium. Consequently, the overall Risk Index is High.

Table III. Typical calculations

<table>
<thead>
<tr>
<th>Error Priority (j)</th>
<th>Test Series (i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

The rationale for the error priority weights of 1, 2, or 4 in Equation 1 is based on the notion that software reliability (correctness) increases with a decrease in the error density resulting from testing in a more complete system configuration together with additional operational-like scenarios. The 1, 2, and 4 weightings reflect an inverse slope of the desired error rates as specified in [7].

The rationale for the Risk Index and criteria specified in Table II is based upon numerous review board meetings that suggest similar criteria were frequently utilized as a basis for a ‘decision to proceed’ with software utilized for training and/or operations. The rationale/criteria and details for review board decisions are normally undocumented and therefore unavailable for analysis and study. This procedure will provide for such analysis, study, and confirmation or refinement.

3. Correctness Analysis

The correctness analysis complements the reliability analysis by addressing the software defect experience at the Computer Software Unit (CSU) level. Critical CSUs are those that compose CSCIs that are deemed critical for safe and reliable operation. This procedure indicates whether the detailed code functions performed are sufficiently correct to meet the software design in response to requirements and mission objectives.

The Correctness Analysis of the software is measured by the error detection experience (errors reported during CSU testing) and the percent of test coverage during CSU testing [8]. The analysis requires that the paths traversed through the executable code are monitored during the CSU
testing and the observed errors are recorded.

The testing at the CSU level can be the Code and Unit Testing and/or the CSC Integration and Testing as specified in DoD-STD-2167A. The error detection and correction rates from both of these tests reflect the correctness of the CSU design and code. However, it is ‘atypical’ for software development projects to record or track error detection and correction during code and unit testing. By the same token, it is typical to track and capture error detection and correction data after a CSU has completed Code and Unit Test, the CSU has been placed in a software development library, and the CSU exists under some form of configuration management. When only CSC Integration and Testing error data are available, the analysis should proceed with the available data.

The Correctness Analysis procedure calls for the use of a tool (e.g., McCabe & Associates) that will trace the paths traversed through the code during Code and Unit Testing and/or CSC Integration and Testing. From the work performed and reported by Slonim [9], significant error detection begins when about 25% of the code paths have been traversed. The testing should continue until at least 50% of the paths have been traversed and until the rate of error detection falls off significantly. The results of the CSU testing [7] are used along with the code testing coverage data to arrive at an RI for code correctness based upon Table IV.

### Table IV. Code correctness risk index

<table>
<thead>
<tr>
<th>Risk Index</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High</strong></td>
<td>Any Priority 1or 2 errors detected, not corrected, and not validated prior to completion of the CSU testing. and/or 50% or less of the code traversed during the CSU testing for the critical CSUs. and/or Any errors detected in the last 10% of the CSU tests not previously executed.</td>
</tr>
<tr>
<td><strong>Medium</strong></td>
<td>More than 50% but less than 60% of code traversed during testing for the critical CSUs. and/or Any errors detected in the last 15% of the CSU tests not previously executed on the critical CSUs.</td>
</tr>
<tr>
<td><strong>Low</strong></td>
<td>60% or more of code traversed during the CSU testing for the critical CSUs. and/or No errors detected in the last 15% of the CSU tests integration not previously executed on the critical CSUs.</td>
</tr>
</tbody>
</table>

The highest risk index is assigned when more than one criteria is satisfied.

The rationale for the Risk Index and Criteria specified in Table IV is the same as that utilized in the Reliability Analysis for Table III. This analysis and the corresponding criteria are utilized infrequently based upon a paucity of data and experience. It is hypothesized that as software development projects become aware of this correctness analysis procedure, test coverage results will be included in the software development process.

### 4. Safety Analysis

The safety aspects of software are composed of two issues [10]. The first issue is whether or not inputs and commands, when processed by the software, achieve the desired, expected, and safe results. The second issue is whether non-anticipated inputs and commands, when processed by the software, permit the desired, expected, and safe results to be achieved. The first issue is addressed as part of the other three analyses. The second issue is determined by performing a system/software hazard analysis.

The objective of the hazard analysis is to ensure that the software will perform the designed and coded processing while satisfying the following two criteria:

1. The software produces the desired, expected, and safe outputs and results.
2. The software leaves the processing and data in a state that ensures additional processing that meets the item 1 criterion is possible without human intervention.

Ideally, the system Preliminary Hazard Analysis (PHA) has been completed prior to the requirement for a software hazard analysis. This simplifies the effort to accomplish the software hazard analysis. In any event, a software hazard analysis is performed on the critical software by following the general procedure specified in MIL-STD-1629A [11] with some tailoring for software. The SED has a procedure with a supporting workshop which is included in the Software Engineering Evaluation System (SEES) that implements MIL-STD-1629A, i.e., Software Failure Modes, Effects, and Criticality Analysis, SED-SES-SAP-FMAA-001 [12].

The SEES hazard analysis procedure results in the completion of the SEES Software Failure Modes, Effects, and Criticality Analysis Form (SFMECAF). A typical SFMECAF that has been completed is shown in Figure 1. Upon completion of the SFMECAF, the Criticality column is reviewed to determine which hazards have a criticality assessment other than none. The criticality assessments that can be assigned are specified in Table V.
### SEES Software Failure Modes, Effects, and Criticality Analysis Form (SFMECAF)

**Program ID**: Sure Shot Missile  
**Technical Lead**: J. Micom  
**System**: Missile Guidance

<table>
<thead>
<tr>
<th>RRLF Item No.</th>
<th>Interface Data</th>
<th>System Hardware Interface</th>
<th>Software element</th>
<th>System failure mode</th>
<th>Effects</th>
<th>Criticality</th>
<th>Comments/Rec. sw/hdw changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1. Position</td>
<td>1. Radar Input</td>
<td>Missile Data</td>
<td>1. No Data</td>
<td>1. No Nav. command updates</td>
<td>Catastrophic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Coordinates</td>
<td>Buffer</td>
<td>Processor</td>
<td>2. Data</td>
<td>2. Erratic cmds. generated &amp; error message</td>
<td>Critical</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2. Time</td>
<td>2. Radar Input</td>
<td>Missile Position</td>
<td>Inconsistent</td>
<td>operator error message</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Buffer</td>
<td>Buffer Handler</td>
<td>with Missile</td>
<td>Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3. Irregular data</td>
<td>3. Erratic Nav. cmds. and erroneous guidance</td>
<td>Critical</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>values (out of</td>
<td>reasonableness</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>range)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4. Input timing</td>
<td>4. Missile guidance precision</td>
<td>Marginal</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>incorrect</td>
<td>loss</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
If the hazard analysis being reviewed was accomplished prior to this certification review, it is important to perform an update to reflect any necessary changes in the software to resolve the previous hazard analysis problems.

The hazard analysis, together with the completed and updated SFMECAF, is evaluated to determine the Risk Index for the safety analysis. Table VI is used to determine the Risk Index.

<table>
<thead>
<tr>
<th>Category</th>
<th>Criticality States</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Catastrophic - may cause death or weapon system loss</td>
</tr>
<tr>
<td>II</td>
<td>Critical - may cause severe injury or major property damage, or major system</td>
</tr>
<tr>
<td>III</td>
<td>Marginal - may cause minor injury, minor property damage, or major system damage resulting in delay, loss of availability, or mission degradation</td>
</tr>
<tr>
<td>IV</td>
<td>Minor - causes injury or property damage resulting in unscheduled maintenance or repair</td>
</tr>
</tbody>
</table>

Table VI. Safety analysis risk index

<table>
<thead>
<tr>
<th>Risk Index</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Category 1 or 2 &gt; 0</td>
</tr>
<tr>
<td>Medium</td>
<td>Category 3 &gt; 0</td>
</tr>
<tr>
<td>Low</td>
<td>Category 4 &gt; 0</td>
</tr>
</tbody>
</table>

The Categories specified in Table V are defined in MIL-STD-1629A [11]. The rationale for the Risk Index and Criteria specified in Table VI is directly related to the definition of the Categories specified in Table V and the SED risk policy on Safety Critical Systems.

5. Operational Usability Analysis

The objective of this analysis is to provide for user involvement in accomplishing an assessment of the software regarding the usability and safe performance of the software for deployment. This analysis compliments the previous three analyses by focusing the assessment on the important operational utilization of the software from the user viewpoint. This analysis incorporates user input into the assessment validation to evaluate whether the system is capable of satisfying operational requirements while operating the software in a safe and effective manner.

There are three methods for performing this analysis:

1. Automated simulation of user/command inputs with the target system and software.
2. User operation of the target system and software in a test environment.
3. User operational scenario simulation.

Any one or a combination of the three may be used to complete this analysis. The common thrust of each method is that the user defines an operational scenario to be executed and the user evaluates the results of the software’s processing and outputs [13]. In the first two methods, the operational scenario is converted to software inputs and state data which is executed in a target system or target system breadboard. The user should be informed that the results of the operational scenario processing/simulation will be utilized as a basis for an airworthiness recommendation. Therefore, the scenario chosen should emphasize operational situations representing user preference with respect to safe and reliable software processing and outputs.

The first method is typically used for man-rated systems that use an automated interactive data generator and an environment simulator to provide feedback responses to the target system. This method is often used for embedded systems that operate in response to a prepared and remotely transmitted command sequence and/or a set of commands generated from sensors and associated electronics encountered in a weapon system or unmanned spacecraft. The user observes the system and/or reviews the results of the software processing/outputs for determination of suitability for mission operations (deployment). In each case where the results are unsatisfactory to the user, a defect is reported and assigned one of the error priorities listed in Table VII.

The second method is utilized on systems where the user plays a major role in the system/software operations by providing inputs to the system (e.g., avionics, weapon systems). The user inputs, as well as other data inputs, may be predetermined and provided to the system from a semi-automated, metered, or event-triggered input mechanism. Alternatively, the user may follow a script for user inputs, and other required data inputs are provided by an environment/predetermined mechanism. As in the first method the user observes the system and/or reviews results of the software processing/outputs for determination of suitability for mission operations (deployment). In each case where the results are unsatisfactory to the user, a defect is reported and assigned one of the error priorities listed in Table VII.
The third method is utilized when a target system is not available, or an automated input or environment mechanism does not exist, or a predetermined input is not practical, or the simulation environment necessary for the test is not practical. An example is to perform a scenario that requires the software to safely function when a near-miss nuclear explosion or a flight equipment anomaly occurs. This method requires that at least five individuals participate in a question and answer session. It is desirable to have more than one individual available for each of the following positions:

1. A user.
2. A software requirements expert.
3. A software design and code expert.
4. An independent test expert previously involved in CSCI and above testing.
5. A system/hardware expert.

As in the other two methods, an operational scenario meeting the specified requirements is used. The user interprets the operational scenario and postulates a specific operational situation. Each of the four remaining participants respond with their understanding of the software processing, the outputs produced, and the software/system state. Each individual justifies their response by citing documentation (requirements or design), test cases, results, or code references. The responders debate disagreements until consensus is achieved. The user evaluates the agreed upon answer to determine if the software processing and results, as specified by the group, are as expected, safe, and suitable for mission operations (deployment). In each case where the results are not satisfactory to the user, a defect is reported and assigned one of the error priorities listed in Table VII.

When one or more of the three methods has been completed and the defects documented, the defects and priorities are reviewed to determine the Risk Index for these analyses. Table VIII is used to determine the Risk Index.

### Table VIII. Operational usability analysis

<table>
<thead>
<tr>
<th>Risk Index</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Priority 1 or 2 &gt; 0</td>
</tr>
<tr>
<td>Medium</td>
<td>Priority 3 &gt; 0</td>
</tr>
<tr>
<td>Low</td>
<td>Priority 4 &gt; 0</td>
</tr>
</tbody>
</table>

The rationale for the Risk Index and Criteria specified in Table VIII is similar to that utilized in the Reliability Analysis for Table III. During the past 10 years, User inputs have assumed a significantly increased role in the evaluation of Army tactical systems. Users have become involved in system tests, field exercises of systems prior to deployment, and involvement in integrated product development teams. The criteria became less tolerant of errors as part of the changing emphasis during the past decade.

### 6. Airworthiness Recommendation

After the four analyses have been completed, the Risk Indices are reviewed to arrive at an airworthiness recommendation. In many cases, results from these analyses may lead to a qualified recommendation. For example, the analyses may indicate that further action to mitigate the risk is necessary in order to make a recommendation without qualifications. Table IX is suggested for determining an overall risk recommendation.
Table IX. Risk recommendations and criteria

<table>
<thead>
<tr>
<th>Risk Analyses</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>All risk analyses are Low</td>
<td>Recommend deployment and immediate effort be applied to all defects that resulted from these analyses.</td>
</tr>
<tr>
<td>One or more risk analysis is Medium</td>
<td>Recommend delaying deployment until all Medium risk issues are completely resolved or re-evaluated to be Low based upon software network and/or additional analyses. The certification procedure should be re-executed addressing those analyses and software areas that were involved in the Medium risk assessment.</td>
</tr>
<tr>
<td>One or more risk analysis is High</td>
<td>Recommend against deployment until all Medium and High risk issues are resolved. The certification procedure should be re-executed addressing those analyses and software areas that were involved in the Medium and High risk assessment.</td>
</tr>
</tbody>
</table>

7. Future Research

There are a few items that need additional research to make the critical software certification process applicable to a wider range of projects. The first item is to refine the criteria for determining the risk and suitability of the software for deployment when one of the four analyses cannot be performed. This could happen for any number of reasons with the most obvious being that one or more of the test activities was not performed, or that there was not a test coverage tool available when Code and Unit Test was performed, or the coverage data was incomplete or not available. As the approach now stands, there is no criteria provided when one of the four analyses is not possible.

The second issue is that other methods may be suitable or even better for any one of the four analyses. For example, would the defect data from formal inspections be better than or supplement the results of the CSCI level testing? If other alternate methods and associated Risk Index criteria were specified, then the likelihood of one of the analyses not being performed is lessened.

And it goes without saying that the Risk Index criteria in each case must be refined based upon experience using the existing procedure and criteria. The difficulty is that data collection and correlation with safety and airworthiness notions may require evaluation of several projects, together with deployment experience, for an extended period of time. However, one of the key benefits associated with this procedure is the use of standard criteria across numerous projects and the opportunity to evaluate the results on operational systems.

The Software Engineering Evaluation System team as well as other SED staff will continue to evaluate this procedure and research these areas for future research in an effort to continually improve the applicability and fidelity of the critical software certification procedure.

8. References