SAE The Engineering Society For Advancing Mobility Land Sea Air and Space®
INTERNATIONAL
400 Commonwealth Drive, Warrendale, PA 15096-0001

SURFACE VEHICLE/ AEROSPACE STANDARD

SAE JA1002

ISSUED JUL1998

Issued 1998-07

An American National Standard

Software Reliability Program Standard

Foreword—In 1994, the SAE G-11 Reliability, Maintainability, Supportability and Logistics (RMSL) Division chartered a software committee, G-11SW, to create several software standards and guidance documents across the RMSL spectrum, including a software reliability program standard. The committee was formed as a cross section of international representatives from commercial industries and governments.

The G-11SW committee has attempted to develop a standard that is consistent with a SAE G-11 system level reliability program standard and augmented by necessary software-specific support information. The G-11SW committee believes this document reflects the best current commercial practices, and meets the objectives of the United States Department of Defense Acquisition Reform initiative. This document is performance based and is intended to be used by industries to address market demands for reliable software products that improve system productivity, time to market, and cost-effective implementation. As appropriate, governments may also reference this document.

Software has been recognized by SAE G-11 as an important system component that is not adequately addressed at the system level. Software requires interpretation and variations on RMSL methods used by hardware. This document relies on the simple concept of supplier-customer dialogue and partnership to define, meet, and demonstrate assurance of software product reliability requirements. This document describes, within a Plan-Case framework, what performance requirements are necessary. An accompanying implementation guide (SAE JA1003) sets forth current best practices for how to structure the Plan in terms of activities, tasks, and methods so as to achieve the requirements of this document and provide demonstration evidence of reliability achievement in the form of a Case.

Development of this document has required dedication by a few participants and extended review by a wider audience of potential users. The professionalism of all these individuals and the support they received from their companies, governments, and other organizations is gratefully acknowledged.

Abstract—This SAE Standard defines a simple and flexible framework for the performance-based management of a software reliability program. The principal mechanisms are termed the “Software Reliability Plan” and the “Software Reliability Case.” The Plan and Case are general purpose management tools which are suitable for use in many fields of system engineering and will be observed throughout SAE RMSL program standards.

The Plan and Case in combination provide a means of tracking progress, performance achievement, and sustainment of a reliability goal. The Plan and Case support the philosophy of early fault removal and continued fault prevention throughout the software life-cycle. The Plan provides a forward view of intended reliability processes, activities, and performance requirements, while the Case provides evidence of software product reliability achievement as documented by quantitative and qualitative performance measures.

TO PLACE A DOCUMENT ORDER; (724) 776-4970 FAX: (724) 776-0790
SAE WEB ADDRESS http://www.sae.org

TABLE OF CONTENTS

1. Scope

1.1 Context—This SAE Standard provides a framework for the management of software reliability within system reliability requirements. It is based around the Software Reliability Plan and Software Reliability Case and emphasizes the importance of evaluating progress towards meeting software reliability requirements throughout the project life-cycle.

1.2 Range of Application—This document can be applied to all projects that incorporate software. This includes the integration of Off the Shelf (OTS) software products and custom software. OTS software sources include commercial vendors, government, and industry (e.g., reused library software). Custom software is generally newly developed software or a significant rework/upgrade of existing software that is for use with a specific application.

1.3 Roles

a. The Software Reliability Plan and the Software Reliability Case are intended to serve the needs of industry organizations in meeting software product reliability objectives.
b. The Software Reliability Plan and the Software Reliability Case can be employed as deliverables contacted between a customer and a supplier. Additionally, the Software Reliability Plan and the Software Reliability Case can contribute to any proposal for development or production and should be requested in a Request for Proposal (RFP).
c. A Software Reliability Case can be created or maintained to serve the needs of a support organization in sustaining reliability objectives.
d. A Software Reliability Case can be used to supply the data needed by independent, regulatory, and/or third party certification bodies.

1.4 Relationship to Management Plans—The Software Reliability Plan and the Software Reliability Case extend existing practices and may be integrated within other project management mechanisms provided that reliability progress is identifiable and traceable.

1.5 Technical Guidance—This document does not describe or recommend software development practices. However, the associated SAE Software Reliability Implementation Guide [JA1003] provides information about specific tasks which can contribute to the overall reliability goal and also includes further details of Plan and Case development. Other SAE program standards such as [JA1000] on system reliability may be consulted as appropriate. The references [R013], [ROMELAB], and [DEFSTAN0042] provide additional standards and guidance on software reliability implementation.

2. References

2.1 Applicable Publications—The following publications form a part of the specification to the extent specified herein. Unless otherwise indicated the latest revision of SAE publications shall apply.

2.1.1 SAE PUBLICATIONS—Available from SAE, 400 Commonwealth Drive, Warrendale, PA 15096-0001.

SAE JA1000 (Draft)—Reliability Program Standard
SAE JA1001 (Draft)—Software Reliability—An Overview
SAE JA1003 (Draft)—Software Reliability Implementation Guide

2.1.2 ANSI PUBLICATION—Available from ANSI, 11 West 42nd Street, New York, NY 10036-8002.

ANSI/AIAA R-013-1992—AIAA Recommended Practice for Software Reliability, February 1993.

2.1.3 IEEE PUBLICATION—Available from IEEE, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331

MUSA92—"Operational Profiles in Software Reliability Engineering", Musa, John D., IEEE Software, March 1992

2.2 Other Publications

ROMELAB—Lakey, Peter B., and Neufelder, Ann Marie, "System and Software Reliability Assurance Notebook," produced for Rome Laboratory, September, 1996

DEFSTAN0042—Defence Standard 00-42 (PART 2)/Issue 1, "Reliability And Maintainability Assurance Guides, Part 2: Software," United Kingdom Ministry of Defence, 1997

3. Definitions

3.1 Software Reliability—(1) The probability of failure-free operation of a software program for a specified time under specified conditions; (2) a set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

3.2 Software Failure—The inability of a software component to perform its required functions within specified performance requirements.

3.3 Software Fault—An accidental condition that causes a software functional unit to fail to perform its required function.

4. Software Reliability Program

4.1 Objectives—The objectives of a software reliability program are to:

a. Ensure the delivery of a software product that has been adequately designed to achieve its performance specifications; and,
b. Ensure there is adequate evidence that the performance specification for the delivered software product has been achieved and continues to be met during operational use.

4.2 Principles—The three principles of a software reliability program are that customer requirements for software reliability shall be:

a. determined;
b. met; and
c. demonstrated.

4.3 Management Framework—The framework for the management of software reliability is built around two key components: the Software Reliability Plan and the Software Reliability Case.

The Software Reliability Plan addresses the software aspects of the overall System Reliability Plan and issues that arise from the use of software. It describes the activities that are to be undertaken to achieve software reliability requirements and also describes the activities that are to be undertaken to demonstrate that software reliability requirements have been achieved.

The Software Reliability Case provides a justification of the approach and, during the project, documents the evidence which verifies that the software meets the reliability requirements.

The Software Reliability Plan and the Software Reliability Case may be employed as deliverables contracted between a supplier and purchaser. The Plan provides a forward view of intended processes while the Case justifies the Plan and also looks back at achievement. Any proposal should include not only a Plan but also a Case in order to justify design and process decisions upon which the proposal is based. The Case continues to be developed throughout a project and provides visibility of progress. Iterations of the Case may be linked to project milestones and may continue to the in-service phase.

5. Determining Customer Requirements

5.1 General—At the outset of a project, it is typical that no requirements for software reliability are offered by the customer. It is common for reliability requirements to be specified at the system level. It is the responsibility of the supplier to determine how the system requirement should be allocated among system components, including software. The supplier is in the best position to determine how system level requirements can be achieved most cost-effectively.

5.2 Factors to be Considered

5.2.1 DEFINITIONS OF FAILURE—It is important to establish reliability aspects of the behavior of the proposed system, including definitions of failure. A starting point in determining reliability requirements for software is to examine system reliability requirements together with required system functionality in order to distinguish critical functions and their definitions of failure, and also any degraded modes of operation. In many applications, it will be necessary to identify multiple reliability requirements, depending on the criticality of the function served and on the expected time to recover from failure.

5.2.2 OPERATIONAL PROFILES—Operational profiles [MUSA92] determine the relative demand of functions within the whole range of demands placed upon the system. Consideration of operational profiles will assist in the understanding of software reliability requirements, will provide benefits in prioritizing development activities, and will support demonstration that the requirements have been met.

5.3 Roles of the Plan and Case

5.3.1 SOFTWARE RELIABILITY PLAN—The Software Reliability Plan should describe the activities which will be performed throughout the project to ensure that:

a. Requirements for software reliability are derived from and remain consistent with system level requirements;
b. Requirements for software reliability are understood and implemented by the development organization; and
c. The Software Reliability Case remains accurate and current, and is presented effectively.

5.3.2 SOFTWARE RELIABILITY CASE—The Software Reliability Case should present evidence throughout the project that software reliability requirements are consistent with system level requirements, are achievable, are understood by the development organization, and that ambiguities have been resolved. The Software Reliability Case structure and organization will require adequate planning.

6. Meeting Customer Requirements

6.1 General—Achieving software reliability depends on the prevention, removal, and mitigation of faults in the final software solution. For software of even modest complexity, the ideal of zero defects has proven to be elusive. The potential for faults to be injected begins at project inception and continues throughout development and support. Faults entering early in the development cycle, and not removed at the same stage they were injected, will be much more difficult and costly to remove as work continues. Faults encountered during operation may require a certain level of tolerance and subsequent recovery capability to reduce the impact of system failure.

6.2 Strategy for Software Reliability Achievement

6.2.1 SYSTEM REQUIREMENTS—The quality of system requirements will influence the achievement of software reliability objectives. Therefore, the supplier should carefully derive, evaluate, and validate the software requirements before beginning software or system design to make sure that the customer's needs are fully understood. This will involve a dialogue between the supplier and customer.

An important aspect of requirements are abnormal conditions. The supplier should review requirements for handling abnormal conditions and should add requirements for handling abnormal conditions that may arise from software design. The customer should review and approve any such requirements and the actions which result from abnormal conditions.

6.2.2 SOFTWARE ENGINEERING PROCESS—A mature software engineering process is necessary but not sufficient for the achievement of software reliability. An established process is necessary to ensure that appropriate methods and techniques are carried out at the correct point in the development, to enforce configuration control, and to enable adequate management of the project. However, development processes do not generally allow predictions or demonstrations of software reliability to be made. To provide assurance that software products meet their reliability requirements, a software reliability program should be planned as a specific activity within the software development process.

6.2.3 ALLOCATION OF RELIABILITY REQUIREMENTS TO SOFTWARE—Software components may be identified at various levels in a system breakdown, and may be implemented in a number of ways, including the operating system, application software, firmware, and OTS. The supplier should allocate system reliability requirements to the major software components as an initial step in software reliability planning. This enables activities to be planned on the basis of their ability to achieve and demonstrate the specific software reliability requirements. An understanding of the operational profile (see 5.2.2) will contribute to this activity.

6.3 Role of the Software Reliability Plan—The Software Reliability Plan documents the management and technical activities that bear on the achievement of software reliability, including the maintenance and updating of the Software Reliability Case. It should be traceable to system reliability planning and to avoid unnecessary replication, should be integrated with software development and quality management planning. The Software Reliability Plan should describe how the software engineering process specifically addresses software reliability issues.

Software reliability planning should address the following activities:

a. Allocating reliability requirements to software;
b. The strategy for software reliability achievement;
c. The techniques, methods and tools, including measurements to be used for the evaluation of the achieved software reliability at each life-cycle phase;
d. Risk analysis for the software reliability objectives;
e. Identification of data base tools that support data collection, analysis, and storage;
f. The identification, selection, and integration of OTS software;
g. The means by which all staff, including subcontractors, are made aware of their specific responsibilities in meeting the software reliability requirements;
h. Specific training activities related to reliability models, methods, and techniques;
i. The procedures for software reliability progress reporting, including the phased update of the Software Reliability Case and how information in the Case is validated;
j. The distribution of resources employed to address software reliability issues, including the involvement of the customer and any third party; and
k. The timing of the elements of the Software Reliability Plan relative to the system reliability plan.

7. *Demonstrating Requirements Satisfaction*

7.1 General—Demonstration of software reliability achievement is obtained from two types of evidence:

a. Direct evaluation of the achieved reliability of the software products; and
b. Indirect assessment during development by measuring process and product attributes which are known to have a correlation with software reliability.

Evidence should be compiled to form a Software Reliability Case. This is an assembled rationale that would be convincing to a third party and consists of:

a. Claims, about properties of the software;
b. Evidence and assumptions, which are used to support reliability claims; and
c. Rationale linking evidence to claims.

Both the achievement and demonstration of software reliability should be design drivers. Therefore software and the software development process should be designed to facilitate production of the Software Reliability Case. This approach should help minimize delays and costs for the supplier due to software reliability problems and will assist future maintenance of the Case.

7.2 Role of the Software Reliability Case—The Software Reliability Case presents arguments and evidence that the requirements can be achieved, will be achieved, and have been achieved. For maximum effect, the Case should be developed and witnessed as development decisions are made. It is not intended to be a retrospective justification of the solution.

The Software Reliability Case should be a readable overview of the evidence that the software meets its reliability requirements, with references to project development records and the results of analyses of software components as appropriate. The Case provides significantly more than proof that the Plan has been executed as it provides evidence about the products of the development process. This evidence should address the direct evaluation of the reliability of the software elements (e.g., from analysis of the design and reliability tests and trials), and also the suitability of the software architecture and the software engineering process.

The Case should be a living document and its development should proceed through a number of stages of increasing detail during the project. At the beginning of a project, before committing significant resources, the Case should provide evidence there is an acceptable level of confidence that the reliability requirements will be met. During development, the Case should provide evidence that the reliability requirements are being met by the software products. During operational use, the Case should provide evidence that the software is reliable and continues to be so in the event of any maintenance.

7.3 Structure of a Software Reliability Case—In general, diverse types of evidence may be used within complementary approaches to demonstrate the reliability of software. Reliability evaluation of software with stringent reliability requirements is especially difficult. For most situations where stringent reliability requirements are to be met, the Supplier should plan to provide diverse forms of evidence to cover each aspect of software reliability.

A Software Reliability Case is a structured summary and overview of the evidence as it is collected during the project, with reference to detailed results as appropriate. This may include direct evidence in the form of:

a. Evidence of defects collected during analysis, design, and coding, typically generated from inspections or other peer reviews, and subsequently, evidence of corrections;
b. Evidence from testing, including reliability growth modeling, statistical testing, data from tests and trials, and performance testing, with an assessment of its accuracy and relevance to operational use;
c. Field data, including a description of any claims made on the basis of previous in-service experience, including a comparative analysis of relevant criteria;

d. Results of any analytic arguments deployed to show the absence of certain faults, and the assumptions on which they are based. Analytic arguments can be applied to abstract concepts which precede practical testing; and
e. Analyses of any third party software that is to be used, and a description of the way in which it will contribute to software reliability, taking into account any changes in operating environment and use profile.

The Software Reliability Case should also present indirect evidence in the form of an overview of the software engineering process to explain how it is appropriate for the reliability requirements for the software. Items that might be included are:

a. An analysis of the methods, techniques, and procedures to be used, with an assessment of their suitability for the reliability requirements, including:

1. A description of the generic types of software faults that will be specifically addressed and minimized;
2. An analysis of reliability data on other software developed using the proposed or similar methods, techniques, and processes (sources of these data include the supplier's own records and published literature); and
3. A justification of the choice of tools and support software, with a description of the way in which known problems (e.g., compiler faults) are to be handled and recorded.

b. A description of how reliability progress is to be measured. This should include:

1. Any measurements that are to be taken (e.g., as part of inspections) during the software development life-cycle of the performance of the software engineering process;
2. The verification and validation test strategy, including the test coverage to be achieved and the means for generating the tests; and
3. Acceptance criteria for these measures, based on previous experience with the software engineering process. During development, these data should be recorded in the Case and compared with the acceptance criteria. The corrective action to be taken, if the acceptance criteria are not met, should be described.

c. Confirmation that the minimum levels of personnel competency defined in management plans have been achieved.

7.4 Phasing of the Software Reliability Case—The development of the Software Reliability Case takes place in identifiable phases:

a. Predevelopment-Stage Software Reliability Case, which can form part of any proposal in order to justify design and process decisions upon which the proposal is based;
b. Development-Stage Software Reliability Case, which can be a contracted deliverable, phased as appropriate; and
c. In-Service Software Reliability Case, which can be a contracted deliverable, considered at an early stage or negotiated at a later date.

7.4.1 PREDEVELOPMENT-STAGE SOFTWARE RELIABILITY CASE—The Predevelopment-stage Software Reliability Case should provide an overview and analysis of the supplier's approach to reliability achievement. It should provide evidence that:

a. The individuals and organizations involved, including subcontractors are capable of supplying software that is commensurate with the reliability requirements of the proposed system;
b. The Software Reliability Plan is appropriate for the reliability requirements of the proposed system;
c. Technical proposals and decisions upon which the tender is based are appropriate for the reliability requirements of the proposed system; and

d. The Supplier will be able to demonstrate reliability achievement during the project.

7.4.2 DEVELOPMENT-STAGE SOFTWARE RELIABILITY CASE—The Development-stage Software Reliability Case should be derived from the Predevelopment-stage Software Reliability Case taking into account the effects of negotiations and technical changes.

During development, the Supplier should update the Software Reliability Case with a summary and appraisal of the results of the activities that contribute to the reliability evaluation. By the time of acceptance into service, the Case should contain the complete set of evidence that the reliability requirements of the software will be met.

The Software Reliability Case should describe measurements taken of the software products and the software engineering process which provide evidence that the development is proceeding satisfactorily; earlier versions of the Case should predict values for these measurements that the customer can compare to the results actually obtained later in the life-cycle.

Versions of the Case should be planned for appropriate milestones in the software development life-cycle, for instance, after reliability tests and trials. The current version of the Case should be presented at design reviews and the outcome included in the Case.

Versions of the Case should be managed under a configuration control mechanism to ensure compatibility with development status.

In the event that development continues after a system is taken into service, then a Development Stage Case should be continued to an agreed point of satisfaction.

7.4.3 IN-SERVICE SOFTWARE RELIABILITY CASE—In-service software reliability management includes the collection of operational and usage data and the maintenance of an In-service Software Reliability Case. The Case should contain a description of the field experience with the software or any part of it, and an analysis of the impact of any software failures on the reliability of the system. The analysis might address the potential consequences of the failure, its root causes, mitigations (perhaps through fault tolerance and recovery mechanisms), and the lessons learned for the software engineering process.

In-service reliability data can be used for:

a. Assessing or confirming reliability achievement;
b. Determining reliability impact when software is used in a new environment;
c. Gathering experience on the performance of particular methods, techniques, and processes for the benefit of future projects and development organizations which operate within a process improvement framework; and
d. Assessing effectiveness of change implementation during software support.

Options for employing an In-service Software Reliability Case, including responsibilities and procedures for managing reliability data should be considered during negotiations for support.

8. Contractual Application of the Program Standard—This section summarizes the possible customer application of this Software Reliability Program Standard to supplier contracts. This section explains what could be required in contractual documentation and what might be expected of customers and suppliers. A generalized view is presented which can be developed to deal with different phases of the procurement cycle.

General application of this Software Reliability Program Standard is illustrated in Figure 1. This sequence chart is read from top to bottom and shows the ordered flow of data between purchaser and suppliers.