Scorecard Reviews

for Improved Software Reliability

Timothy Pohland ■ David Bernreuther

Software reliability poses a significant challenge for the Army as software is increasingly important in a broad range of applications. The safety, welfare and effectiveness of our Soldiers directly depend on the ability of software to perform as intended and operate reliably in adverse and austere conditions. The ability of the Defense Department (DoD) to provide a high level of Soldier services while minimizing overhead and other sustainment costs is tied directly to the reliability of large and complex software systems.

Software is a key enabler in compiling intelligence, conducting analysis, and performing command-and-control functions. For example, the Army must support numerous command, control, communications, computers, intelligence, surveillance and reconnaissance (C4ISR) systems at a cost in excess of $200 million annually. Furthermore, embedded software has become an essential feature of virtually all hardware systems. This necessitates assessing system reliability through a holistic accounting of hardware, software, operator and their interdependencies.

Pohland *is a computer engineer and a member of the U.S. Army Materiel Systems Analysis Agency (AMSAA) Reliability Branch with 5 years of reliability analysis, software development, and process improvement experience.* **Bernreuther** *is an operations research analyst at AMSAA Reliability Branch at Aberdeen Proving Ground, Md., and has 30 years of software development, technical analysis and process improvement experience.*

Figure 1. Sample Element Description (Low Risk)

Design for Reliability
Developmental Testing
• All modules of software are covered by unit testing and all are included in integration testing. • All external systems are included (or surrogated) in integration testing. • Multiple sets of test data are available to support both unit testing and integration testing during development. The test data adequately represents the scale of operations that the software will encounter when used operationally.

Issues with the performance and reliability of software throughout the DoD led to development of the Capability Maturity Model (now Capability Maturity Model Integration or CMMI) in the late 1980s. Similarly, industry standards such as the IEEE/ISO 12207 also have emerged to address sound software practices. Additionally, a plethora of tools and techniques, from requirements management tools to dynamic code analyzers, are available to support the engineering of reliable software. Yet, as mature as software reliability practices and standards have become, reliability remains a significant issue for government and industry. A recent sample of combined hardware-software systems tested by the Army illustrates the significance of the issue. It found software failures pervasive throughout the sampled systems, constituting 52 percent of the overall failures (worst case: 82 percent).

Benefits of a Scorecard Approach

Software development often is complex and expensive, while available resources and time are generally limited. This exacerbates the challenges of ensuring appropriate and effective practices are followed. To facilitate development of more reliable software, the Army Materiel Systems Analysis Agency (AMSAA) has developed a software reliability scorecard. This instrument extends and complements an existing scorecard on general reliability practices. It also complements organization-centric approaches, such as CMMI, by assessing the level of risk associated with reliability-specific practices within an individual software project.

The scorecard is a structured and transparent instrument for assessing the health of an individual software development effort and is invaluable in isolating weak areas for further analysis and work. It enables scarce resources to be prioritized and, subsequently, more reliable software to be developed. The research, discussion and reflection undertaken while applying the instrument can be as valuable as the resultant score(s). It provides a common structure for multiple disciplines to see the interrelationship and importance of various reliability issues and practices outside their own domain.

The scorecard focuses on areas that warrant additional research and analysis. It highlights areas of weakness and, through the evaluator's recommendations, gives insight on how to address those weaknesses. However, it is not prescriptive as to specific actions that should be taken. AMSAA recommends that the areas of concern, once discovered, be investigated further to identify the best software and reliability practices and tools to be applied at those areas.

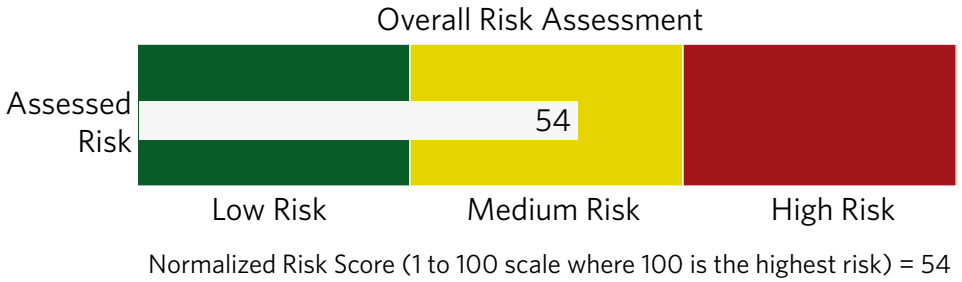
How It Works

The software reliability scorecard adheres to conventions similar to the existing General Reliability Scorecard, which is used by numerous DoD organizations. It is self-contained in an Excel spreadsheet with 57 specific elements to be evaluated and rated. All the elements are grouped into seven categories, along with definitions for each rating level, and laid out in a single input sheet. Each element is rated red, yellow or green to represent high, medium or low risk, respectively. An example of a rating definition of Low Risk for Developmental Testing appears in Figure 1. Additionally, cells are provided for the rater to enter a rationale and recommendations for each element. These offer important insights for the feedback process, helping turn the scorecard ratings into focused and effective actions.

The scorecard processes the individual ratings and derives a total score assessing how much the program is at risk. The ratings are adjusted by weighting factors assigned to each individual element. The overall risk assessment is normalized to a value between 1 and 100, where 1 is low risk and 100 is high risk. A top-level quantitative assessment is illustrated in Figure 2.

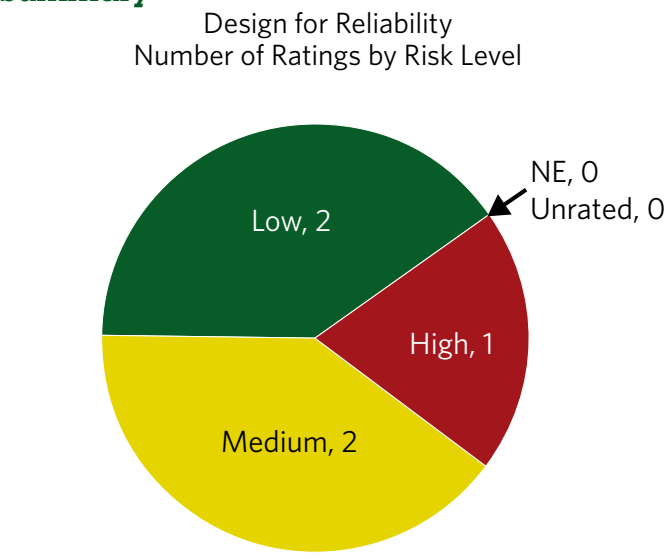
The scorecard also generates summaries of each of the seven categories to help focus on strengths and weaknesses. These pictorially represent the number of high-, medium- and low-risk elements in each category, as shown in Figure 3.

Figure 2. Sample Overall Rating



Normalized Risk Score (1 to 100 scale where 100 is the highest risk) = 54

Figure 3. Sample Category-Level Summary



Normalized Risk Score (1 to 100 scale where 100 is the highest risk) = 40

It is highly recommended that a multidisciplinary team apply the scorecard to a project. Initially, each member should assess the project from his or her perspective and knowledge. The team then should compare and discuss its members' ratings of each element to achieve consensus. These deliberations usually are as insightful and valuable as the rating scores. The team should use the Rationale and Recommendation fields to help record the key points of its deliberations and draft recommended actions.

The instrument is designed to be of value at any stage of software development. Although many of the elements lend themselves to specific stages of the software life cycle, the results are most insightful when all elements have been assessed regardless of the project's current status. If the instrument is being applied later in a project's life cycle, assessment of the earlier activities provides valuable insight about the character of the existing design and code. Conversely, activities that are usually a priority later in a project (preparations for Fielding and Sustainment, for example) are "leading indicators" of the quality and thoroughness of the reliability tests being conducted. These later elements are not weighted as highly as earlier activities but still reduce risk. A project that proactively starts them early is reducing the overall risk of the project.

There may be cases where an element simply does not apply to a given project. For these circumstances, there is a "not evaluated" option for reviewers; the scorecard will drop these elements from its calculations and provide a normalized risk score from the remaining elements. In the summary chart for each category, a count of the "NE" elements is provided.

Categories and Elements

The software reliability scorecard addresses seven key categories of reliability practices most applicable to software development: program management, requirements management, design capabilities, system design, design for reliability, (customer) test and acceptance, and fielding and sustainment. Each category then covers a number of specific elements focusing on key reliability practices. A detailed examination of every element within the scorecard is beyond the scope of this article. However, a discussion of each of the categories and some of their key features should provide an understanding of the breadth and depth of the examination of good reliability practices.

Program Management

Development of reliable software requires that limited resources and time be well managed; that the customer and team work collaboratively; and that reliability-enhancing activities receive the necessary resources, visibility and priority. The Program Management category addresses these needs by looking at 12 elements:

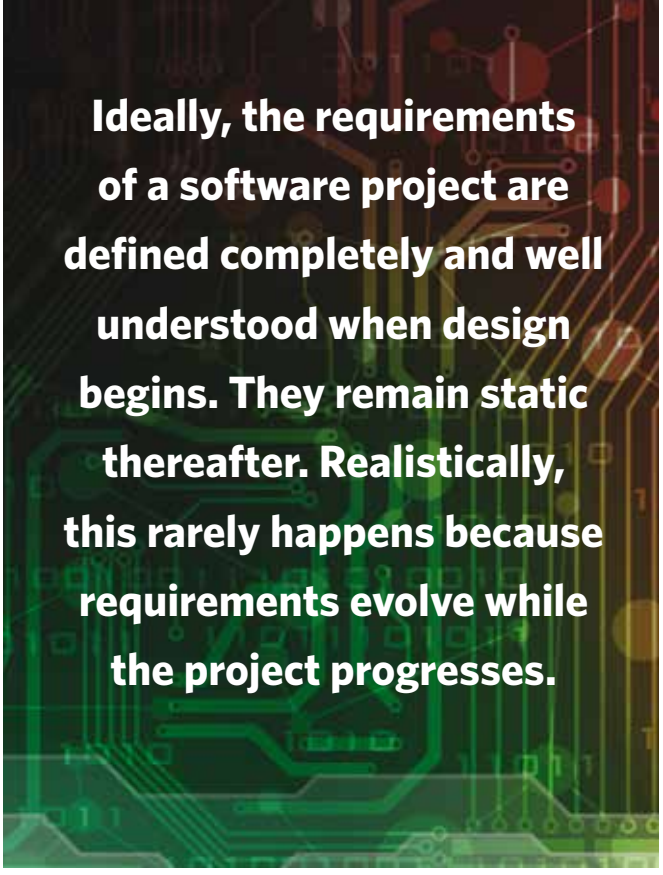
- Developer's Experience
- Process Maturity
- Program Planning
- Currency of Plans
- Progress Reviews
- Readiness
- Reliability Engineering
- Reliability Growth Management
- Verification and Validation (V&V)
- Supporting Disciplines
- Risk Management
- Commercial Off-the-Shelf (COTS) Management

CMMI, or similar organization maturity levels, is considered part of the developer's experience. As organizational maturity does not guarantee success, this element also considers the developer's experience with applications of the domain, size and complexity under consideration. Supporting Disciplines include human factors, technical writing, and others—essential to the software, but not necessarily involved with a project's core design and coding activities.

Requirements Management

Any successful development project relies on complete and clearly understood requirements. These should include well-considered and -defined reliability goal(s). A good system for recording requirements and linking them to the design capabilities that will provide them is also essential. The elements in Requirements Management are:

- System Requirements
- Currency
- Reliability Goals
- Requirements Allocation
- Quality of Requirements



- Use Cases
- Interoperability
- Dependency
- Other Characteristics

Ideally, the requirements of a software project are defined completely and well understood when design begins. They remain static thereafter. Realistically, this rarely happens because requirements evolve while the project progresses. Accordingly, Requirements Management also assesses how well the project realizes requirements have changed and adapts its plans and activities. A mature developer will seek to discover and accommodate the changes. This not only reduces risk but minimizes costs and time, as resources are more wisely allocated to accommodate the newest requirements.

Use Cases is a notable element, because these cases provide a powerful method to enhance understanding between the customer and developer and among the various involved disciplines. They also facilitate requirements definition, development, testing and documentation such as users guides. Their use greatly aids the development of reliable and effective software.

Design Capabilities

By assessing the developer's Design Capabilities, the scorecard seeks to ensure that sufficient capabilities to design reliable software are in place and implemented effectively. The seven elements of Design Capabilities are:

- Development Process
- Process Implementation
- Documentation and Repository
- Configuration Management
- Collaboration Capabilities
- Development Samples
- Analysis of Alternatives

The Development Samples element is particularly relevant to the DoD, where the applications/systems to be supported are often unique or highly specialized.

System Design

The software scorecard is intended to examine the enabling practices and capabilities being applied to a software development effort. It is not intended to make a detailed software engineering assessment of the code and design itself. However, the following key elements of the design can be assessed to ensure that the design addresses reliability drivers:

- System Architecture
- Modular Design
- Data Architecture
- Interface Design
- Fault Tolerance
- Usability
- COTS Selection

The scorecard treats Usability as an element integral to the software's reliability. This is particularly true for DoD applications, where a misunderstanding by the user or a series of small disruptions can endanger Soldiers in combat. Similarly, Soldiers rely on software prepared to handle Faults (internal and with other interfaced systems) with minimal disruption. The physically dispersed and austere locations in which the Army operates require that a well-designed Data Architecture consider the physical location of data and users and the realistic levels of communication between them.

Design for Reliability (DFR)

The category of DFR elements endeavors to ensure that practices crucial to the design of any reliable product are integral to the development effort. These practices need to be applied from the very start of development; program risk should be assessed higher if they are not. The DFR elements consist of:

- Failure Management
- Developmental Testing
- Reliability Monitoring
- System Reliability Analysis
- Independent Reviews

It is significant to note that Developmental Testing is called out as a separate element. Solid DFR requires that most reliability be "baked in" the software and design before customer testing. This means that the developer must expose failure modes as early as possible. Each mode cannot be resolved immediately. However, a general characterization of the failure

mode enables the developer to plan when and how it should be addressed. Mature developers will reduce risk and costs by conducting developmental testing to expose, characterize and prioritize failure modes as soon as possible. The subsequent customer test then becomes, as appropriate, a refinement stage to iron out a minimum of issues.

(Customer) Testing and Acceptance

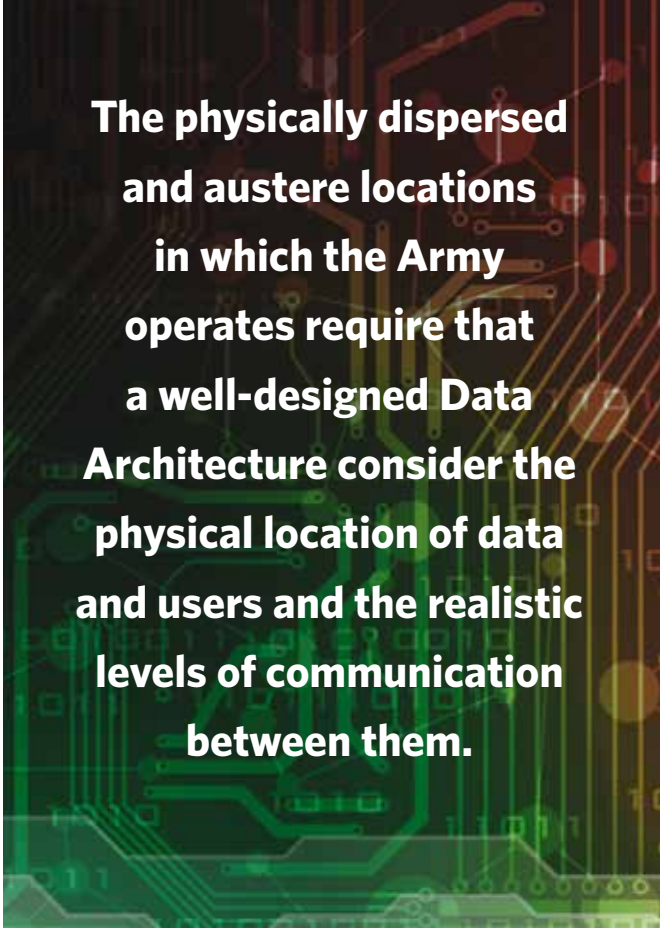
It is preferable that a developer bring a highly reliable product to customer testing—one that requires minimal corrections. The reality is that many software systems are provided to the customer with a low level of reliability. This has been particularly true of many DoD applications that call for a level of sophistication, fault tolerance and size that are hard to simulate in a development environment. Generally, new DoD systems experience new failure modes and a subsequent decline in reliability when they are fielded. Correcting these failures is extremely expensive and difficult. Accordingly, it is vital that customer acceptance-testing expose and resolve as many remaining failure modes and problems as possible. Further, the developer must have the capabilities and tools in place to handle the new issues as they arise. The numerous elements of (Customer) Testing and Acceptance seek to address these issues. The elements are:

- Test Coverage
- Companion Test Systems
- Test Depth
- Usability and Suitability
- Scalability
- Non-functional Characteristics
- Embedded Systems Testing
- Failure Analysis
- Software Code Analysis Tools

Fielding and Sustainment

The utility of a software reliability scorecard is less apparent when a project has reached the Fielding or Sustainment stage. However, the fielding and maintenance of existing software requires significant resources. For large applications, the financial cost is substantial. The elements of this category assess how well a project has prepared for these stages of the life cycle. It is recommended that the prior elements of the scorecard also be assessed at this time. They provide insight into the quality and character of the software on hand. A mature developer will, for example, know what issues (e.g., failure modes) remain unresolved. The elements addressed regarding Fielding and Sustainment are:

- Software Maintenance
- Field Support
- Documentation for Sustainment
- Dependencies and Interoperability
- Sustainment Testing
- Training
- Distribution
- Continuity of Operations Plan (COOP)



In the earlier stages of software development, scorecard users are encouraged to consider and assess the Fielding and Sustainment elements. Though these activities may not be due yet, a proactive developer will start working on them in parallel with earlier activities. They can serve as "leading indicators" of a low-risk project.

Summary and Conclusion

AMSAA developed the software scorecard to complement its existing hardware reliability portfolio and facilitate an increasing software reliability workload. Its primary use is to assess the reliability practices of individual software projects and to focus more detailed analysis and work on the areas of most concern. It is a particularly useful approach when "triaging" ongoing software projects to identify where to focus analysis and support. AMSAA applies the scorecard methodology to provide Army and other DoD programs an independent assessment of the project's level of risk. However, the instrument can be applied by the customer or as a self-assessment tool by the developer. Valuable ideas and feedback from industry partners and other DoD organizations have been used to improve this tool for more general use.

The software scorecard is available at no cost to any U.S. government agency and its contractors. Information to request a copy is available at www.amsaa.army.mil/ReliabilityTechnology/RelTools.html.

The authors can be contacted at **timothy.g.pohland.civ@mail.mil** *and* **david.g.bernreuther.civ@mail.mil.**