Issues in the Next Generation of Dependability Standards

Abstract

Existing IEEE software reliability standards do not address the characteristics of distributed systems, including client-server systems. Furthermore, these standards were issued before the widespread application of COTS and safety-critical systems. In addition, these standards do not take into account the influence on reliability of such process improvement measures as inspections, reuse, and object-oriented design paradigms. Lastly, these standards do not consider both hardware and software reliability nor do they include availability and maintainability. To be of value, the next generation of dependability standards must address these deficiencies. With the active participation of the audience, the panel will identify and debate the future direction of dependability standards. Panel participants are the following:

Norman Schneidewind (Chair), Naval Postgraduate School, USA
Jean-Claude Laprie, LAAS-CNRS, France
Allen Nikora, Jet Propulsion Lab, USA
Michael Lyu, Chinese University of Hong Kong, China
John Musa, Software Reliability Engineering and Testing Courses, USA
Bill Everett, SPRE, Inc - Software Process & Reliability Engineering, USA

Overview

Norman Schneidewind (Chair), Naval Postgraduate School, USA

In general, existing software reliability standards do not address the characteristics of distributed systems, including client-server systems. This is an amazing situation given the importance of distributed systems in contemporary society. The reason for this situation is the rapid technological development of distributed systems and networks, most notably the explosive growth of the Internet. This rapid growth has made it difficult for the software reliability and standards development communities to respond to the challenge of the new technology. However, to address this deficiency, the software engineering community of the Institute of Electrical and Electronic Engineers will develop a new standard, which will include software reliability, maintainability, and availability for distributed systems.

The extant software reliability standards and practices, using IEEE 982 [1] and AIAA R013 [2] as examples, were issued before the widespread application of distributed systems, client-server systems, COTS, and safety-critical systems. In addition, for the most part, these standards do not take into account the influence on reliability of such process improvement measures as inspections, reuse, and object-oriented design paradigms. Furthermore, in general, existing reliability standards do not consider both hardware and software reliability nor do they include availability and maintainability. Note: whether this standard should include both hardware and software -- a systems standard - is an issue that the panel and audience should discuss. Lastly, existing non-reliability standards do not provide for substantive application of even existing software reliability prediction and assessment technologies. Therefore, one of the tasks of the working group will be to recommend appropriate RMA technologies to include in new and revised software engineering standards.

Too often, the models assume that the software involves either a single module or node [3]. The reality in today's increasing use of distributed systems is that there are multiple entities of software that execute on multiple nodes. A software failure in a critical node would not necessarily cause the system to fail. Therefore, we don't model such a system as a single entity because the predicted reliability will be much lower than the true reliability. This is the case because the prediction will not account for criticality and redundancy. The first factor accounts for the possibility that the survivability of some clients and servers will be more critical to continued system operation than others. The second factor accounts for the possibility that using redundant nodes allow for system recovery should critical nodes fail. Thus, we identify the critical and non-critical client and server nodes in order to achieve realistic predictions and assessments of reliability. The new standard will recognize this environment.

In order to address these shortcomings, it is proposed that a new life-cycle reliability, maintainability, and availability (RMA) process standard be developed and that a Project Authorization Request be written and a working group formed for this purpose. A process standard is recommended rather than a product standard for two important reasons:

1) Developing a standard that would specify specific numerical RMA quantities to be achieved and subsequently verified for a distributed safety-critical system product, for example, is infeasible with current technology. However, it is important that the standards user realize that in implementing an RMA process for such a system it is necessary to consider criticality and redundancy in the system. For example, there have been cases where every failure in a distributed system was considered the equivalent of a system failure. In these cases the evaluators did not take into account the existence of multiple nodes in the system with different degrees of criticality and did not consider that less critical nodes could substitute for failed critical nodes. Such an approach would severely underestimate system reliability and availability.

2) The proposed approach would minimize the emphasis of whose model is best or best for a particular application (i.e., "model wars") as was the case in part when R-013 was developed. Instead, the emphases would be on an appropriate RMA process, independent of the model that is used.

Dependability

Jean-Claude Laprie
LAAS-CNRS - 7 avenue Colonel Roche, 31077
Toulouse, France

Computer systems are characterized by four fundamental properties: functionality, performance, cost, and dependability. Dependability is the property of a computer system such that reliance can justifiably be placed on the service it delivers [4]. A structured view of dependability takes the form of a tree:

- The threats to dependability are faults, errors, failures; these are undesired - but not unexpected - events or states.

- The means for dependability are fault prevention, fault tolerance, fault removal, fault forecasting; these are the methods and techniques enabling one a) to provide the ability to deliver a service on which reliance can be placed, and b) to reach confidence in this ability.

- The attributes of dependability are availability, reliability, safety, confidentiality, integrity, maintainability; these a) enable the properties that are expected from the system to be expressed quantitatively, and b) allow assessment of the system with respect to the threats and the means to oppose them. Depending on the application(s) intended for the system, different emphasis may be put on different attributes.

A major strength of the dependability concept, is its integrative nature, which enables to put into perspective the more classical notions of reliability, availability, safety, security (the combination of confidentiality, integrity and availability), maintainability, which are then seen as attributes of dependability. The fault-error-failure model is central to the understanding and mastering of the various threats which may affect a system, and it enables a unified presentation of these impairments, though preserving their specificities via the various fault classes which can be defined. The model provided for the means for dependability is extremely useful, as those means are much more orthogonal to each other than the usual classification according to the attributes of dependability, with respect to which the design of any real system has to perform trade-offs, due to the fact that these attributes tend to be in conflict with each other.

Why Do We Need a Software Dependability Standard?

Allen P. Nikora, Jet Propulsion Laboratory, USA

Over the past several years, several standards related to software dependability have been published, of which the best-known are IEEE Std 982 [1] and AIAA/ANSI R-013-1992, the AIAA Recommended Practice for Software Reliability [2]. With these standards already available, we ask:

o Do we need yet another standard?

o If so, what kind of standard do we need?

To answer the first question, we can summarize the available standards. Standards like IEEE Std 982 provide definitions of measures related to dependability, but do not describe the broader context in which these measures are used. The AIAA Recommended Practice describes methods for estimating software reliability, selecting appropriate models, and collecting data, but as with IEEE Std 982, does not place these methods into the context of the overall software development process. There does not seem to be one single standard to which practitioners can turn to include dependability into the software development process. In addition, software technologies that are now widely used were still being developed at the time these standards were created. A software dependability standard should be produced to address these difficulties.

One possible model for a standard is ISO9000 and its companion standards and guidelines. Following this model, the standard would describe the processes involved in:

o Specifying, designing, and implementing a dependable software system.

o Measuring the system's dependability.

Excluded from the standard, but included in companion guidebooks, would be detailed descriptions of:

o Specific life cycle models for developing dependable software.

o Specific techniques for estimating/forecasting a system's dependability.

o Criteria for selecting dependability estimation methods appropriate to a particular development effort.

By standardizing only the way in which these types of items are put together into a process, the standard would be more accessible to practitioners, and it would be less prone to becoming prematurely obsolete.

On the Development of Process-Oriented Dependability Standards

Michael Lyu, Chinese University of Hong Kong, China

Most software dependability standards -- including reliability, availability, safety, security, etc. -- are product-oriented in nature. While these standards are well-defined, quantitatively measurable, and closely related to customer perceived software quality, they suffer two major drawbacks: (1) They are usually applied to end products, which could be too late in the process to make any changes when the products do not meet the required dependability requirements. (2) Their verification becomes much harder when it comes to safety-critical applications whose dependability requirements are extremely high.

We advocate the formulation and usage of process-oriented software dependability standards as a complement to the current product-oriented standards for a comprehensive approach to software dependability evaluation. We classify three software reliability engineering phases where dependability is achieved: (1) modeling and analysis phase, (2) design and implementation phase, and (3) testing and evaluation phase. In the modeling and analysis phase, dependability is *planned*. The applicable standards are related to software *architectures*, including work breakdown structures, component dependability measures, reliability diagrams, component interactions, failure mode effect and criticality analysis, and fault tree analysis. In the design and implementation phase, dependability is *built*. The applicable standards are related to software *methods*, which include software fault avoidance and software fault tolerance mechanisms. Dependability is *confirmed* in the testing and evaluation phase. The applicable standards in this phase are related to software *metrics*. In addition to the usual product-related metrics like failure rate, reliability, failure intensity, fault density, we would recommend the measurement of metrics reflecting the maturity of software testing and evaluation process. They include: testing coverage measure, fault tolerance coverage factor, reliability growth tracking, and defect classification scheme.

These process-oriented standards, when appropriately applied and refined, provide end-to-end dependability solution to a complex software system. They facilitate early warning for timely process correction and feedback control for process improvement. Moreover, their domain-specific and architecture-based nature offers a feasible approach to safety-critical applications, whose dependability verification will heavily rely on the structure of the target systems.

Software Reliability Engineering Standards: A User View

John Musa, Software Reliability Engineering and Testing Courses, USA

An important issue in developing standards is who will use them and how. I will take a user perspective, based on my experience at AT&T as a software practitioner, manager of software practitioners, and consultant to many software projects, plus my experience in the last two years as an international independent consultant to a wide variety of software projects in different companies.

My sense is that the users and prospective users of software reliability engineering (SRE) standards are mostly practitioners, contracting organizations, and software lawyers. The majority of practitioners are new to SRE and are looking for guidance in how to practice it. The principle disciplines interested in SRE are software test, system engineering, software architecture, and project management. Contracting organizations need to ensure that the quality characteristics of reliability, delivery time, and cost they need in software-based systems they are receiving are met. Software lawyers are encountering an increasing number of lawsuits related to reliability, and what is adequate practice can be useful evidence in such suits.

If we combine the viewpoints of these different users, I would rank order the needs as follows: minimum adequate practice requirements, best practice recommendations, and standard terminology.

What do these needs imply? First, that product metrics are most important because they relate directly to customer values of reliability, delivery time, and cost. Process metrics are of interest only where they strongly influence important product metrics. Second, users are interested solely in proven practices. Third, they expect choices among practices to be resolved for them. Fourth, they expect recommendations for practices to be fit to the nature of the product and the development environment.

In summary, the standards that are most valued are those that provide useful guidance, especially to practitioners whose experience is moderate.

Do we need yet another *ility*?

Bill Everett, SPRE, Inc - Software Process & Reliability Engineering, USA

I am taking a different stand from the other panelists by questioning whether we need yet another *ility*. We have seen the increase in the number of disciplines that should be followed and the associated *ilities* that go along with them. There is reliab*ility*, availab*ility*, maintainab*ility*, and serviceability and now we have dependabILITY. What developers need is an integrated discipline for building *reliable* software products. Unfortunately, software developers today are faced with a number of standards they are told to follow in building their products but are reasonably confused about how they fit together. Also, they are asking "How do I follow all these standards and yet deliver my product within a reasonable time and cost?"

What I propose is integrating all these *ilities* together under one discipline (let's call it reliability) and developing one integrated standard for building *reliable* software. One needs to consider all the *ilities* mentioned earlier in building *reliable* software and an integrated software reliability standard should address all these, but in an *integrated* way.

References

[1] IEEE Standard Dictionary of Measures to Produce Reliable Software, Std 982-1-1988, The Institute of Electrical and Electronic Engineers, Inc, April 30, 1989.

IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software, Std 982-2-1988, The Institute of Electrical and Electronic Engineers, Inc, June 12, 1989.

[2] Recommended Practice for Software Reliability, ANSI/AIAA, R-013-1992, American National Standards Institute/American Institute of Aeronautics and Astronautics, 370 L'Enfant Promenade, SW, Washington, DC 20024, February 23, 1993.

[3] Norman F. Schneidewind, "Software Reliability Engineering for Client-Server Systems", Proceedings of The Seventh International Symposium on Software Reliability Engineering, White Plains, New York October 30, 1996 - November 2, 1996, pp.226-235.

[4] J.C. Laprie (Ed.), Dependability: Basic Concepts and Terminology, Springer-Verlag, Vienna, 1992.