How Can Software Reliability Measures Help You?

Users of software reliability measures have found [11] that developer-customer dialog is substantially enhanced. It is necessary to define "failure" for the system concerned. This definition is, in effect, a negative specification of requirements, and it generally leads to a clarification for everyone of what these requirements are. Reliability figures can readily be related to the operational costs of failure. Thus the customer comes to understand the real reliability requirements of the system in question. Similarly, the developer can relate reliability level requested to development costs. Thus, the stage is set for negotiation of an optimum solution for the customer of the sum of capital (purchase price of the system) and operational costs. By increasing the precision with which the customer's needs are met, productivity in the broadest sense is enhanced.

Software reliability measures guide the developer to better decisions. In the system engineering stage, they promote quantitative specification of design goals, schedules, and resources required. They let you determine quality level during test and thus provide the means for evaluating the effect of various actions on quality so that it can be controlled. The measures also help in the better management of project resources.

The user will also benefit from software reliability measures, because the user is concerned with efficient operation of the system. If operational needs with respect to quality are inaccurately specified, the user will either get a system at an excessively high price or with an excessively high operational cost.

The models associated with software reliability measurement structure and enhance both developer and customer understanding of software quality and the factors affecting it. The models include the time the program has been executing, software product characteristics, development process characteristics (including resources), and the operational environment or ways in which the software is used. These models permit the prediction, during test, of when various levels of quality will be obtained. Thus, once a quality objective has been chosen, release date can be predicted.

Developer and user, through accurate specification of what is a failure and what failure rate (or quality level) is optimum, can each increase customer satisfaction, provided the specification is met. The improved reputation resulting from high levels of customer satisfaction generally leads to a greater market share and higher profitability.

Basic Concepts

Software reliability is defined as the probability of failure-free operation of a computer program for a specified time in a specified environment. For example, a program might have a reliability of 0.82 for 8 hours of execution. A failure is a departure of program operation from requirements. Failure intensity, an alternate way of expressing software reliability, is defined as failures occurring with respect to some time unit. An expression equivalent to the reliability figure given above is that a program has a failure intensity of 0.025 failures per hour of execution. A fault is a defect in a program that causes a failure.
Software reliability is influenced by fault introduction resulting from new or modified code, fault removal that occurs in debugging, and the environment or ways in which the program is used. As a program is executed, failures will occur. If fault removal actions are taken (however imperfectly) in response to the failures, failure intensity will decrease as a function of time. Software reliability models characterize this change, as shown in Figure 1. A number of models have been developed [2-10]; see [1] for a classification and comparison of the models.

Figure 1. Software reliability model

References


Applications and State of the Art

Many applications for software reliability measurement have been developed, and considerable experience has been gained in its use [1]. We are now at the point where practicing software engineers in industry are independently testing the technology. This panel session presents a small sample of this work. It is not intended to be a comprehensive survey of applications. Rather than try to present such a survey or evaluate the state of the art, we will let the practitioners speak for themselves.

Author

John D. Musa is Supervisor of Software Quality at AT&T Bell Laboratories, Whippany, N.J. He has participated in or managed a variety of software products. His technical background and interests include software reliability, software engineering, and human factors. He is principal author (with A. Iannino and K. Okumoto) of the pioneering book "Software Reliability: Measurement, Prediction, Application," McGraw-Hill, 1987. He is a Fellow of the IEEE, cited for "contributions to software engineering, particularly software reliability."
Software reliability differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection. It is one of the most important aspects of software quality. Software reliability measurement: A software’s reliability is measured in two ways. A. Usage and reliability modeling B. Application of reliability measurement. A. Usage and reliability modeling: Reliability depends on the number of remaining faults that can cause a failure. There are two different types of models. 1. Usage specification 2. Reliability model. 1. Usage specification. In this spec Quality Assurance, Quality Control and Testing â€“ the Basics of Software Quality Management. In terms of manufacturing industry, it is similar to pulling a random item from an assembly line to see if it complies with the technical specs. Testing is the basic activity aimed at detecting and solving technical issues in the software source code and assessing the overall. Source code and design Detection Test Engineers, Developers At the testing stage or along with the development process. 6. Quality Assurance, Quality Control and Testing â€“ the Basics of Software Quality Management. If applied to the process of car The importance of software quality and the relationship of software reliability to software quality are discussed. The need for software reliability measures is demonstrated by outlining some possible applications. Basic software reliability concepts are presented, including software modeling. @inproceedings{Musa1987SoftwareQA, title={Software quality and reliability basics}, author={John D. Musa}, booktitle={FJCC}, year={1987} }. John D. Musa. Published in FJCC 1987. The importance of software quality and the relationship of software reliability to software quality are discussed. The need for software reliability measures is demonstrated by outlining some possible applications. Basic software reliability concepts are presented, including software modeling.