BOOK REVIEWS

Ensuring Software Reliability

Reviewed by Robert Megargle, Department of Chemistry, Cleveland State University, Cleveland, OH 44115.

REFERENCE: Neufelder, A. M., *Ensuring Software Reliability*, Marcel Dekker, Inc., New York, 1993, ISBN: 0-8247-8762-5, 242 pages.

Anyone who has worked on a medium to complex programming project, one that requires more than a few days to complete, knows how difficult and time-consuming it is to test software and find errors. It often requires several times more effort to locate and fix bugs than to design and write the program. At first we blame ourselves, thinking it is our own inadequacies that cause us to make so many mistakes. We soon learn that almost everyone has the same experiences. These realizations have led to studies of the process of developing software, undertaken to find out if there are ways to contain the errors or facilitate their locations and corrections. The current status of this field of study is reviewed in this book, which is Volume 38 of Dekker's extensive series on "Quality and Reliability" edited by E. G. Schilling.

The first part of the book contains a general discussion of the nature of software errors, where they come from, and what factors contribute to the problem. Many errors are due to unclear or incomplete functional requirements. The process of transferring a description of the task to be implemented from the minds of the intended users to the minds of the software developers is fraught with peril. Jargon, unstated assumptions, and other flawed communication is common. Poor initial planning also leads to frequent late changes to the requirements, resulting in hasty patches and reworked segments that tend to introduce more errors. Other problems are due to changes in environment, such as new hardware or upgrades to the software tools. Many errors really are coding or logic mistakes by the programmer. Introducing new errors while trying to fix old ones is also common. The first chapters serve to define terminology and give the reader an introductory overview of the subject of software reliability.

The book then discusses formal methods for documenting each error found. The nature of the error, the most probable source, the date and time of its discovery, its severity, how it was fixed, and other such information is recorded. Any kind of formal tracking or analysis of the error patterns requires reasonably objective data. It is the analysis of these patterns that allow problems in software development methodology to be discovered and corrected.

Usually the initial testing of a new program uncovers many errors in a short amount of time. As they are fixed, the error discovery rate decreases until only a few errors are found for extensive testing. When the error discovery rate is sufficiently low, the testing is stopped and the program is declared to be ready for use. Knowing when to make this decision is crucial to happy customers, a good professional image, fewer lawsuits, and more profits. Neufelder briefly reviews a number of software reliability models that rely on the formal error reports, and provide a more objective basis to make management decisions about the project.

The third part of the book deals with methods for improving software reliability. It is a collection of recommendations for writing programs that have fewer errors, in which errors are easier to detect, and where the routines can be better maintained. Neufelder favors structured coding techniques, keeping individual modules simple, minimizing global variables, and extensive error checking within the program modules. She recommends good source code documentation to reduce errors introduced during subsequent revisions. A very thorough list of information to be included is given.

This section also discusses strategies for testing the software. Included is a systematic plan to ensure all pathways through each module are tested. Each algorithm and logic decision point should be verified in all possible combinations. There must be a set of tests that verifies that each functional requirement has been met. Making sure software does not do what it is not supposed to do is more difficult than verifying that it performs correctly when used correctly. There are many more error cases in most programs than correct cases. "Software Fault Tree Analysis" and "Failure Mode Effect and Critical Analysis" are two tools presented to help with this problem.

The book provides a very brief introduction on the use of automated tools for software reliability. Some of the criteria for these tools, and the functions they might perform, are presented. Included are programs to capture and track errors, report results based on particular error models, produce test data cases, and provide configuration management to track the development history of a program. This section is more of an alert to the reader about the possibility of automated tools, as opposed to a comprehensive discussion of available products.

Finally, the book ends with a discussion of tactics for implementing formal software reliability methods in an organization. Neufelder shares her experience on the best ways to demonstrate the advantages of such a program, warns about containing the costs, and recommends keeping a list of lessons learned and successes achieved. As with any new idea that changes the conventional way of doing business, some resistance can be expected until the benefits are clearly established.

This book is a good introductory overview of software reliability, an emerging subdiscipline within the field of software engineering. The level of presentation in some sections is not sufficient to actually implement the procedures. However, the book gives good references that can be used to fill in the details. The book is well organized and understandable, but the style of writing is a detraction. It could benefit from good editing. There are too many sentences that end disconcertedly with prepositions. There is an awkwardness in numerous places from the repeated use of the same phrase, sentence after sentence. Aside from this, the content of the book is sound and worthy of a place on the bookshelves of those who make their living writing software.

