Quantifiable Software Assurance

Kishor Trivedi and John Board

Duke University, ECE Department

Durham, NC 27708-0291

(919)660-5269

{kst,jab}@ee.duke.edu

1. Fighting Bugs

Despite many advances in formal methods, programming methodology, testing, and software reliability growth modeling, software verification has not reached the stage to allow for the routine production of ultra-low defect software systems. Yet complex software-based systems are expected not to fail. A multi-pronged strategy to deal with software faults is thus essential.

A large percentage of software faults cause failures that are easily reproduced. These types of faults are relatively easily found and removed during the test/debug or early deployment phase. They have been christened Bohrbugs.

A small percentage of faults are, however, elusive. These Mandelbugs, may not manifest themselves under seemingly identical conditions. Retrying the same operation might not result in a failure manifestation. Likewise failover to an identical replica of the same software system might not result in a failure.

An interesting subtype of Mandelbug appears to have the characteristic that its failure manifestation rate increases with the time of execution. Such faults have been observed in many software systems and have been called aging-related bugs. Software rejuvenation is a proactive technique to postpone or eliminate failures caused by aging-related bugs.

A judicious combination of several fault removal and recovery techniques can be used to nearly eliminate a system level failure. Such techniques are already used in commercial high availability software systems such as IBM's Websphere.

2. Challenges and Key Questions

Our ultimate goal is to identify and deploy a set of fault mitigation and reduction strategies that lead to quantifiable improvements in system dependability, especially with respect to resilience against Mandelbugs. This work is thus complementary to continued work in formal methods and programming methodology which seeks to

stop bugs at the source. To accomplish this, we first need to determine by data analysis of large projects what fraction of faults are indeed Mandelbugs, and more specifically aging-related bugs. What factors affect these percentages? What is the trend over time in the relative prevalence of fault types? We have begun one such effort together with Allen Nikora of JPL examining data from several large NASA projects.

Software testing techniques detect and remedy faults, especially Bohrbugs, during the test/debug phase or the early deployment phase of a project. Although Software Reliability Growth Models (SRGM) have been well researched to gauge the progress of testing and debugging in reducing extant bugs, they have not been well accepted in practice, and the projection of the failure rate during the operational phase from these models is not yet well understood. One avenue for additional work in this area is to take a compositional view of software systems, modeling reliability at the level of modules and the interactions of modules. Activity along these lines has been ongoing and increasing.

Methods of dealing with Mandelbugs such as retry, failover to a replica, restart and reboot each have their own efficacies (coverages) and delays. These coverage and delay parameters have to be determined experimentally and their collective effects on system availability and other metrics studied via stochastic models. We recently did one such project at IBM for high availability WebSphere for a Telco application. What we found is that the awareness of the key factors needs to be engendered and the methods of formal or semi-formal expression of fault recovery subsystem design needs to be cultivated. Our experience indicated that we had to spend considerable time in "discovering" the key aspects of system behavior by asking many questions.

Once such a system availability model has been developed, sensitivity to parameters can be determined and the optimal order of the application of recovery strategies can be studied.

Automated construction of such stochastic models from a formal description of fault recovery behavior expressed in UML or similar languages will be desirable.

Being able to deal with very large models is another topic of continued research. As an example, in reliability modeling of a major subsystem of a commercial aircraft under design, special bounding methods had to be developed in order to solve the large reliability graph problem.

In the IBM WebSphere application we also needed to explore user-oriented metrics such as the number of telephone calls lost due to a failure. The way in which such metrics can be modeled and the set of key system parameters that affect these metrics was quite different than that for system availability models of the same system.

Development of stochastic models for security in response to malicious attacks needs to be carried out in framework similar to that for system availability modeling. We

developed one such model for the SITAR architecture under a DARPA program. Design and quantification of system survivability in face of natural disasters needs to be explored. We worked with Lucent Bell labs to help quantify four different architecture proposals for POTS survivability. Much additional research is needed on this topic.

Stochastic models make many assumptions about system behavior, distributions, independence and values of parameters. Hence validation against real measurement data is crucial but rarely done. This must change.

Prediction of impending failures due to aging-related bugs (and possibly other types of bugs) based on monitoring of system health is an important topic. Selection of key variables to be monitored and the monitoring frequency need to be explored further. Statistical methods such as design of experiments, time series analysis, support vector machines and other machine learning techniques are being applied.

Much of availability monitoring and quantification takes place in an off-line manner. It is important to develop on-line methods of monitoring and subsequently control of availability. We have had some initial success in implementing such a monitor in an IBM project.

3. Author Bios

Kishor Trivedi holds the Hudson Chair in the Department of Electrical and Computer Engineering at Duke University, Durham, NC. He has been on the Duke faculty since 1975. He is the author of a well known text entitled, Probability and Statistics with Reliability, Queuing and Computer Science Applications, published by Prentice-Hall; a thoroughly revised second edition (including its Indian edition) of this book has been published by John Wiley. He has also published two other books entitled, Performance and Reliability Analysis of Computer Systems, published by Kluwer Academic Publishers and Queueing Networks and Markov Chains, John Wiley. He is a Fellow of the Institute of Electrical and Electronics Engineers. He is a Golden Core Member of IEEE Computer Society. He has published over 350 articles and has supervised 39 Ph.D. dissertations. He is on the editorial boards of IEEE Transactions on dependable and secure computing, Journal of risk and reliability, international journal of performability engineering and international journal of quality and safety engineering.

John Board is an associate professor of Electrical and Computer Engineering at Duke University. He has been on the Duke faculty since 1987. He has expertise is in parallel computing, parallel scientific software, and performance evaluation.