N92-15870

1991

NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

MARSHALL SPACE FLIGHT CENTER
THE UNIVERSITY OF ALABAMA

DEVELOPMENT OF A CALIBRATED SOFTWARE RELIABILITY MODEL FOR FLIGHT AND SUPPORTING GROUND SOFTWARE FOR AVIONIC SYSTEMS

Prepared By:	Stella Lawrence
Academic Rank:	Professor
Institution:	Bronx Community College, Department of Engineering Technologies

NASA/MSFC:

Office:	Information & Electronic Systems Laboratory
Division:	Software & Data Management
Branch:	Systems Software
MSFC Colleague:	Kenneth Williamson
Contract No.:	NGT-01-008-021 The University of Alabama

T

The object of this project was to develop and calibrate quantitative models for predicting the quality of software. Reliable flight and supporting ground software is a highly important factor in the successful operation of the space shuttle program. Reliability is probably the most important of the characteristics inherent in the concept of "software quality." It is the probability of failure free operation of a computer program for a specified time and environment.[1]

A software reliability model specifies the general form of the dependence of the failure process on the principal factors that affect it: fault introduction, fault removal, and the environment.

Since some of the factors involved in the preceding are probabilistic in nature and operate over time, software reliability models are generally formulated in terms of random processes. Analytic expressions can be derived for the average number of failures experienced at any point in time, the average number of failures in a time interval, the failure intensity at any point in time, the probability distribution of failure intervals.

A good software reliability model gives good predictions of future failure behavior, computes useful quantities, is simple, widely applicable, and based on sound assumptions. Prediction of future failure behavior assumes that the values of model parameters will not change for the period of prediction.

The models use the Poisson process to model software occurrence, either the HPP (Homogeneous Poisson Process) or NHPP (Nonhomogeneous Poisson Process).

The models used in the present study consisted of: 1. SMERFS (Statistical Modeling and Estimation of Reliability Functions for Software).[2] There are ten models in SMERFS: error count models (generalized Poisson model, NHPP model, Brooks and Motley, Schneidenwind model, S-shaped reliability growth model) and time-between-error models (Littlewood and Verrall Bayesian model, Musa execution time model, geometric model, NHPP model for time between error occurrence, Musa logarithmic Poisson execution time model), 2. Kenneth Williamson's NHPP Binomial type software reliability model,[3] 3. Goel-Okumoto NHPP model.[4]

The software utilized consisted of the IMCE (Image Motion Compensation Electronics) software flight data, BATSE (Burst Transient Source Experiment, Gamma Ray Observatory) software, and a further program will also utilize POCC (Payload Operations Control Center for the Space Shuttle), Payload Checkout Unit Software for the Space Shuttle and HIT software (space shuttle telemetry systems).

Before discussing the results obtained with the models used in the present study, it must be kept in mind that software reliability modeling is just one of many tools. It cannot provide all the answers to the problems managers must face.

It must be taken as a bit of information, which along with others, is helpful in making a realistic judgement concerning a program's status.

For a first run, the results obtained in modeling the cumulative number of failures versus execution time showed fairly good results for our data. Plots of cumulative software failures versus calendar weeks were made and the model results were compared with the historical data on the same graph. If the model agrees with actual historical behavior for a set of data then there is confidence in future predictions for this data.

Considering the quality of the data, the models have given some significant results, even at this early stage. With better care in data collection, data analysis, recording of the fixing of failures and CPU execution times, the models should prove extremely helpful in making predictions regarding the future pattern of failures, including an estimate of the number of errors remaining in the software and the additional testing time required for the software quality to reach acceptable levels.

It appears that there is no one "best" model for all cases. It is for this reason that the aim of this project was to test several models. One of the recommendations resulting from this study is that great care must be taken in the collection of data. When using a model, the data should satisfy the model assumptions.

As previously stated, the data has to have the ability to correctly identify and measure what is desired. The data provided must satisfy the following: 1. It should be correctly recorded. 2. It should consist of samples that are random in nature. 3. It should be stated in CPU hours per failure, i.e. state CPU time as well as the date of the error. 4. All errors should be accurate.

Reliability should improve if the field software is corrected as failures occur. What about repeated failures due to the same fault? Fixing of faults leading to failures has to be properly recorded and properly attended to. The record of failures must be obtained for a sufficient length of time. Recent theory indicates that the failure intensity function probably decreases exponentially with time, i.e. a plot of the rate of occurrence of failures versus the number of faults found decreased asymptotically to zero.

There are also several recommendations regarding the use of the models: 1. The models require the insertion of various parameters. The models should be run with various values of these parameters, which should be carefully chosen for optimum results. 2. The data should be modeled piecewise, in addition to running the models for the total data. 3. Various forms of data input are provided including time between failure data and error count data and the model may yield different results for different types of data input.

4. The length of the trial should be a proportion of the expected life of the system; predictions made from a very small set of data tend to be poor. 5. The rate of manifestation of errors varies greatly from fault to fault, models which treat all faults as having the same rate may lead to optimistic bias estimates. Perhaps, some type of analysis should be performed to classify failures by severity, what kind of failure is it and is it critical or not?

To sum up the preliminary trials indicate that the models tested show much promise and that with their proper use and tailoring they are expected to yield an accurate reliability prediction for the flight and supporting ground software of embedded avionic systems.

REFERENCES

1. Musa, John S., Iannino, A., and Okumoto, Kazuhira, Software Reliability Measurement, Prediction, Application, McGraw-Hill Book Company, New York, 1987.

2. Farr, William H., Strategic Systems Department, Naval Systems Warfare Center, Dahlgren, VA 22448-5000, Smith, Oliver D., EG & G Washington Analytical Service Center, Inc., 1900 Dahlgren Road, Dahlgren, VA 22448, Sponsored by Strategic Systems Programs, Washington, DC 20376-5002, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS), Report No. NSWC TR 84-373, Revision No. 2, March, 1991.

3. Williamson, Kenneth, Non-Homogeneous Poisson Process Binomial Type Software Reliability Model, Preliminary Edition, EB41/Software & Data Management Division, EB42/Systems Software Branch, Marshall Space Flight Centeó, Huntsville, AL 35811, July, 1991.

4. Vienneau, Robert, Computerized Implementation of the Goel-Okumoto NHPP Software Reliability Model, ITT Research Institute, Beeches Technical Campus, Route 26N, Rome, N.Y. 13440, November, 1987.