www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 3 Issue 5 may, 2014 Page No. 6228-6233

A STUDY ON SOFTWARE RELIABILITY, RELIABILITY TESTING AND GOMPERTZ MODEL

Sandeep Sharma
M.Tech(CSE) Student
Arni University, Indora Kangra, India
Sanintel123@gmail.com

ABSTRACT

Reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period of time.There are various parameters which improves the reliability of the software.It is necessary to maintain the reliability of the software to keep track of correct information about any company the details of the information includes resource, money, employees, transaction details and many more, Now a days demand on complex systems has increased more rapidly. The size and complexity of computer systems has grown during the past decades in a very impressive manner.Due to the increase in size and complexity of the systems it become difficult to maintain the reliability of the system.Software reliability is closely related to safety engineering and system safety, in that they use common methods for their analysis and may require input from each other. Software reliability focuses on costs of failure caused by various threats, software failure and many more.Various approaches can be used to improve the reliability of the software, however, it is hard to balance development time and budget with software reliability. But the best approach to assure software reliability is to develop a high quality software through all of the stages of software life cycle.

KEYWORDS

Software reliability, Reliability testing, Gompertz Model,

1.INTRODUCTION

Software is Non-Tangible as it cannot be seen and touched, but it is very necessary for the successful use of computers. It is necessary that the reliability of software should be measured and evaluated, as it is in hardware[2]. IEEE 982.1-1988 defines Software Reliability Management as "The process of optimizing the reliability of software through a program that emphasizes software error prevention, fault detection and removal, and the use of measurements to maximize reliability in light of project constraints such as resources, schedule and performance[1].

One of the purposes of software engineering is to produce reliable software. Software plays a critical role not only in scientific and business applications, but also in all our daily life. Although several works have been done towards the production of fault-free software, developing reliable software is one of the most difficult problems facing software industry nowadays. Developing reliable software can be costly and time consuming process. Yet more, the fact that software project managers require accurate information about how software reliability grows in order to effectively manage their budgets and projects[1].

There has been extensive work in measuring reliability using mean time between failure and mean time to failure. Successful modeling has been done to predict error rates and reliability These activities address the first and third aspects of reliability, identifying and removing faults so

that the software works as expected with the specified reliability. These measurements have been successfully applied to software as well as hardware. But in this paper, we would like to take a different approach to software reliability, one that addresses the second aspect of reliability, error prevention.

2. ERRORS FAULTS AND FAILURES

The terms errors, faults and failures are often used interchangeable[3], but do have different meanings. In software, an error is usually a programmer action or omission that results in a fault. A fault is a software defect that causes a failure, and a failure is the unacceptable departure of a program operation from program requirements. When measuring reliability, we are usually measuring only defects found and defects fixed. If the objective is to fully measure reliability we need to address prevention as well as investigate the development starting in the requirements phase what the programs are developed to.

It is important to recognize that there is a difference between hardware failure rate and software failure rate.When the component is first manufactured, the initial number of faults is high but then decreases as the faulty components are identified and removed or the components stabilize. The component then enters the useful life phase, where few, if any faults are found. As the component physically wears out, the fault rate starts to increase. The following figure graphically shows the increase in failure rate due to side effect

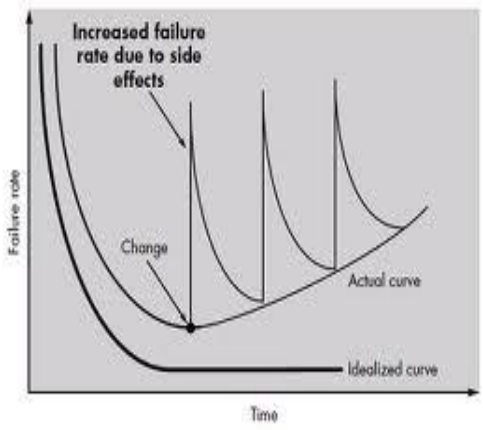

Figure 1

Software however, has a different fault or error identification rate. For software, the error rate is at the highest level at integration and test. As it is tested, errors are identified and removed. This removal continues at a slower rate during its operational use the number of errors continually decreasing, assuming no new errors are introduced. Software does not have moving parts and does not physically wear out as hardware, but is does outlive its usefulness and becomes obsolete.

3. RELIABILITY TESTING

The purpose of reliability testing is to discover potential problems with the design as early as possible and, ultimately, provide confidence that the system meets its reliability requirements.Reliability testing may be performed at several levels and there are different types of testing[2,3].Complex systems may be tested at component, circuit board, unit, assembly, subsystem and system levels. For example, performing environmental stress screening tests at lower levels, such as piece parts or small assemblies, catches problems before they cause failures at higher levels. Testing proceeds during each level of integration through full-up system testing, developmental testing, and operational testing, thereby reducing program risk. However, testing does not mitigate unreliability risk.

It is not always feasible to test all system requirements. Some systems are prohibitively expensive to test; some failure modes may take years to observe; some complex interactions result in a huge number of possible test cases; and some tests require the use of limited test ranges or other resources. In such cases, different approaches to testing can be used, such as accelerated life testing, design of experiments, and simulations.

The desired level of statistical confidence also plays an role in reliability testing. Statistical confidence is increased by increasing either the test time or the number of items tested. Reliability test plans are designed to achieve the specified reliability at the specified confidence level with the minimum number of test units and test time[5]. Different test plans result in different levels of risk to the producer and consumer. The desired reliability, statistical confidence, and risk levels for each side influence the ultimate test plan. The customer and developer should agree in advance on how reliability requirements will be tested.

A key aspect of reliability testing is to define "failure". Although this may seem obvious, there are many situations where it is not clear whether a failure is really the fault of the system. Variations in test conditions, operator differences, weather and unexpected situations create differences between the customer and the system developer. One strategy to address this issue is to use a scoring conference process. A scoring conference includes representatives from the customer, the developer, the test organization, the reliability organization, and sometimes independent observers. The scoring conference process is defined in the statement of work. Each test case is considered by the group and "scored" as a success or failure. This scoring is the official result used by the reliability engineer.

As part of the requirements phase, the reliability engineer develops a test strategy with the customer. The test strategy makes trade-offs between the needs of the reliability organization, which wants as much data as possible, and constraints such as cost, schedule and available resources. Test plans and procedures are developed for each reliability test, and results are documented.

4. IMPORTANCE OF RELIABILITY TESTING

The application of computer software has crossed into many different fields, with software being an essential part of industrial, commercial and military systems. Because of its many applications in safety critical systems, software reliability is now an important research area. Although software engineering is becoming the fastest developing technology of the last century, there is no complete, scientific, quantitative measure to assess them. Software reliability testing is being used as a tool to help assess these software engineering technologies.

To improve the performance of software product and software development process, a thorough assessment of reliability is required. Testing software reliability is important because it is of great use for software managers and practitioners.

5. TYPES OF RELIABILITY TESTING

Software reliability testing includes feature testing, load testing, and regression testing, reliability growth test and Reliability evaluation based on operational test [4].

I. .Feature test

Feature testing checks the features provided by the software and is conducted in the following steps:

Each operation in the software is executed once. Interaction between the two operations is reduced and Each operation is checked for its proper execution.The feature test is followed by the load test.

II. Load test

This test is conducted to check the performance of the software under maximum work load. Any software performs better up to some amount of workload, after which the response time of the software starts degrading. For example, a web site can be tested to see how many simultaneous users it can support without performance degradation.This testing mainly helps for Databases and Application servers.Load testing also requires software performance testing, which checks how well some software performs under workload.

III. Regression test

Regression testing is used to check if any new bugs have been introduced through previous bug fixes. Regression testing is conducted after every change or update in the software features. This testing is periodic, depending on the length and features of the software.

IV. Reliability growth test

This testing is used to check new prototypes of the software which are initially supposed to fail frequently. The causes of failure are detected and actions are taken to reduce defects. Suppose T is total accumulated time for prototype. n(T) is number of failure from start to time T. The graph drawn for n(T)/T is a straight line. This graph is called Duane Plot. One can get how much reliability can be gained after all other cycles of test and fix it.

$$ln\left[\frac{n(T)}{T}\right] = -\alpha ln(T) + b; \qquad .....Eq:1$$

solving eq.1 for n(T),

$$n(T) = KT^{1-\alpha}; \qquad .....Eq:2$$

where K is e^b. If the value of alpha in the equation is zero the reliability can not be improved as expected for given number of failure. For alpha greater than zero, cumulative time T increases. This explains that number of the failures doesn't depends on test lengths.

Designing test cases for current release

If we are adding new features to the current version of software, then writing a test case for that operation is done differently.

- First plan how many new test cases are to be written for current version.
- If the new feature is part of any existing feature, then share the test cases of new and existing features among them.
- Finally combine all test cases from current version and previous one and record all the results.

There is a predefined rule to calculate count of new test cases for the software. if N is the probability of occurrence of new operations for new release of the software, R is the probability of occurrence of used operations in the current release and T is the number of all previously used test cases then

$$NewTestcases_{(currentrelease)} = \left(\frac{N}{R}\right) * T$$

V. Reliability evaluation based on operational test

The method of operational testing is used to test the reliability of software. Here one checks how the software works in its relevant operational environment. The main problem with this type of evaluation is constructing such an operational environment. Such type of simulation is observed in some industries like nuclear industries, in aircraft etc. Predicting future reliability is a part of reliability evaluation.

6. GOMPERTZ MODEL

The gompertz model includes Standard Gompertz Model and Modified Gompertz model :

i. The Standard Gompertz Model

The standard Gompertz reliability growth model is often used when analyzing success/failure data and reliability data obtained in developmental reliability growth programs. The standard Gompertz model is most applicable when the reliability data follow a concave shape, as shown in the figure.

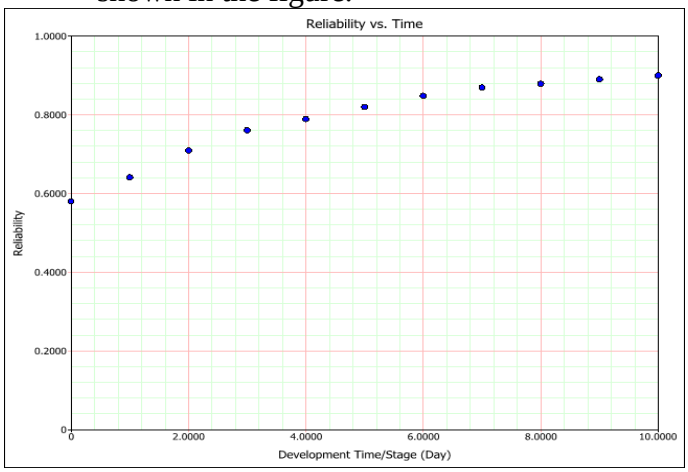

Figure 2

The standard Gompertz model is mathematically given by the following 3-parameter equation:

$$R = ab^{c^T}$$

where:

- $0 < a \leq 1$
- $0 < b < 1$
- $0 < c < 1$
- T: time, launch number or stage number, $T > 0$
- R: the system's reliability at T.
- a: the upper limit that the reliability approaches asymptotically as $T \infty$, or the maximum reliability that can be attained.
- ab: initial reliability at $T = 0$.

- *c*: the growth pattern indicator (small values of *c* indicate rapid early reliability growth and large values of *c* indicate slow reliability growth).

The estimated parameters in RGA are unitless. The solution for the parameters, given T_i and R_i, is accomplished by fitting the best possible line through the data points. Many methods are available, all of which tend to be numerically intensive. For more details about the parameter estimation method used in RGA.

ii. The Modified Gompertz Model

Sometimes reliability growth data with an S-shaped trend, such as the one shown in the next figure, cannot be described accurately by the standard Gompertz or logistic curves. Since these two models have fixed values of reliability at the inflection points, only a few reliability growth data sets following an S-shaped reliability growth curve can be fitted to them. A modification of the standard Gompertz curve overcomes this shortcoming by considering a shift in the vertical coordinate:

$$R = d + ab^{c^T}$$

where:

- *T*: time, launch number or stage number, $T > 0$.
- *R*: system's reliability at *T*.
- *d*: shift parameter.
- $d + a$: upper limit that the reliability approaches asymptotically as $T \infty$.
- $d + ab$: initial reliability at $T = 0$.
- *c*: growth pattern indicator (small values of *c* indicate rapid early reliability growth and large values of *c* indicate slow growth.

The Reliability vs. Time plot for this model looks like the figure shown next.

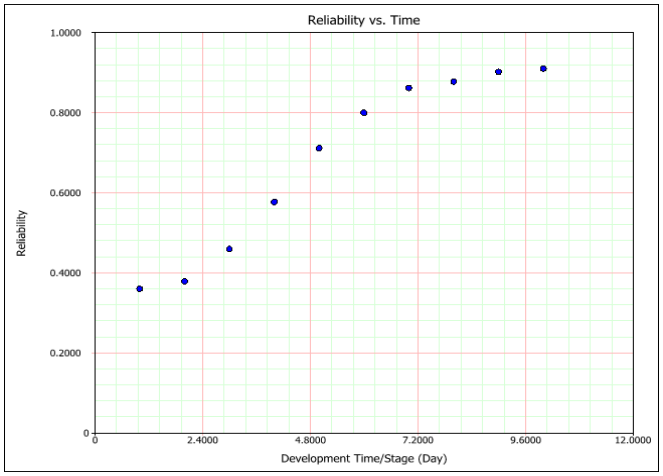

Figure 3

The parameters of the modified Gompertz model can be estimated using linear regression and confidence bounds can also be estimated; for more details about how to estimate confidence bounds. Many scenarios can be modeled with the S-shape behavior of the modified Gompertz. The S-shape behavior essentially distinguishes between multiple "phases" in the product reliability growth program. In the beginning of testing, scarcity in discovering problems and delays in identifying fixes can cause the improvement in reliability to be small. During the second phase, fixes become available and, with any additional discovered failures and implemented fixes, significant improvements are observed and the reliability grows at a fast pace. In the third phase, fewer and fewer failure modes are uncovered, as most of the failure modes have already been discovered. In addition, the reliability growth program might start running into limitations of technology and design that slow down the reliability growth.

7. CONCLUSION

This study includes Software reliability, reliability testing and Gompertz model. The increase in number of software faults and failures affect the performance of various services provided by the software Therefore software reliability has become more & more important. Reliability is the capability of software to maintain a determined level of performance within the time period. To improve the performance of software product and software development process, a thorough assessment of reliability is required. Gompertz reliability growth model is often used when analyzing success/failure data and reliability data

obtained in developmental reliability growth programs. We do focus on the various parameters which affects the performance of the software with the help of certain mathematical calculations we can determine the failure and the fault occurred with time. Identify those components which are not working properly apply various testing on it, update the software properly with time improves the software performance.

REFERENCES

1. A. Yadav & R. A. Khan "Critical Review on Software Reliability Models" International Journal of Recent Trends in Engineering, Vol 2, No. 3, November 2009.

2. Dr. Linda Rosenberg ,Ted Hammer, Jack Shaw "SOFTWARE METRICS AND RELIABILITY".

3. Chiu, Kuei-Chen "A discussion of software reliability growth models with time-varying learning effects "American Journal of Software Engineering and Applications July 20, 2013.

4. Zainab Al-Rahamneh, Mohammad Reyalat, Alaa F. Sheta, Sulieman Bani-Ahmad, Saleh Al-Oqeili "A New Software Reliability Growth Model: Genetic-Programming-Based Approach" Journal of Software Engineering and Applications, 2011, 4, 476-48.

5. Latha Shanmugam, Dr. Lilly Florence" An Overview of Software Reliability Models" International Journal of Advanced Research in Computer Science and Software Engineering

Sandeep Sharma did diploma in computer science engineering from Govt. polytechnic college kangra H.P(India) in December 2007.After diploma he joined B.tech in computer science engineering in Lovely Professional University (LPU) jalandhar Punjab(India) and got Degree in 2011.After B.tech he joined M.tech in computer science engineering in Arni university Indora H.P(India) and he is a M.tech(computer science engineering) research scholar at Arni university Indora H.P(India).