Software Reliability Engineering–A Review

Pankaj Nagar and Blessy Thankachan

***Abstract*—Software Reliability has a very important contribution towards software quality. Software reliability is the probability of the failure free operation of a computer program after a specific period of time. It is always required to test the reliability of a system to know its operations with 100% perfection. Whether it is a manufacturing system, or a software or a medicine even a human being. This article provides a general idea for observing the reliability of a software and a suggestive algorithm for its improvement by incorporating software engineering methodologies. This algorithm may help in right selection of model for improving software reliability.**

***Index Terms*—Reliability, Waterfall, Prototyping, Incremental, Rapid, Evolutionary, Agile**

I. Introduction

Computer software reliability has become an imperative standard in measuring service continuity. System performance is measured by how long it provides services without malfunctioning. Successful operation of any computer system depends on the successful operation of software. Thus, it is very important that the underlying software will operate correctly, perform its intended functions properly and fully deliver its desirable output. The most effective way to improve software quality and reliability is to integrate them into the design and manufacturing process. It is a useful tool that can be integrated into the early stages of the development cycle.

In the light of software engineering, software quality refers to two related but distinct notions that exist wherever quality is defined in a business context:

- Software functional quality refers how well it complies with or conforms to a given design, based on functional requirements or specifications.
- Software structural quality reflects how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly.

Structural quality is evaluated through the analysis of the software inner structure, in effect how its architecture adheres to sound principles of software architecture, its source code. In contrast, functional quality is enforced and measured by applying various methodologies of software testing.

Authors are with Department of Statistics, University of Rajasthan, Jaipur, Rajasthan, India (email: pnagar121@gmail.com, blessyt218@gmacom) .

A. Software Reliability Curve

Software errors have caused human fatalities. Software does not change over time unless changed or upgraded intentionally. Software does not age, rust, wear-out, or deform. Software failures may be due to errors, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of software or other unforeseen problems.

Over time, hardware exhibits the failure characteristics as shown in Figure 1. Known as bathtub curve.

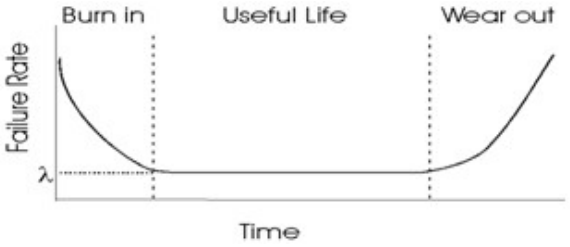

Fig. 1. Bathtub curve for hardware reliability

Software is not susceptible to the environmental maladies that cause hardware to wear out; therefore, the failure rate curve for software take the form of the "idealized curve" as shown in Figure 2.Undiscovered defects may cause high failure rates early in the life of a program. Once these are corrected (possibly without introducing other errors) the curve flattens. In the useful life phase, software will experience a drastic increase in failure rate each time an upgrade is made[1].

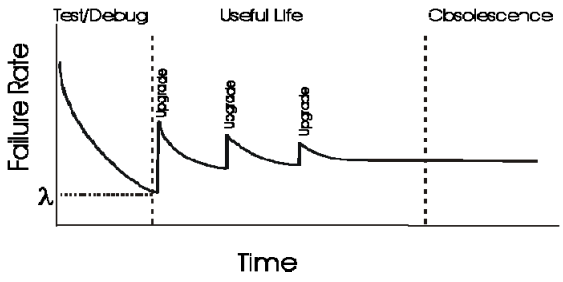

Fig. 2. Software Reliability curve

II. Software Development Methodologies

Every software development methodology acts as a basis for applying specific approaches to develop and maintaining software. Several software development approaches have been used since the origin of information technology.

Broadly speaking, the reliability of the software can be largely improved if appropriate engineering tools are integrated in software development methodology.

A. Waterfall Development

The Waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing, (validation), integration, and maintenance. The first formal description of the method is often cited as an article published by Winston W. Royce [2] in 1970 although he did not use the term "waterfall" in this article. The reliability of the software can be improved if:

- Special emphasis is laid on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
- Tight control is maintained over the life of the project via extensive written documentation, formal reviews, and approval/rejection by the user and information technology management occurring at the end of most phases before beginning the next phase.
- Waterfall model is used when the requirements are well understood and unlikely to change during system development.

B. Prototyping

Software prototyping, is the development approach of activities during which prototypes (incomplete versions of the software program) are developed [3]. The reliability can be improved if:

- Attempts are made to reduce inherent project risk by dividing a project into smaller segments and providing more ease-of-change during the development process.
- User is involved throughout the development process, due to which the likelihood of user acceptance of the final implementation is increased.
- Small-scale mock-ups of the system are developed following an iterative modification until the prototype evolves to meet the user's requirements.
- A basic understanding of the fundamental business problem is generated to avoid solving the wrong problem.

C. Incremental development

Various methods are acceptable for combining linear and iterative systems development methodologies, with the primary objective of each being to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process [4]. The reliability of software can be improved by:

- If series of mini-Waterfalls are performed, then all phases of the Waterfall should be completed for every small part of a system carefully, before proceeding to the next increment, or
- Overall requirements should be well defined before proceeding to evolutionary, mini-Waterfall development of individual increments of a system, or
- The initial software concept, requirements analysis, and design of software and system core should be defined via Waterfall, followed by iterative Prototyping, which culminates in installing the final prototype, a working system.
- Coupling between the modules should be reduced and cohesion within the modules should be increased [4].
- Increments should be relatively small, and each increment should deliver some system functionality [4]

D. Spiral Development

The spiral model combines the advantages of top-down and bottom-up concepts. It is a meta-model, a model that can be used by other models [4]. The reliability of software reliability can be improved if:

- Elaborate focus is applied to evaluate risks and weigh consideration of project continuation throughout the life cycle.
- While each trip around the spiral traverses four basic quadrants: (1) determine objectives, alternatives, and constraints of the iteration; (2) evaluate alternatives; Identify and resolve risks; (3) develop and verify deliverables from the iteration; and (4) plan the next iteration[5][6]. Efforts should be made to control and understand the interdependency among the modules.
- Reviews should be designed to ensure that design satisfies the requirements and is of good quality.

E. Rapid application development

Rapid application development (RAD) is a software development methodology, which involves iterative development and the construction of prototypes [4]. The reliability of the software can be improved if:

- Attempts are made to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- The primary aim is to produce high quality systems quickly, primarily via iterative Prototyping (at any stage of development), active user involvement, and computerized development tools like Graphical User Interface (GUI) builders, Computer Aided Software Engineering (CASE) tools, Database Management Systems (DBMS), fourth-generation programming languages, code generators, and object-oriented techniques etc.
- Project control involves prioritizing development and defining delivery deadlines. If the project starts to slip, emphasis should be on reducing requirements to fit the deadline, not in increasing the deadline.
- Joint application design (JAD), where users are intensely involved in system design, via consensus building in either structured workshops, or electronically facilitated interaction.
- Active user involvement is imperative.
- Iteratively produces production software, as opposed to a throwaway prototype.
- Proper documentation is done to facilitate future development and maintenance.
- Standard systems analysis and design methods are also fitted into this framework.

F. Evolutionary Development Method

This model is based on the idea of developing an initial implementation, exposing it to the user and then refining it through many versions until a final system is evolved [4]. The reliability of the software can be improved if:

- The process is visible.
- Systems are properly structured.

G. Rational Unified Process

It is an example of hybrid process model, which brings together elements from all generic process models. Fundamental best practices which are recommended to improve software reliability [4]:

- Develop software iteratively.
- Manage requirements
- Use components based architecture
- Visually model software
- Verify software quality
- Control changes to software

H. Component based software development

It is based on systematic reuse where systems are integrated from existing components or COTS (Commercial-OFF-the-shelf) systems [4]. The reliability of the software can be improved if:

- Requirements are not compromised as it may lead to a system that does not meet the actual need of the users.
- Control over the system is not lost as new versions of the reusable components may not be under the control of the organisation using them.

III. Computer-Aided Software Engineering

Computer Aided Software Engineering (CASE), in the field software engineering is the scientific application of a set of tools and methods to software which results in high-quality, defect-free, and maintainable software products [7]. It also refers to methods for the development of information systems together with automated tools that can be used in the software development process [8]. Two basic shortcomings of Computer-aided Software System Engineering (CASE) are [4]:

- Existing CASE tools automate routine activities but attempts to harness artificial intelligence technology to provide support for design have not been successful.
- CASE technology does not provide much support for team interaction, as it is one of the vital ingredient for good software engineering.

Efforts should be made to improve the above problems by the application of software engineering tools.

A. Integrated development environment

An integrated development environment (IDE) also known as integrated design environment or integrated debugging environment is a software application that provides comprehensive facilities to computer programmers for software development [3]. An IDE normally consists of a:

- Source code editor
- Compiler and/or interpreter
- B tools
- Debugger (usually).

IDEs are designed to maximize programmer productivity by providing tight-knit components with similar user interfaces. Typically an IDE is dedicated to a specific programming language, so as to provide a feature set which most closely matches the programming paradigms of the language.

IV. Agile Software Development

"Agile Software Development" is a group of software development methodologies based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. The Agile Manifesto [9] introduced the term in 2001. Agile management methods can also be applied in other development projects than software development

- Agile methods are adaptive rather than predictive.
- Agile methods are people-oriented rather than process-oriented.

Agile methods break work into small increments with minimal planning, and do not directly involve long-term planning. Iterations are short time frames that typically last for few weeks. Each iteration involves a team working through a full software development cycle including planning, requirements analysis, design, coding, unit testing, and acceptance testing. This reduces the overall risk and allows the project to adapt to changes quickly. Stakeholders produce documentation as they require. Iteration may not add enough functionality to warrant a market release, but the goal is to have an available release with minimum bugs at the end of each iteration [10]. Multiple iterations may be required to release a product or any new feature.

There are many agile development methods. Most promote collaboration, development, teamwork, and process adaptability throughout the life-cycle of the project.

A. Extreme Programming

Extreme Programming (XP) is a software development method which is intended to improve software quality and responsiveness to changing customer requirements. As it is a type of agile software development, it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. Extreme Programming (XP) is the most widely used agile methodology. XP shares the values espoused by the manifesto but goes further to specify simple set of practices. The XP is a lightweight software development methodology which consists in dividing a project into a series of iterations rather than an exhaustive structure based on the thorough and time-consuming analysis. Each iteration ends with a rigorously tested release that works in its limited way, and then fit it into a specific structure which is designed to simplify and expedite the process of software development. Extreme Programming was created by Kent Beck [10] during his work on the Chrysler Comprehensive Compensation System (C3) payroll project.

B. Scrum

Scrum approach has been developed for managing the system development process. Although the Scrum approach was originally suggested for managing product development projects, its use has revealed the fact that it can be used for the management of software development projects, and it can well be used to run software maintenance teams or as a general project/program management approach. This approach is developed for applying the industrial process control theory to system development resulting in an

approach that reintroduces the idea of adaptability , flexibility and productivity (Schwaber and Beedle 2002). It does not define any specific software development technique for the final implementation phase.

Scrum concentrates on the technique the team members should function in order to produce the system flexibility in a constantly changing environment. The main idea of Scrum is that system development involves many environmental and technical variables that are likely to change during the development process. This feature makes the development process complex and unpredictable.

V. Software Reliability Modeling Selection Algorithm

To study any system, it is necessary to experiment with the system itself or with the model of the system. Experimenting with the system itself may be very risky and expensive too. Therefore system studies are often done with models, which are substitutes of the system, instead of the system itself. A simulation model provides an alternative to analytical models as it describes a system being characterized in terms of its events, artifacts, interrelationships and interactions in such a way that one may perform experiments on the model, rather than on the system itself, ideally with indistinguishable results. Simulation is a technique of imitating the character of an object or process in a way that permit us to make quantified inferences about the real object or process. In the area of software reliability, simulation can mimic the key characteristics of the processes that create, validate, and revise documents and code [1].

An algorithm is designed for the correct selection of development model.

Algorithm:

```
Choose_model( )
{
Specify the problem and perform a requirement Analysis in concise
    If (Simulated Model Required)
    {
        Repeat until (success)
        {
            Plan the study
            Redefine the model
            Validate the model
            Run for more and more data, Verify the results and stop
        }
    }
    Else            // Use analytical model
    {
        Repeat until (success)
        {
            Plan the study
            Redefine the model
            Validate the model
            Run for more and more data, Verify the results and stop
        }
    }
}
```

An initial step is to analyze the problem to be solved in a concise manner. Based on the problem a model is defined. At this point it must be decided whether to use analytical model or simulation model. Once the model is decided, the model should be verified and execute a series of runs. Verification of results should be done after each run.

VI. Conclusions

Many models exist, but no single model can capture a necessary amount of software characteristics. There is no single model that is universal to all the situations. Selection of the appropriate model for software development is very necessary for improving the overall reliability of the software. For the right selection of the development model experimenting with the system itself will be risky and expensive therefore modeling can be done of the actual system. Simulation model can be used for the better selection of the right development model. The application of software engineering tools in each step from requirement analysis to implementation can highly improve the quality, efficiency and effectively of the software. Computer aided software engineering is also another milestone for development of highly reliable software. It can be improved by sufficiently understanding the type of the software, developing methodologies, and the environment in which it is working. It should be understood that sound software design is the building block of reliable software. The quality or the reliability of the software can be highly improved if development methodologies are improved. i.e. that improvement in the software development methodologies can improve the overall reliability of the software.

Acknowledgment

With the deepest gratitude Dr. Pankaj Nagar would like to thank the Editor in chief to consider this paper for publication in its reputed journal.

Blessy Thankachan would like to acknowledge and express her deepest gratitude to her guide Dr. Pankaj Nagar, for his magnificent support and guidance for the completion of this article.

She also thanks her family for their constant support and encouragement.

References

[1] Aasia Quyoum, Mehraj–Ud - Din Dar, S. M. K. Quadri “Improving Software Reliability using Software Engineering Approach- A Review” in International Journal of Computer Applications (0975-8885) Volume 10– No.5, November 2010.

[2] Royce “Managing the Development of Large Software Systems: Concepts and Techniques” in Proc. WESCON, IEEE Computer Society Press, Los Alamitos, CA, 1970.

[3] Software development methodologies. Webarticle.

[4] Ian Sommerville “Software Engineering” Eight Edition, Pearson Education, 2009.

[5] Barry Boehm (1996). "A Spiral Model of Software Development and Enhancement". In: ACM SIGSOFT Software Engineering Notes (ACM) 11(4):14-24, August 1986.

[6] Richard H. Thayer, Barry W. Boehm Tutorial: software engineering project management. Computer Society Press of the IEEE, 1986.

[7] Kuhn, D.L (1989). "Selecting and effectively using a computer aided software engineering tool". Annual Westinghouse computer symposium; Pittsburgh, PA (USA); DOE Project 6-7 Nov, 1989.

[8] P. Loucopoulos and V. Karakostas "System Requirements Engineering". McGraw-Hill, 1995.
[9] Beck, Kent "Manifesto for Agile Software Development". Agile Alliance, 2001.
[10] Kent Back,"Extreme Programming Explained: Embrace Change ",2nd Edn., ISBN 0201616416.

Pankaj Nagar is an Asstt. Professor, in Department of Statistics, University of Rajasthan, Jaipur, Rajasthan, India. He did his M.Sc. in Statistics in 1986, M.Phil. (Statistics) in 1987 and Ph.D. (Statistical Inference) in 1992. In addition to teaching assignment he is guiding research in the area of Applied Statistics & Computational Statistics. He is an author of six course based books for Under Graduate students. (Email: pnagar121@gmail.com)

Blessy Thankachan is a research scholar at the Department of Statistics, University of Rajasthan, Jaipur, India. She received her degree of Masters of Computer Application (MCA) from the prestigious University of Rajasthan. She is an author of course book for BCA students. Currently she is also HOD, Department of Computer Science, Apex Institute of Engineering and Technology, Jaipur, Rajasthan. She manages all the curricular and extracurricular activities associated with the department.(Email: blessyt218@gmacom)