Approaches for Enhancing Reliability of Software Product

Priyanka#1, Trilok Gabba*2

#1,2 Department of Computer Science & Engineering
BITS, Bhiwani, India

Abstract— In modern world, we are highly dependent upon computer for most of our works. As we know, all computers are controlled by software. So, to operate a computer in a proper manner, software reliability is very necessary.

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. The high complexity of software is the major contributing factor of Software reliability problems. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability. For good reliability, two approaches have to be used, namely, reactive and proactive approach.

This paper provides an overview of Software reliability, hardware reliability, reactive and proactive approaches.

Keywords—Software Reliability, Hardware Reliability, Reactive Approach, Proactive Approach

I. INTRODUCTION

As we know computers are playing a very big role in our life. So, it is virtually impossible to conduct many day-to-day activities without the aid of computer systems which are controlled by software. As more reliance is placed on these software systems, it is essential that they should operate in a reliable manner. So failure in software products can result in a high monetary, property or human loss.

NASA Software Assurance Standard, NASA-STD-8739.8, defines software reliability as a discipline of software assurance that:

1. Defines the requirements for software controlled system fault/failure detection, isolation, and recovery.
2. Reviews the software development processes and products for software error prevention and/or reduced functionality states.
3. Defines the process for measuring and analysing defects and defines/derives the reliability and maintainability factors.

II. RELIABILITY

Software does not break or wear out so the term reliability is often misunderstood by us in the field of software. Reliability affects the system reliability. It either works in a given environment or it does not. There are two types of reliability:

A. Hardware Reliability

Hardware component have three phases in its life
i) Burn in
ii) Useful life
iii) Wear out phase

These phases are shown in its bath tub curve in below fig. 1.1(The curve has a bath-tub like shape so it is known as “bath-tub” curve).

Burn-in phase: Due to initially testing in the premises of the organization failure rate is often quite high initially and it goes on decreasing gradually with time in hardware reliability in this phase.

Useful life period: Failure rate is approx constant. So it has a straight line in bath-tub curve.

Wear out phase: Due to aging of the components failure rate increases in wear out phase.
B. Software Reliability

It differs from hardware reliability as it does not have wear out phase in its curve. The expected curve for software is shown below fig 1.2. Software may be retired only if it is no longer in use. Some of the contributing factors are given below:

1. Environmental change
2. Technological change
3. Lot of changes in requirements
4. Increases in complexity of software
5. More difficulty in maintenance of software
6. Deterioration in structure of the code
7. Slow execution speed of software
8. Poor GUI (Graphical User Interfaces)

Software reliability reflects the design perfection, and hardware reliability reflects manufacturing perfection. Here, it differs. The high software complexity is the main problem in software reliability. Software reliability is not a function of time. No good quantitative methods have been developed to represent software reliability without excessive limitations. Software reliability can be improved by using various approaches, however, many a times it is quite difficult to balance between development time and budget with software reliability.

Software reliability is an important attribute of software quality, together with functionality, usability, performance, serviceability, capability, maintainability, and documentation. It is difficult to reach a certain level of reliability when any software have a high complexity, system developers tend to push complexity into the software layer, with the rapid growth of system size and it can be done by upgrading the software.

Table 1. Software VS Hardware

<table>
<thead>
<tr>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software defects are mainly design defects</td>
<td>Hardware defects are mainly manufacturing defects.</td>
</tr>
<tr>
<td>Software does not have a wear-out phase.</td>
<td>Hardware has a wear-out phase.</td>
</tr>
<tr>
<td>Software problems can be fix by periodic restarts.</td>
<td>Hardware problems can’t be fix by periodic restarts.</td>
</tr>
<tr>
<td>Software reliability is not a function of operational time.</td>
<td>Hardware reliability is a function of operational time.</td>
</tr>
<tr>
<td>Environmental factors do not affect Software reliability.</td>
<td>Environmental factors affect hardware reliability.</td>
</tr>
<tr>
<td>Software interfaces are purely conceptual.</td>
<td>Hardware interfaces are purely visual.</td>
</tr>
<tr>
<td>Software reliability can’t be predicted from any physical basis.</td>
<td>Hardware reliability can be predicted from any physical basis.</td>
</tr>
<tr>
<td>There are no standard components for software.</td>
<td>There are standard components for hardware.</td>
</tr>
</tbody>
</table>

III. APPROACHES FOR SOFTWARE RELIABILITY

Today, there is a period of recession in the software industry. But slowly it is growing all over the world. Optimization of the software reliability is the major objective of the software industry. Combining both the reactive approach and the proactive approach makes the software quite more reliable. These two approaches are the great achievement in making the software reliable.

A. Reactive Approach

Reactive approach uses the checkpoint in software industry. Checkpoint is used to mask the error and avoids failure. Software reliability uses the reactive approach for the considerable gain. The Performance Testing activity is often uses the reactive approach for performance management. For most of the cases, the system performance can never be checked during the early phases of Software Development Life Cycle phases (SDLC). Performance is often thought of only as a last activity after the System Testing phase. Performance of any software depends on system architecture or system design, many a time it has a very high cost then we have to think about it. In certain cases because of huge deviation in the performance constraints we have to put the system in trash.
Dealing with a performance problem at the end is always not a better approach for software development. During early life cycle phase approach are known as reactive approach as there is not much importance is given to the system. It is more a ‘fix-it-later’ approach which is not that effective.

B. Proactive Approach

This approach is used during initial phase of life cycle of software development. For this reason this approach is very important than reactive approach. It uses the dynamic medication of the program for avoiding the future failure. It anticipates the performance problem well in advance and adopts techniques for mitigating them.

The disadvantages of ‘fix-it-later’ approach are well understood & engineering practices are adopted to analyse the system design in performance angle.

The integration of performance engineering activities with the SDLC phases is provided in the below Table 2

**Table 2 SDLC phases integrated with Performance Engineering**

<table>
<thead>
<tr>
<th>SDLC Phases</th>
<th>Performance Engineering Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Check whether Performance fall under CTO</td>
</tr>
<tr>
<td>Requirement Analysis</td>
<td>Performance Requirement analysis</td>
</tr>
<tr>
<td></td>
<td>Workload Modeling</td>
</tr>
<tr>
<td>Architecture &amp; Design</td>
<td>POC Validation</td>
</tr>
<tr>
<td></td>
<td>Performance Modeling Review</td>
</tr>
<tr>
<td>Implementation</td>
<td>Code Profiling</td>
</tr>
<tr>
<td>Testing</td>
<td>Performance Testing</td>
</tr>
<tr>
<td></td>
<td>Performance Tuning</td>
</tr>
<tr>
<td></td>
<td>Capacity Planning</td>
</tr>
<tr>
<td>Release &amp; Maintenance</td>
<td>Deployment architecture review</td>
</tr>
</tbody>
</table>

IV. CONCLUSIONS

There are many way to enhance the reliability of software product. Software reliability can be enhanced by the concept of combining use of reactive and proactive approach. To deal with the break point in software we use reactive approach. To make software more reliable we have to advocate the combination of check point and rollback with on line software version change as a practical technology. These techniques provide us considerable gain in reliability. This gain increases as the probability of a failure being transient increases and as the decrease in failure rate obtained by removing a fault increases.

We believe that the general concept of combining reactive and proactive approaches can produce rich dividends for enhancing software dependability.

REFERENCES