International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 3, Number 5 (2013), pp. 371-376

http://www. irphouse.com /ijict.htm

A Holistic Qualitative Approach to Software Reliability

S. Guru Narayanan[1] and Vibhore Gupta[2]

[1]Manager- R&D, Schneider Electric, Sector 126, Noida, India.
[2]Engineer- R&D, Schneider Electric, Sector 126, Noida, India.

Abstract

This paper discusses the reliability driven approach towards developing software solutions and products. Requirements deficiencies were the cause of 99% of their field reliability problems. This has been expanded by Brendan Murphy to say that the problems on computer systems were mostly due to requirements deficiencies and interface weaknesses. This paper aims to provide a consolidated checklist, methods and tools that add to strength the existing requirement engineering process. It will facilitate to convert customer requirements into robust product specifications and it also proposes a way of how to deal with changing requirements. All these methods elaborate on identifying the risks and its mitigation through standard procedures. This paper also deals with Interface based Software Design approach where reusability of the already tested software module can be used for the new product or solution to reach high level reliability. The paper throws light on redundancy based architecture to achieve high reliability. It also describes some checklist for effective software coding. It also discusses the various testing methods that ensure to meet reliability objectives without compromising product features. These procedures defines a customer oriented and holistic approach, when adhered to, along with the usual activities of SDLC, will add considerable value to the requirement engineering domain and ensures that final product is engineered through proper design ,quality control measures and software testing procedures.

Keywords: Reliability, Requirement Engineering, Reusability, Software development process.

1. Introduction

Whenever any software development model is chosen, reliability has always been of topmost concern. The reason is simple to understand. The product needs to be able to cope with the ever changing needs of the customer. Product needs to be versatile enough to adapt itself to various situations and challenges. Reliability for software systems can be defined as "Failure-free operation for a specified period of time and under specific set of inputs". Keeping this in mind, it is of utmost importance that certain processes should be followed in pitch-perfect manner. Following are the guidelines, which are divided into different phases, to illustrate the process which enables us to move closer to reliable systems.

2. Requirement Gathering for Reliability

This is the process where maximum ambiguities are injected into the system and if they are not addressed early, the effort required to repair them in later stages is very cumbersome. This re-work may interfere with the timelines and with the budgeting of the project as well. This stage deals with collecting information that are necessary for developing the product and it is sub divided into three stages.

2.1 Analyzing the Available Data

This phase calls for a permanent co-operation between the marketing team and the requirement gathering team to analyze the current needs and also try to anticipate the future ones. Marketing team plays a very important role in keeping market needs up-to date and in building trust relationship with the customers. Continuous interaction with the marketing people also helps in coming up with the right proposal to the customer. Analysis of the existing project in the same domain can be very useful in this context. It will help in understanding the various challenges faced and strategies used to cope with the changing requirements. Domain analysis will also helps in breaking complex system into sub components and understanding their functionality. This break-up will also help in estimating the probability of change in future. Allocating adequate budget and making a well informed team in this phase is very important.

2.2 Requirement Gathering and Elicitation

To accomplish this task, groups must be formed to gather specific type of information. These groups must be provided with a questionnaire that should extensively covers all aspects of requirements. Observing every answer and reaction of the customer will help in drilling down to his major issues and also helps in understanding his preferences. These groups must be trained to improve their cognitive abilities and should be taught to engage in direct and in-direct questioning. Keeping an open mind and ability to gauge customer's intention should be the key when interview is being conducted. Visualization and dependency of various requirements can be easily done by preparing pictographs and preparing dependency matrices. These requirements should be updated in the repository for future reference.

2.3 Creating Systems Specifications

Here, once the requirements are gathered, feasibility study is performed. This study helps in realizing the feasibility of various requirements and their implementation. This study provides a consistent method to forecast and analyze the risks involved. These matrices should consist of priority index and gravity of the risk. These matrices will pave the way for versioning and future testing procedures. These differentiations will help us in building the prototypes, if necessary and identify the contingency and mitigation plan for high risk requirements. As the requirement change, impact analysis should be done with development time and cost point of view. It is always important to put in place the product evolution strategy by anticipating the future needs and enhancing the skills of the team accordingly.

3. Software Engineering Process

Choosing a proper SDLC should be based on by analyzing the requirement and the market needs. Normally Agile process is now preferred approach for the changing requirement and faster time to market. Nevertheless, when implementing some standards or under known requirements, a V Model SDLC will reduce the time and cost where verification and validation is planned concurrent for each phase. For Mission critical and High Risk projects, Spiral model will be more suitable where prototypes are first built by continually interacting with customers.

Organization should have strong process like CMMI/ ISO 9001:2000/IEC 15504 in place as to convert requirement into high quality products and systems. Clear communication should be established between stake holders and project team. Regular meetings with Customers will reduce the risk and will help to understands requirements in broader and end-user perspective. Resources deployed in the project should have experience in the product development on similar fashion. Proper training should be given on Domain Engineering and Software Technology, Process, Methods and Tools. Use of CASE tools will help in improving the productivity, quality and improve time to market.

4. Software Design for Reuse and Reliability

Software Design for reliability must make sure that the Systems to be built can be broken into reusable modules or components based on the detailed Application Domain Engineering Analysis. Even though reusable modules add more cost, payback is compelling in the long run where by reusing validated modules into the system reduces the software failure risks and saves the cost for future projects.

Using COTS as a component will also be useful for faster implementation and also reduces failure risks. APIs designed to communicate between modules should be robust and should not change often or it will introduce more cost and bugs. Redundancy is also to be incorporated in the Systems Design so as to improve the availability of the systems. Proper information should be conveyed in the Failure logs

for faster resolution. Design should also have proper error recovery procedures. System Resource utilization like (CPU usage) and System Requirements like under Normal and heavy Usage should be properly designed and communicated. Using proven Design patterns will solve considerable design risk issues and improve reliability.

5. Software Implementation for Reliability

All developers in the team should follow same coding standards. Secure coding is necessary for protection from hackers .Code should comply with Cyber security Standards. Software should be able to handle all types of valid and invalid inputs, and have strong exception handling mechanisms for stack overflow, type checking etc. Memory management should be carefully done. While doing multi-threaded programs Shared objects address corruption, Dead-locks and Race-conditions should be meticulously thought and should be eliminated. Code Review using static review tools and Peer Review will highly useful to solve bottle necks in the programming.

6. Testing Strategy for Software Reliability

Testing is the most powerful way to improve Software Reliability. Customers and Stake holder's feedback should be invited during Alpha, Beta Testing. Test scenarios should accommodate commonly used software inputs under normal and heavy load conditions to remove errors. Unit Testing, Integration Testing, System Testing should be extensively performed. Test Coverage with respect to requirement to be done to find missing features. Automated Test is mandatory for Larger Software Systems having huge test cases. Security Testing is necessary for Sensitive Information Software Systems. Degraded Testing like Negative testing, Recovery Testing and Destructive or Fault Injection Testing is necessary for Robust Performance. It's to better to have performance limits of Software defined by conducting Stress Testing and communicate clearly to the stakeholders. Soak Testing or Endurance Testing is crucial to understand if the system is able to tolerate Sustained load for longer duration. Suitable Software Testing Life cycle (STLC) process is to be deployed for easy maintenance and for change request, bug tracking and resolution.

7. Conclusion

The main purpose of this paper is to achieve reliability of highest order and strengthen the whole process of software development by adopting various techniques and standards. Reliability is a complex issue having many dimensions to be addressed. This paper clearly dealt with reliability issues and how to address them in all phase of Software Life cycle. All these techniques and methods are subjected to various goals of the project and can be altered according to the nature of the project. These processes when followed meticulously will certainly drive the software systems towards better

reliability. Moreover, with software systems complexity evolving at exponential rates, this strives for more reliable systems will go on uninterrupted in search of more such models, process. Methods and tools

References

[1] Elizabeth Hull, Ken Jackson, Jeremy Dick- Requirements Engineering
[2] Ian Sommerville, Pete Sawyer - Requirements Engineering: A Good Practice Guide.
[3] Kenny Kerr Best Practices for Writing Efficient and Reliable Code with C++/CLI
[4] M.R.Lyu Reliability Oriented Software Engineering :Design, Testing and Evaluation Techniques
[5] Roger S. Pressman 2005. *Software engineering : a practitioner's approach* (sixth edition)