Soft Testing Life Cycle: A guideline to model, analysis and verify NFR

Anirban Bhar
Department of Information Technology
Narula Institute of Technology, Kolkata, West Bengal, India

Dr. Sabnam Sengupta
Department of Information Technology
B. P. Poddar Institute of Management and Technology, Kolkata, West Bengal, India

Abstract- At early stage in the lifecycle, during the Inception phase, one of the issues to be considered as part of requirements gathering is to identify non-functional requirement (NFRs), also called quality of service (QoS) or simply quality requirements. Identifying the right NFRs for the particular system will keep impact on the technical decisions of the developer to design the initial architectural strategy. So it is also necessary to validate these requirements. This work is a guideline to deal with NFRs at early and every stage of developing a software.

Keywords – Life-cycle, NFR, QoS, requirements.

I. INTRODUCTION

Non-Functional Requirements, also known as system qualities controls some attributes of the system mainly Reliability, Scalability, Usability, Maintainability and Security. Although there is no formal definition or complete list of NFR [1] and NFRs are among the most critical and expensive to deal with [2]. Currently the software industries have highlighted the importance of NFRs to sustain in the competitive market and nowadays it is gaining popularity to be considered as the integral part of development, even treat NFRs as the first class of requirement. For a software product it is important to identify non-functional requirements for successful development and deployment. As per the RADC consumer-oriented attributes (TABLE 1) the NFR are listed those are applicable for all the information system but for special classes of software additional NFRs may be applicable [10].

Table 1: The RADC Software Quality Consumer-Oriented Attributes

<table>
<thead>
<tr>
<th>Acquisition</th>
<th>User Concern</th>
<th>Quality Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance-How well does it function?</td>
<td>How well does it utilize a resource?</td>
<td>Efficiency</td>
</tr>
<tr>
<td></td>
<td>How secure is it?</td>
<td>Integrity</td>
</tr>
<tr>
<td></td>
<td>What confidence can be placed in what it does?</td>
<td>Reliability</td>
</tr>
<tr>
<td></td>
<td>How well will it perform under adverse conditions?</td>
<td>Survivability</td>
</tr>
<tr>
<td></td>
<td>How easy is it to use?</td>
<td>Usability</td>
</tr>
<tr>
<td>Design-How valid is the design?</td>
<td>How well does it conform to requirements?</td>
<td>Correctness</td>
</tr>
<tr>
<td></td>
<td>How easy is it to repair?</td>
<td>Maintainability</td>
</tr>
<tr>
<td></td>
<td>How easy is it to verify its performance?</td>
<td>Verifiability</td>
</tr>
<tr>
<td>Adaptation-How applicable is it?</td>
<td>How easy is it to expand and upgrade its capability or performance?</td>
<td>Expandability</td>
</tr>
<tr>
<td></td>
<td>How easy is it to change?</td>
<td>Flexibility</td>
</tr>
<tr>
<td></td>
<td>How easy is it to interfere with another system?</td>
<td>Interoperability</td>
</tr>
<tr>
<td></td>
<td>How easy is it to transport?</td>
<td>Portability</td>
</tr>
<tr>
<td></td>
<td>How easy is it to convert for use with another application?</td>
<td>Reusability</td>
</tr>
</tbody>
</table>

II. RELATED WORK

According to the literature, most of the consideration to develop a software goes to functional requirements. Some work has been done on non-functional requirement also with the gaining popularity of NFRs and in order to sustain in the competitive market.

A work has been carried out to identify issues in testing NFRs considering the quality factors along with their impact and effect on testing [2]. Some goal oriented approaches to handle NFR have been suggested at requirement engineering phase such as i*, GRL, SIG [13, 14, 15, 16]. In these work NFRs are identified and decomposed as FR and NFRs are identified as the soft goal.
Some effort has been made to propose some model for NFR framework like Optimization model [17] and Process-Oriented model [18] but the limitation in scope of non functional requirements and lacking of a single tool to integrate several quantitative methods leave the scope for further enhancement of those works.

Even some work has been done with some individual Non Functional Aspect like Reliability [19, 20], Scalability [21], Reusability and Replaceability [22], Testability [23] etc.

An effort has been made to detect all types of software bugs and faults [24] but the fault tolerance approach is still under research to make much reliable software.

These works has deep impact in their respective domain and provide considerable outcomes but these works has been done either with some particular already developed software or only by considering one or two NFRs among many. We feel the requirement of a model that can deal with many different important NFRs for that particular application from the very early stage of development.

III. APPROACH FOR NFR ANALYSIS

Testing consumes almost 70% of resource required to develop a system [3,4,6,7,8,9]. According to L. M. cysnerio “Overlooking a crucial requirement” and “Modelling only Functional Requirements” are among the top 10 risks of Requirement Engineering [11]. The main reason behind is that modelling of NFR and implementing those in the information system is not an easy task. The universally accepted fact is that for the acceptability of any software system NFR play very important role because if NFR are not satisfied, results goes into low acceptability which goes against the product in the increasing competitive market, which fails to meet the expectations of stakeholders [12]. This work provides a guideline to model, analysis and verify NFR.

NFR for a particular software system requires considering the following issues:

1. Identification: Need to identify the particular NFRs for the particular system.
2. Relevance boundary: For example performance may be important for main application, but may be irrelevant and costly for administration and supporting applications.
3. Dependency: NFRs are expected to be independent to each other or dependent within a particular defined cluster of NFRs so that evaluation and testing them becomes impact-less of the other system qualities.
4. Modeling the NFRs: NFRs should be stated with clear objectives, measurable and testable idea, otherwise it will be tough to model and application may be impact-less.

Figure 1 shows the steps of approaches to achieve these goals.

Figure 1. Step-wise Approaches for NFR Analysis
1. Identification and elicitation of NFRs
Identification of NFRs is achieved by the goal based approach that reports the following (Figure 2):
• Identify the stakeholders and their priority rating from 5 to 1 (5 is the highest priority)
• Identify the goals and sub-goals are expected with the help of the stakeholders
• Identify NFRs and point out how the goals are to be reached under constraints.

2. Analyzing and identifying of the boundary of the NFRs
This can be easily achieved by the priority matrix (Table 2) based approach that deals with priority rating from 1 to 5 by the stakeholders of all the identified NFRs.

```
<table>
<thead>
<tr>
<th>Stakeholders and Priority</th>
<th></th>
<th>NFR 1</th>
<th></th>
<th>NFR 2</th>
<th></th>
<th>NFR 3</th>
<th></th>
<th>...........</th>
<th></th>
<th>NFR n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder 1</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Stakeholder 2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Stakeholder m</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Rating</td>
<td>3.46</td>
<td>3.2</td>
<td>2.33</td>
<td></td>
<td>3.13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

3. Analyzing and identifying the inter-dependencies of NFRs
This can be easily achieved by an already established approach: Soft Goal Interdependence Graph which is Adapted from L, Chung, B. Nixon, E. Yu and J. Mylopoulos, “Non-Functional Requirements in Software Engineering”, Kluer Academic Publishers, 2000 (Figure 3).
4. Modeling the NFRs
Modeling the identified NFRs required to rely on the vague ideas to measure the quality attributes. Figure 4 shows an example of this approach. (Source: Budgen, 1994, pp 60-1).

5. Identifying test strategies
In order to identify test strategies we propose a Risk matrix (Figure 5) in which Non Functional Threats are mapped to the Focus Area. A typical Risk matrix looks like the following:

6. Test cases design
In simple terms test case refers to a documentation which specifies input, pre-conditions, set of execution steps and expected result. With the above identifications and modeling those cases of a particular application is generated. The general template of the test cases is shown in Figure 6.
7. Test execution, Result Analysis and Checking benchmarks and recommendations

This phase is actually overlapping three phases and totally dependent on the particular application. Several modules of this phase may be discriminated by their boundary or maybe not. Figure 7 is a probable diagrammatic approach to undergo these phases.

III. CONCLUSION AND FUTURE WORK

This work provides a guideline to deal with application oriented NFRs. It guides the software programmers to state the desired outcomes. Measurable NFRs can provide the proof to achieve the goals of application oriented software systems.

This proposed model for the NFR framework relies on stakeholders’ views in initial stage which is dependent on their perception of the functionality of the software system. Moreover, their perceptions may be conflicting and dependant of their knowledge about the system and their viewpoints. Although considering wise stakeholders this work can provide a self sufficient framework to analyze, model and verify NFRs but in future we want to extend our work by introducing a method that will map the stakeholders’ conflicting point of views in order to overcome their psychological conflicts of perceptions about the system.

As a conclusion, we believe that from the early stage of development, the appropriate treatment of NFR can help to improve quality attributes of the software in order to improve end-users’ satisfaction to the final products as well as to improve development time and reduce development cost.

REFERENCES


[20] Dr. Linda Rosenberg, Ted Hammer and Jack Shaw, SOFTWARE METRICS AND RELIABILITY.