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 Arrtibutes

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

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

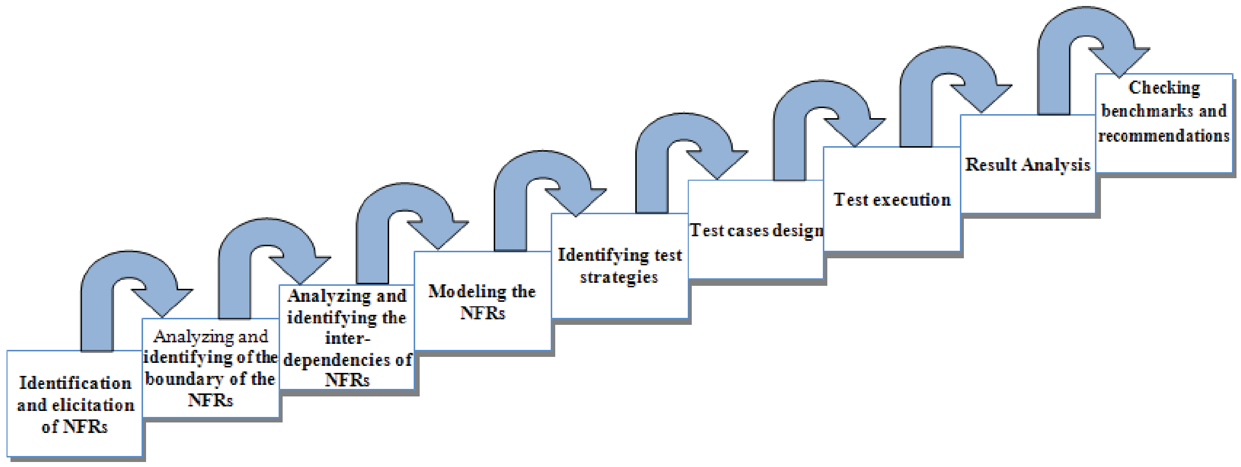
Testing consumes almost 70% of resource required to develop a system [3,4,5,6,7,8,9]. According to L. M. cysnerio “Overlooking a crucial requirement” and “Modelling only Functional Requirements” are among the top 10 risks of Requirement Engineering [11]. The main reason behind is that modelling of NFR and implementing those in the information system is not an easy task. The universally accepted fact is that for the accaptability 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.

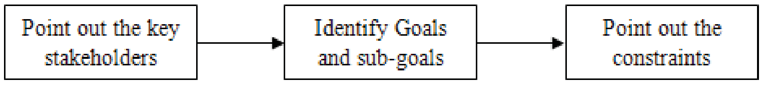
Figure 1. Step-wise Approaches for NFR Analysis


Identification and elicitation of NFRs
Analyzing and identifying of the boundary of the NFRs
Analyzing and identifying the inter-dependencies of NFRs
Modeling the NFRs
Identifying test strategies
Test cases design
Test execution
Result Analysis
Checking benchmarks and recommendations

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.

Figure 2. Identification of NFRs


Point out the key stakeholders
Identify Goals and sub-goals
Point out the 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.

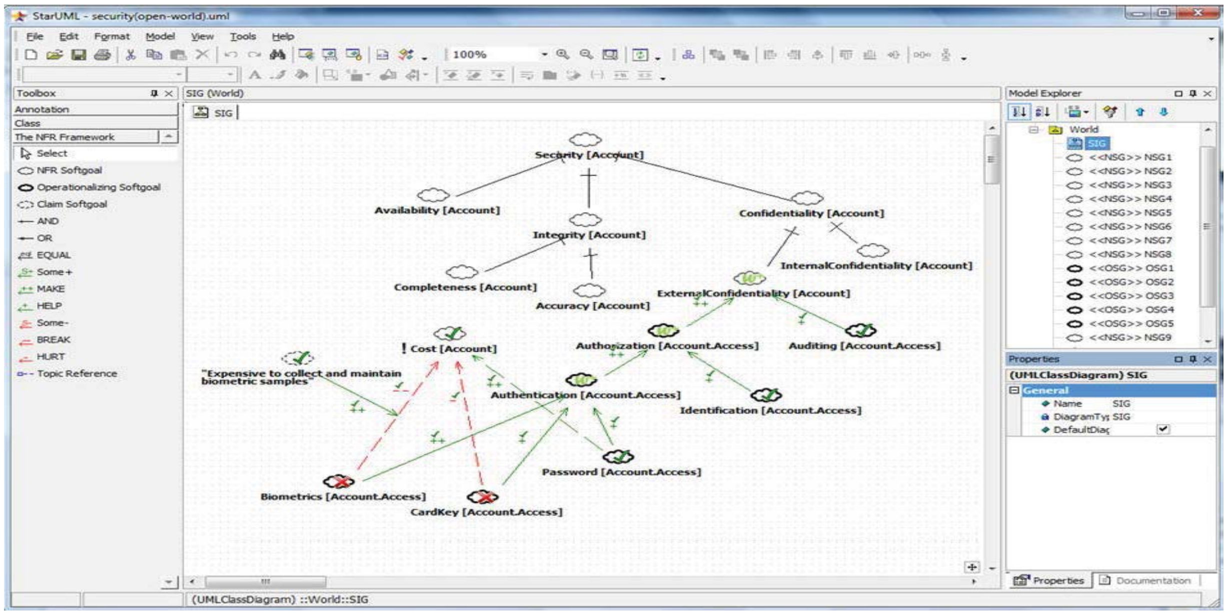
Table 2. Priority Matrix

Stakeholders and Priority		NFR 1	NFR 2	NFR 3	....	NFR n
Stakeholder 1	5	5	4	3		3
Stakeholder 2	4	4	4	2		4
. . .						
Stakeholder m	4	4	3	3		4
Rating= $\frac{\sum (\text{Stakeholder Priority} \times \text{NFR Priority})}{5m}$		3.46	3.2	2.33		3.13

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).

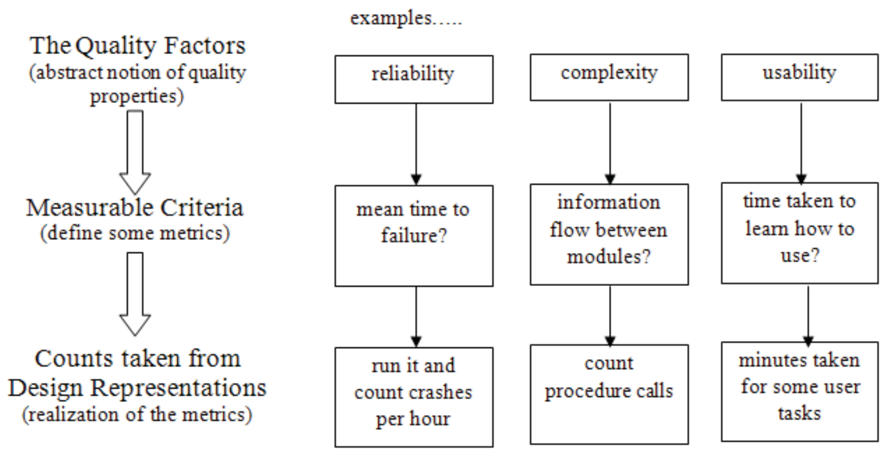
Figure 3. Soft Goal Interdependence Graph



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).

Figure 4. Example of Modeling the NFRs


examples.....
The Quality Factors
(abstract notion of quality properties)
Measurable Criteria
(define some metrics)
Counts taken from Design Representations
(realization of the metrics)
reliability
complexity
usability
mean time to failure?
information flow between modules?
time taken to learn how to use?
run it and count crashes per hour
count procedure calls
minutes taken for some user tasks

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:

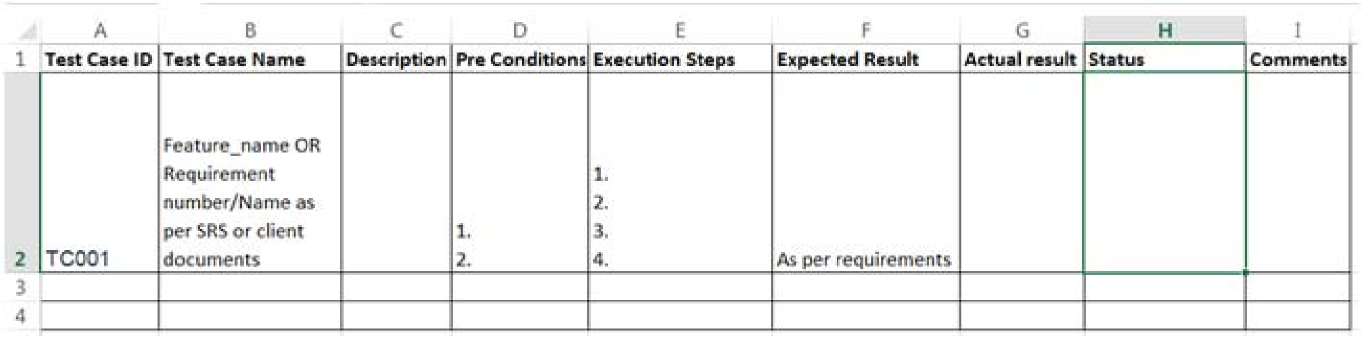
Figure 5. Example of a Typical Risk Matrix

Focus Areas							
Domain	Threat	Online Transaction Processing	End of Day (EoD) Processing	Adhoc Reports	Interfaces	Application Behaviour	Infrastructure Utilization
Scalability	Processing Overlap	HI	HH	MI	HI	HI	MI
	Concurrency	HH	HI	LI	MI	HH	MI
	Integration Complexity	MI	MI	NI	HH	NI	LI
	Horizontal Scalability	HI	MI	NI	MI	MI	NI
Reliability	Processing Overlap	MI	MI	LI	LI	MI	NI
	Stress Conditions	MI	HH	MI	MI	HI	MI
	Concurrency	HH	MI	LI	LI	MI	NI
	Prolonged Usage	HI	MI	MI	MI	HH	NI
HH→High Impact + High probability of Occurrence HI→High impact MI→Medium Impact LI→Low Impact NI→Negligible Impact							

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.

Figure 6. General Template of Test Case

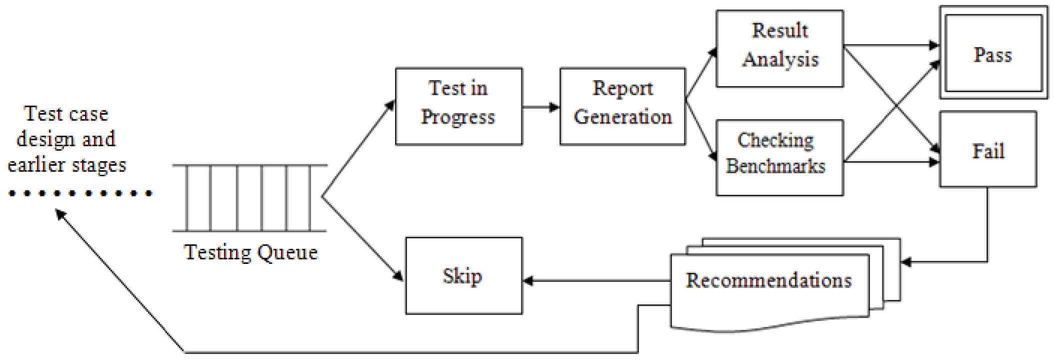


	A	B	C	D	E	F	G	H	I
1	Test Case ID	Test Case Name	Description	Pre Conditions	Execution Steps	Expected Result	Actual result	Status	Comments
2	TC001	Feature_name OR Requirement number/Name as per SRS or client documents		1. 2.	1. 2. 3. 4.	As per requirements			
3									
4									

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.

Figure 7. Diagrammatic Approach to Test execution, Result Analysis and Checking benchmarks and Recommendations



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

[1] Representing and Using Non-Functional Requirements: A Process-Oriented Approach,John Mylopoulos, Lawrence Chung and Brian Nixon

[2] ISSUES IN TESTING OF SOFTWARE WITH NFR, Pratima Singh and Anil Kumar Tripathi, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.4, July 2012

[3] Pressman, R. S. (2005). Software Engineering: A Practitioner's Approach (6th ed.). New York:-McGraw-Hill Publication.

[4] Srinivasan Desikan,Gopalaswamy Ramesh ,Software Testing :Principles and Practices, Pearson,2006

[5] Rex Black, Foundations of software testing2008, cengage learning press.

[6] Sommerville, I. Software Engineering.

[7] Pankaj Jalote: Practical approach to software engineering.

[8] Rajib Mall ,Fundamentals of Software Engineering, Third Edition, PHI Publication.

[9] Glenford J Myaers ,The Art of Software testing, second Edition ,John willey.

[10] T. P. Bowen et al., :Specification of software quality attributes," Rep.RADC-TR-85-37, Rome Air Development Center, Griffiss Air Force Base, NY, Feb. 1985.

[11] Luiz Marcio Cysneiros, Member,Julio Cesar Sampaio do Prado Leite, Member, Nonfunctional Requirements:From Elicitation to Conceptual Models, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 30, NO. 5, MAY 2004.

[12] Lawrence Chung and Julio Cesar Sampaio do Prado .T. Borgida et al. On Non-Functional Requirements in Software Engineering (Eds.): Mylopoulos Festschrift, LNCS 5600, pp. 363–379, 2009.© Springer-Verlag Berlin Heidelberg 2009.

[13] Sam Supakkul and Lawrence Chung. Integrating FRs and NFRs: A Use Case and Goal Driven Approach. Proc. SERA 04, pages 30–37, 2004.

[14] Goal Modelling in Requirements Engineering: Analysis and Critique of Current Methods Evangelia Kavakli , Pericles Loucopoulos, DOI: 10.4018/978-1-59140-375-3.ch006, http://www.igiglobal.com/chapter/goal-modeling-requirements-engineering/23011

[15] Jane Cleland-Huang, Raffaella Settimi, Oussama BenKhadra, Eugenia Berezhanskaya, Selvia Christina ,Goal-Centric Traceability for Managing Non-Functional Requirements,ICSE'05, May 15– 21, 2005, St. Louis, Missouri, USA.

[16] Jonathan Lee and Nien-Lin Xue, Analyzing User Requirements by Use Cases: A Goal-Driven Approach, National Central University. IEEE Software J u l y / August 1999 0740-7459/99/1999 IEEE

[17] Amy Affleck, Aneesh Krishna and Narasimaha R Achuthan, Optimal Selection of Operationalizations for Non-Functional Requirements. Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling (APCCM 2013), Adelaide, Australia.

[18] John Mylopoulos, Lawrence Chung and Brian Nixon, Representing and Using Non-Functional Requirements: A Process-Oriented Approach. IEEE Transactions on Software Engineering.

[19] Sandeep Sharma, A STUDY ON SOFTWARE RELIABILITY, RELIABILITY TESTING AND GOMPERTZ MODEL. International Journal Of Engineering And Computer Science ISSN: 2319-7242 Volume 3 Issue 5 may, 2014 Page No. 6228-6233.

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

[21] Anirban Bhar, Narula Institute of Technology & Dr. Sabnam Sengupta, B.P.P.I.M.T, Kolkata, Proto-Spiral: A Hybrid SDLC Model for Measuring Scalability Early in Development Using a Probabilistic Approach, ETCC 2014.

[22] M. Donatelli, A. Omicini, G. Fila and C. Monti, TARGETING REUSABILITY AND REPLACEABILITY OF SIMULATION MODELS FOR AGRICULTURAL SYSTEMS. Proceedings of the 8th ESA Congress, 2004, 237-238.

[23] Pratima Singh and Anil Kumar Tripathi, Treating NFR as First Grade for Its Testability. Journal of Software Engineering and Applications, 2012, 5, 991-1000

[24] Anirban Bhar, Soumya Bhattacharyya, "Detecting Software Bugs towards Efficient Software development: A Model Based Approach", INTERNATIONAL JOURNAL OF LATEST TRENDS IN ENGINEERING AND TECHNOLOGY, VOLUME 6 ISSUE 2 - NOVEMBER 2015.