International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 03 - Issue 02 || February 2018 || PP. 61-65

A View Based Model to Improve Quality Factors during Early Stage of Development

Anirban Bhar[1], Soumya Bhattacharyya[2]
[1, 2]*(Department of Information Technology, Narula Institute of Technology, Kolkata, India)*

Abstract: Software Development has started its journey for betterment of human being and to build an ideal future world which surrounds with optimum needful software. Till now there are various way and processes of Software development to fulfill different requirements at different time of Software Development Life Cycle (SDLC). The right Non Functional Attributes, known as Quality Attributes for the particular system surely may keep impact on the technical decision making of the developer to design the initial architectural strategy. Two aspects of quality a developer need to focus on are, firstly, what components it should contain to be a quality product rather than just be a good product and secondly, what are the strategies to imply them in the software during development to make remarkable impacts on an organizations philosophy. This paper presents a development process that focuses on defining appropriate non-functional properties and strategies to imply them for the particular Information System (IS) during development.

Keywords: Architectural Design, Information System, NFR, Quality Attributes, SDLC

I. INTRODUCTION

Quality definition of software may differ from person to person. As per Oxford dictionary quality is the degree of excellence. As per William Edwards Deming quality is the fitness for purpose. As per the famous American quality control expert Armand Vallin Feigenbaum quality is the best for the customer's use and selling price. But finally there should be some standards. ISO defines quality as the totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs. Now the question is how a software developer will define quality- The product which meets the customer requirements. As per the customers point of view quality is the required functionality is provided with user friendly manner. These are some quality definitions from different perspective. By studying the previous work on the same path we identify that there are two major aspects to define software quality [1]:

1) Conformance to specification: Quality that is defined as a matter of products and services whose measurable characteristics satisfy a fixed specification – that is, conformance to an in beforehand defined specification.
2) Meeting customer needs: Quality that is identified independent of any measurable characteristics. That is, quality is defined as the products or services capability to meet customer expectations – explicit or not.

There are many general quality attributes, and describing them all in detail could alone fill thousands of pages. For a given system, it is not possible to set the dial for every quality attribute in that list. For one thing, the cost would astronomical and the time to market would reach infinity. But, in addition, different quality attributes tend to have a negative influence on other attributes. For example, to maximize the security of a system, one would lock the server in a room connected to no other devices which has a negative impact on usability. Some choices make to increase reliability, such as distributed transactions or persistent message queues might have a negative effect on performance. Without a structured approach, implementing non functional testing can result in a failure to take the correct performance demands into consideration, in effect estimating the future use of the system upon vague suppositions. Perhaps the most complex activity during application development is the transformation of a requirement specification into application architecture. Our aim is to identify and define the appropriate system specific quality attributes and the strategies to imply them during development phases of that particular Information System.

II. LITERATURE SURVEY

A systematic literature review conducted by Svensson et al. [2] found no more than 18 empirical research studies centered on investigating the benefits and limitations on methods around NFRs for five identified areas: elicitation, dependencies, level of quantification, cost estimation, and prioritization. The authors of this systematic review themselves have conducted several empirical studies on the topic. Svensson et al. targeted several aspects on NFRs in [3], whilst in [4] they focused on issues related to requirements prioritization. A different perception was reported in [3] for some NFR aspects depending on the role of the interviewee, that supports the idea of replicating empirical studies for different role types. Other empirical works from the authors in more general subjects occasionally provided further evidence for NFRs, e.g., [5][6].

Several studies [7, 8, 9] have looked at non functional requirements interdependencies; for example, Carlshamre et al. identified six different interdependency types in industry [7]. Research related to classification and measurement of quality requirements are also introduced in literature [10, 11]. Olsson et al. conclude that for a method to be successful, it is important that it is flexible enough to handle the diverse nature of quality requirement [11].

Tang et al.'s work on architecture design rationale [12] provides evidence of the requirement of our subject of research and also it is highly relevant for software architects. This paper discusses the role of the architect in comparison to the dedication to different tasks and the design of NFRs appears third in the list (of interest for 64.2% of interviewees), right after overall system design (86.4%) and requirements or tender analysis (81.5%). However, we feel the lack of discussion on the relationship of software architecture and NFRs. Ali Babar et al. [13], in the same field have reported observations about documentation and validation of software architectures.

Considering the role of NFRs, design architecture and goal assessment, Perry and Wolf has proposed an approach [14] to use architectural style for constraining the architecture and coordinating cooperating software architects. They also propose that rationale, together with elements and form, constitute the model of software architecture.

III. C ONCEPTUAL MODEL

All the IT Development Projects, defined as a project where an IT system (application, software, infrastructure or other IT system) is designed, constructed and implemented. As per the experience of the work the more important nonfunctional requirements are, the greater the implied risk to IT project success if they are not fulfilled from the architectural design. On the other hand, several NFR approaches in the conceptual and architectural design could help an IT project deal with NFRs. As per Fairbanks architecture is a risk driven discipline [16]. The prerequisite to dealing with architectural design is the awareness of the risk. The conceptual model is adapted from Eltjo R. Poort et al. [17] is elastrated in Fig. 1.

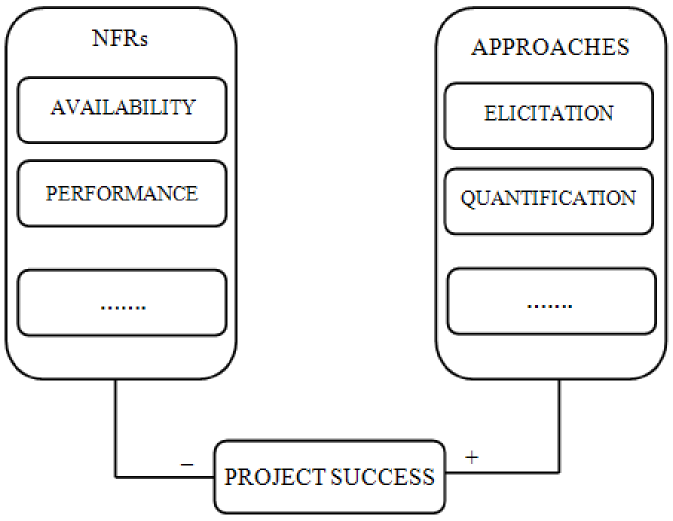

Figure 1: The Conceptual Model

The requirement to integrate NFRs into the conceptual models is to tackle the problem of representing NFRs, and to understand their impact on conceptual model design. This aproach also allows design decisions to become more systematic, since the conceptual model will explicitly deal with design constraints.

This strategy proposes the use of a lexicon, the Language Extended Lexicon (LEL) [18], as an driving force to integrate the NFRs into both the ER and the OO models. This can be achieved by using lexicon symbols to construct both the NFR graph [19] and the conceptual model.

Once all the NFRs are represented in the conceptual model, a verification process is necessary. Such a verification entails checking the NFRs can be represented in a graph with those in the conceptual model. Every NFR that exists in one model must exist in the other one.

IV. CASE STUDY: CONCEPTUAL MODEL OF A LABORATORY INFORMATION SYSTEM

The proposed conceptual model was used in a case study by Luiz Marcio Cysneiros et. al [20]. It was carried out during the development of an LIS for a clinical analysis laboratory that performs about 18,000 tests per day. The laboratory has 33 different places to attend patients all over the city and all of them use the system. The system is based on client–server architecture using four servers and more than 150 clients. All the information in the system is supposed to be stored on a historical basis so that any data from any patient has to be available for at least 30 years. Three different teams developed the system, and each team was responsible for a specific laboratory area. In this study the teams were divided into three areas: an administrative area, an attendance area and a processing area. The teams were composed as follows:

- Processing area: one of the authors, a junior systems analyst, one senior programmer and a junior programmer.
- Administrative area: one senior analyst, two senior programmers and one junior programmer.
- Attendance area: one senior analyst, one junior analyst and two senior programmers.

This case study describes that the Changes Before Software Delivery includes all changes due to NFR satisficing made in the software after the teams started to code and before the software was delivered to the client. It also explains the Changes After Software Delivery includes all changes due to NFR satisfacing during the maintenance phase.

V. PROPOSED MODEL

How to measure the importance of each type of NFR for a project? The experts in a pre-survey workshop agreed that simply asking for the number of requirements for each type of NFR is not valid. Intuitively, a project could have only a few performance requirements that are nevertheless critical for the system. Conversely, it could have more requirements of another type that are not critical.

The project success construct consists of five dimensions that are designed to reflect the interests of the three main stakeholders [15]. Meeting time and budget corresponds to project success from a managerial perspective, as does efficient use of resources. Customer satisfaction is included to reflect the perspective of the customers, and solution quality is the dimension that measures the success from the perspective of the development team.

The need for ways to manage NFRs has led several researchers to propose methods and techniques for dealing with NFRs. A set of similar methods and techniques, related to the same requirements engineering activity, that can be used to deal with or manage NFRs (or requirements in general) is defined as an NFR approach.

In this work the first step of the proposed approach is to identify the appropriate requirements by interviewing the set of sensible users. The interview questions may be a list of technical and non technical queries. Table 1 describes a sample questionnaire set and the following are the points to keep in mind during the elicitation phase.

- Maintain a complete list of NFRs
- For each NFR, formulate atleast one or more questions
- Give visibility of the impact of answering a question one way or another
- Capture the responses to each of the questions
- Give each NFR a priority or weighting

NFR	Questions	Impact	Answers	Priority
Availability	Are there any requirements regarding system "up time"? This may be specified in terms of Mean Time Between Failures (MTBF).	The higher the availability, the longer the time to market.	Availability is a key product feature. The product must have a MTBF of 60 days.	High

Table 1: A Sample Questionnaire Set to Interview the Users

After identification of the NFRs then the tasks and the domains are identified to elicitate the proper requirements. It need to be analyze and accepted after one or more iteration (as described in Figure 2). Here we want to mention that the interdependancies of all the NFRs are important to identify so that no NFR may cause any conflict with another.

Finally in requirement management phase the all requerments including the NFRs are integrated in the system and keeping every thing in mind, with the emphasis of the identified application based NFRs finally the prototype is designed.
If all the prototypes of a particular Information System works fine, then it is expected that the whole system will work fine.

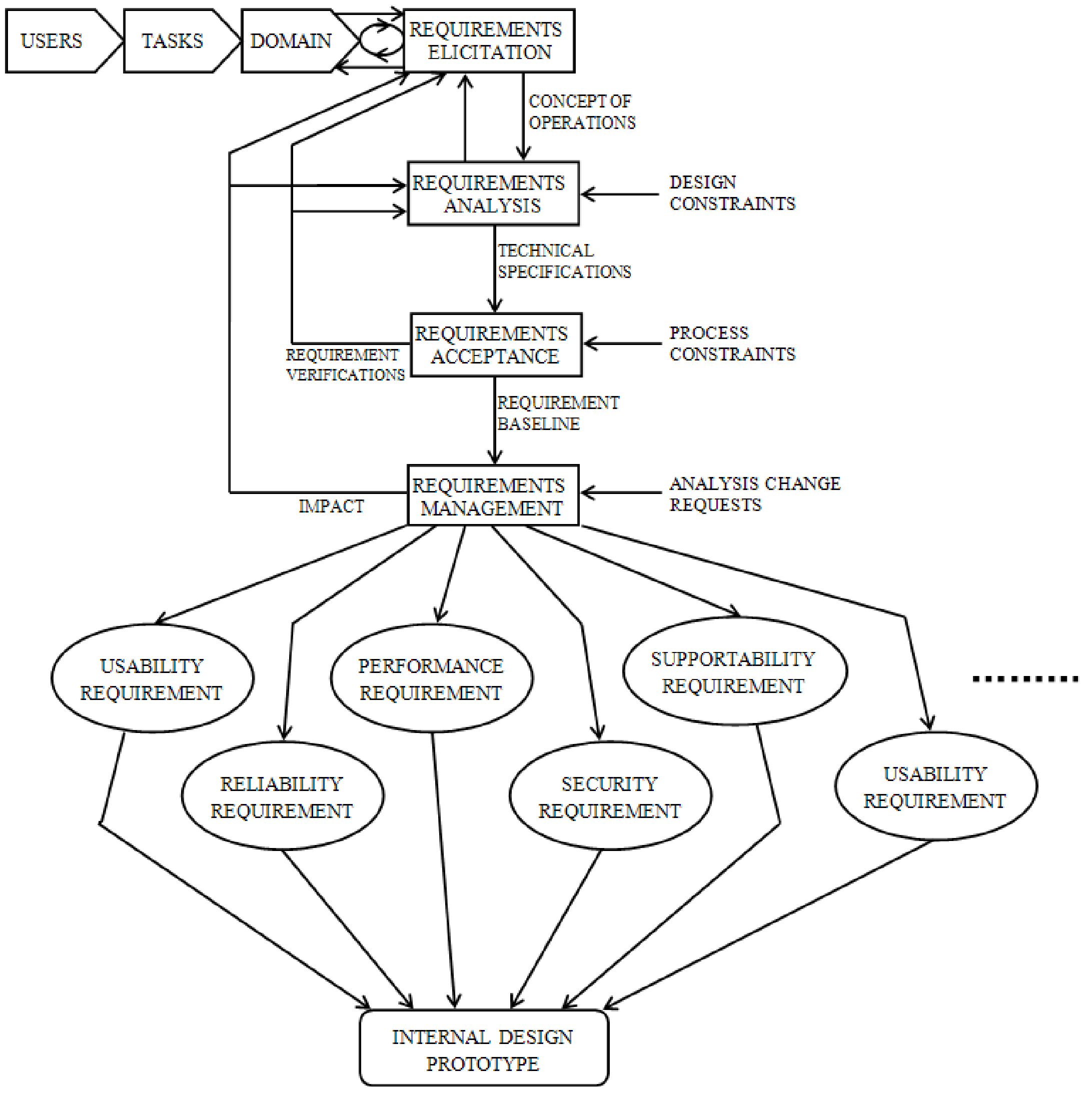

Figure 2: The Proposed Model to Define and Describe NFR Strategies

VI. CONCLUSION AND FUTURE WORK

Non-functional properties must be considered throughout the development cycle of an application system and at different levels of abstraction. In this paper we have presented a study and a model about how software architects deal with NFRs in practice. We have focused our research questions on two major activities: defining the right NFRs for the appropriate information system and the strategies to imply them in software architectural design at the early phases of development.

We want to mention that the NFRs we are dealing with are mostly elicited by the software architects, and hence the NFRs are mostly reported as technical NFRs. Software architects may be happy with those technical NFRs but we feel that the non-technical NFRs are as relevant as technical NFRs from the users' point of view. So we have planned to integrate the non-technical NFRs in our work and we want to mention it as the future scope of this work.

REFERENCES

[1] Hoyer, R. W. and Hoyer, B. B. Y., "What is quality?", Quality Progress, no. 7, pp. 52-62, 2001.

[2] R.B. Svensson, M. Höst, B. Regnell, ”Managing Quality Requirements: A Systematic Review,” EUROMICRO-SEAA 2010.

[3] R.B. Svensson, T. Gorschek, B. Regnell, “Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems,” REFSQ 2009.

[4] R.B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, A. Aurum, “Quality Requirements in Practice: An Interview St-udy in Requirements Engineering for Embedded Systems,” RE 2011.

[5] L. Karlsson, A.G. Dahlstedt, B. Regnell, J.N. och Dag, A. Persson, “Requirements Engineering Challenges in Market-driven Software Development – An Interview Study with Practitioners,” Information and Software Technology 49, 2007.

[6] G. Sabaliauskaite, A. Loconsole, E. Engström, M. Unterkalmsteiner, B. Regnell, P. Runeson, T. Gorschek, R. Feldt, “Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context,” REFSQ 2010.

[7] Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Natt och Dag, J.: An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. In: Proc. 5th IEEE Int. Sypm. on Requirements Engineering. Los Alamitos USA, pp. 84-91 (2000).

[8] Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers (2000).

[9] Cleland-Huang, J., Settimi, R., BenKhadra, O.: Goal-Centric Traceability for Managing Non-Functional Requirements. In: Proc. 27th Int. Conf. on Software Engineering. Saint Louis USA, pp. 362-371 (2005).

[10] Jacobs, S.: Introducing Measurable Quality R equirements: A Case Study. In: Proc. 4th IEEE Int. Symp. on Requirements Engineering. Limerick Ireland, pp. 172-179 (1999).

[11] Olsson, T., Berntsson Svensson, R., Regnell, B.: Non-functional requirements metrics in practice – an empirical document analysis. Workshop on Measuring Requirements for Project and Product Success. Palma de Mallorca Spain (2007).

[12] A. Tang, M. Ali Babar, I. Gorton, J. Han, “A Survey of Architecture Design Rationale”. Journal of Systems and Software 79, 2006.

[13] M.A. Babar, L. Bass, I. Gorton, “Factors Influencing Industrial Practices of Software Architecture Evaluation: An Empirical Investigation,” QoSA 2007.

[14] Dewayne E. Perry and Alexander L. Wolf, "Foundations for the Study of Software Architecture", ACM SIGSOFT Software Engineering Notes, 17(4), 1992, pp. 40-52.

[15] Dvir, D., Raz, T., Shenhar, A.J.: An empirical analysis of the relationship between project planning and project success. International Journal of Project Management 21, 89–95 (2003).

[16] Fairbanks, G.: Just Enough Architecture: The Risk-Driven Model. Crosstalk (November/ December 2010).

[17] Eltjo R. Poort, Nick Martens, Inge van de Weerd, and Hans van Vliet, “How Architects See Non-Functional Requirements: Beware of Modifiability”, B. Regnell and D. Damian (Eds.): REFSQ 2012, LNCS 7195, pp. 37–51, 2012.

[18] Leite JCSP, Franco APM. A strategy for conceptual model acquisition. In: Proceedings of the first IEEE international symposium on requirements engineering, San Diego, CA. IEEE Computer Society Press, Los Alamitos, CA, 1993, pp 243–246.

[19] Mylopoulos J, Chung L, Yu E, Nixon B. Representing and using non-functional requirements: a process-oriented approach. IEEE Trans Software Eng 1992;18(6):483–497.

[20] Luiz Marcio Cysneiros, Julio Cesar Sampaio do Prado Leite and Jaime de Melo Sabat Neto, “A Framework for Integrating Non-Functional Requirements into Conceptual Models”, Requirements Eng (2001) 6:97–115, 2001 Springer-Verlag London Limited.