Non Functional Requirements: the emerging wisdom

Naina Handa
Research Scholar
Lovely Professional University, Phagwara, Kapurthala (Jalandhar)

Dr. Anil Sharma
Associate Professor
Lovely Professional University, Phagwara, Kapurthala (Jalandhar)

Abstract
In the world of software engineering, design engineering is very important. About 70-85 percent of the cost in project reworks is spent on finding and fixing requirement errors. Requirement engineering finds the Non-Functional Requirements (NFRs) handling particularly difficult. Neglecting NFRs leads to the most expensive and difficult errors that can only be corrected once the system is complete. Fast detection and NFR recognition increase the project's chances of success. The present paper is an effort to compare the various Machine Learning Techniques used to extract and classify NFRs. It also brings forth the gaps for future considerations in literature.

Keywords—Requirement Engineering, Non-Functional Requirements, Machine Learning Techniques

Introduction
The elicitation of requirements is a very crucial step in the design process of requirements. The performance of each project depends purely on project specifications. It is possible to divide the requirements into Functional requirements (FRs) and Non Functional requirements (NFRs) [1]. FRs addresses features and functions of the system directly. FRs are specific specifications as these are specifically specified by the customer or application. In fact, NFRs can often be interdependent, which can help or hinder the accomplishment of other NFRs at specific software functions when attempting to achieve one NFR [2]. Such an association produces a large network of interdependencies and tradeoffs between NFRs that is not easy to trace or estimate. Elicitation of NFRs is very important in the early stages. Elicitation is heavily based on brainstorming, checklists and template [3]. Early elicitation of NFRs is critical in early stages. Analysis and execution of NFRs from various sources, such as SRS, instructions, brief notes etc., is a difficult process. Elicitation of NFRs is the major issue in software development framework. Investigators have dedicated substantial energies to NFRs [4]. The management of non-functional requirements is ill described in conventional and agile methodology. Consumer or customer consider what they want the device to do naturally and don't care about the money, portability, reliability, protection or efficiency they require. Nonetheless, NFRs are very difficult to manage and should be recognized in development because they impact database choice, programming language and operating system etc [5].
NFRs define parameters that help to find the system's behaviors. Such specifications have to be differentiated by the main Functional Specifications (FRs) that describe functions and device specific behavior. There are some NFR quality attributes that need to be measured before starting to work on the system like Risk analysis, Durability, Stability and Functionality [6].

Non Functional Requirements ignored

There are numerous examples of mission loss in literature such as the London Ambulance System (LAS), Mars Climate Orbiter, Therac 25, and Siemens.

In 1992, The London Ambulance Service had launched a new computer based dispatch program to streamline the network that dispatched ambulance to respond to public and emergency calls. The new system proved extremely inefficient, which significantly increased emergency response time. This failed completely soon after its launch, and LAS returned to the previous manual system. The system's failure was primarily due to a failure to recognize "physical and operational considerations" in system design [7].

Mars Climate Orbiter is another example of failure due to NFRs. It was one of two NASA spacecraft in the '98 Mars Surveyor program. Because of the problem of program "interoperability" the mission failed. During its flight, the craft drifted off course and into a much lower orbit than expected, and was killed by atmospheric friction. Back on Earth, a software error caused the metric / imperial mix-up that damaged the craft. The whole problem was of interoperability, which is a significant attribute of NFRs [8].

Siemens is also an example of project failure as in 2004, Siemens released a "security" warning that some of its cell phones may have a software problem which could cause them to emit loud noise, potentially resulting in a loss of hearing for the customer. The malfunction occurs only if, while the handset is in use, the battery runs down to the point that the phone automatically disconnects the call and starts to shut down [9].

So it is very important to be considered the NFRs during initial phases of software life cycle. The NFR has various definitions in the literature and the industry. NFR is an umbrella term and it consists of different aspects like performance, usability, efficiency, portability, extendibility and security etc.

Literature Review

Too et al. proposed eight measures outlined about quality attributes. The authors have focused on the questions that what is the finest stage to define specifications for quality attributes [10]. The different researchers have focused on different NFRS in various domains.

Saratha and Uma addressed the Decision Support System Life Cycle and outlined its significance for collaborative software systems during an organization's decision making process in a specific technology domain. For this reason the list of NFRs are divided into three phases like development, Operation and Management of Decision Support System [11].
Khan et al. explored the significance, application and overall effects of the NFRs on software architecture were illustrated. Writer suggested a method that provides support for decision making in a software development environment for the implementation of NFR decision model. In doing this, the productivity of developers will be improved in reusing design decisions [12].

Wang et al. discussed on NFRs how difficult it has been for engineers for several years, although different methods and strategies have been suggested for improving elicitation, reporting and testing for a long time. Authors indicated that learning more about these problems will be beneficial for both parties in their daily routine practice, i.e. practitioners and researchers. Authors conducted an empirical study which was based on 13 interviews with architects [13].

Sachdeva and Chung explored that online banking has shown value for growth and change worldwide and for exploiting companies, culture and individuals. NFRs are as essential as FRs and, in the initial phase, NFRs should be defined. Many of the software projects fail because the NFRs are not considered. Online banking system is routinely critical to NFRs such as consistency, accessibility, security and performance. Authors set a framework for the performance of surveys and send consumers and calculated findings to 122 online banking clients [14].

Conclusion

This article provides a comprehensive review of methods for forecasting NFR. After literature survey, it was found that neglecting NFRs leads to the most expensive and difficult errors that can only be corrected once the system is complete. Extraction and classification of NFRs is very important Requirement Engineering issue.

Different authors have done automating the extraction of NFRs using different machine learning algorithms. It is reported in literature that the NFRs are the big issue in almost every domain like banking, website development and Decision Support System to mention a few.
References


