How do Software Architects Consider Non-functional Requirements: An Exploratory Study

David Ameller[1], Claudia Ayala[1], Jordi Cabot[2], Xavier Franch[1]

[1]Software Engineering for Information Systems Group
Universitat Politècnica de Catalunya (GESSI-UPC)
Barcelona, Spain
{dameller, cayala, franch}@essi.upc.edu

[2]AtlanMod
INRIA - École des Mines de Nantes
Nantes, France
jordi.cabot@inria.fr

***Abstract*—Dealing with non-functional requirements (NFRs) has posed a challenge onto software engineers for many years. Over the years, many methods and techniques have been proposed to improve their elicitation, documentation, and validation. Knowing more about the state of the practice on these topics may benefit both practitioners' and researchers' daily work. A few empirical studies have been conducted in the past, but none under the perspective of software architects, in spite of the great influence that NFRs have on daily architects' practices. This paper presents some of the findings of an empirical study based on 13 interviews with software architects. It addresses questions such as: who decides the NFRs, what types of NFRs matter to architects, how are NFRs documented, and how are NFRs validated. The results are contextualized with existing previous work.**

***Keywords*-Non-functional Requirement; NFR; Quality Requirement; Software Architect; Architectural Decision; Empirical Study**

I. INTRODUCTION

Non-functional requirements (NFRs) express desired qualities of the system to be developed. They refer both to observable qualities such as system performance, availability and dependability, and also to internal characteristics concerning, e.g., maintainability and portability. Other authors use different names, remarkably "quality requirement", as a synonymous of NFR, being the diversity of terminology and meaning well-known by the community [1].

Over the years, a common claim made by software engineers is that it is not feasible to produce a software system that meets stakeholders' needs without taking NFRs into account. As a result, software development projects currently invest a lot into satisfying NFRs [2]. But still it seems to be a lopsided emphasis in the functionality of the system, even though the functionality is not useful or usable when NFRs do not hold [3].

NFRs affect different activities and roles related to the software development process. One of the strongest links is with software architecture, especially architectural decision-making: NFRs often influence the system architecture more than functional requirements do [4]. For instance, Zhu and Gorton state that "the rationale behind each architecture decision is mostly about achieving certain NFRs" [5]; Chung and Leite claim that "[NFRs] play a critical role during system development, serving as selection criteria for choosing among myriads of alternative designs and ultimate implementations" [3]; and Ozkaya et al. say that "business goals and their associated quality attribute requirements strongly influence a system's architecture" [6].

These works provide little direct evidence from real case studies to support the statements. Both requirements engineers [7] and software architects [8] demand field work to sustain or dismiss that "much of a software architect's life is spent designing software systems to meet a set of quality attribute requirements" [9].

Under these circumstances, we decided to design and run an exploratory study around the research question:

> How do software architects deal with non-functional requirements in practice?

The study was conducted over our local network of software architects. Based on the analysis of the answers, we were able to draw some observations about the use and impact of NFRs in industrial practice, align them with the results of previous empirical studies, and discuss possible actions that could eventually help to improve the state of practice in the field.

The rest of the paper is structured as follows. In Section II, an overview of existing empirical studies on the management of NFRs is provided. In Section III, the details of our own study, based on semi-structured interviews, are given. In Section IV, the most relevant observations gathered from the interviewees are enumerated. In Section V, these observations are discussed and aligned with the studies cited in Section II. Finally, Section VI provides some conclusions and future work.

II. EMPIRICAL STUDIES ON NFRS: AN OVERVIEW

In spite of their acknowledged importance, not so many empirical studies centred on NFRs are available, see Table I for a summary. A recent systematic literature review conducted by Svensson et al. [7] found no more than 18 empirical research studies centred on investigating the benefits and limitations on methods around NFRs for five identified areas: elicitation, dependencies, level of quantification, cost estimation, and prioritization. Some findings that are relevant to our aim were: there is no clear view on how to elicit NFRs; quantification of NFRs depends on the market and cost value; different stakeholders may have different views on the importance of NFR types. The need to increase the number and quality of studies on NFRs was pointed out as a key finding of the review.

978-1-4673-2785-5/12/$31.00 © 2012 IEEE

RE 2012, Chicago, Illinois, USA

TABLE I. SUMMARY OF EMPIRICAL STUDIES

Ref.	Subject of research	Type of analysis	Companies	Population
[7]	Elicitation; dependencies; expression; cost estimation; prioritization	Systematic Literature Review	Not specified	1.560 candidate studies, 18 selected studies
[10]	Importance of NFR types; dependencies; expression; satisfaction	Interviews	5 companies	5 project leaders, 5 product managers
[11]	Prioritization	Interviews	11 companies	11 project leaders, 11 product managers
[14]	NFR in general. Elicitation, documentation, test and management in particular	Interviews	2 companies	14 (different roles)
[15]	NFR importance	e-survey	25 companies	6 product managers, 14 project leaders, 11 programmers
[16]	NFR importance	e-survey	Not specified	162 users, 110 managers, 46 developers
[17]	NFRs in OSS adoption	Questionnaire	15 companies	15 developers or project leaders
[18]	Architecture design rationale	Questionnaire	Not specified	81 software architects
[19]	Architecture design documentation and validation	Structured group discussion	10 companies	10 software architects
Ours	Management of NFRs by architects	Interviews	12 companies	13 software architects

The authors of this systematic review themselves have conducted several empirical studies on the topic. In [10, 11], they focused on the analysis of practices on companies that produce market-driven embedded systems. Svensson et al. targeted several aspects on NFRs in [10], whilst in [11] they focused on issues related to requirements prioritization. The findings of this last paper suggest that there seems to be a lack of knowledge about managing NFRs in these companies; the authors hypothesise that this could be related to the lower importance given to them with respect to functional requirements (this is a recurrent argument in several studies). [10] reports a different perception of some NFR aspects depending on the role of the interviewee (e.g., project managers ranked performance as the most important quality aspect, whilst project leaders ranked usability first), which 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., [12][13].

Borg et al. studied in depth two case studies in two Swedish companies [14]. They interviewed 7 professionals for each case. They reported some common findings in both companies (e.g., vagueness of NFRs and difficulty to test), but also some differences, remarkably in the provenance of requirements, which was different in both cases due to contextual factors. The main conclusion of their study is that although both organizations were aware of the importance of NFRs, still their main focus was on functional requirements. The authors made the hypothesis that methods and tools supporting NFRs throughout the entire development process would be the best way to fight against this situation.

Several works focused on the importance of NFR types. In [15], an e-survey with 31 valid responses was conducted with the purpose of analysing the importance of the different types of NFRs depending on factors like type and size of project, role of the observer and application domain. Concerning role, they checked that the same three types were identified by the three analysed roles, although the importance of the types could vary. Similar research questions were explored in [16] also with an e-survey with 318 responses in this case.

In [17], Anh et al. explored several issue related to OSS adoption projects. One of the research questions was about the degree of satisfaction of NFRs by selected OSS components. The authors explored different types of NFRs and showed that performance and reliability are the two types considered most important by interviewees, and that this last type is the worst fulfilled by the components.

Although we have focused on studies centred on NFRs, we can find further evidences in other studies on related topics. For instance, Tang et al. work on architecture design rationale [18] provides evidence that our subject of research 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, the paper does not further discuss the relationship of software architecture and NFRs.

In the same field, Ali Babar et al. [19] reported observations about documentation and validation of software architectures. Participants declared that having a good understanding of the types and levels of required quality attributes is a vital factor as the types of attributes to be evaluated usually have significant influence on the choice of methods and practices.

Compared to these reported empirical studies on NFRs, the main value of ours is focusing on the relation between NFRs and the software architect role. In none of the previous studies this relationship was the real subject of study and thus available evidence is anecdotal, which makes our own study appealing, especially considering the claims that the software architect role is one of the most affected by NFRs. As we discuss in the next sections, we believe that our study brings some new interesting observations to the field. In addition, since available empirical studies are not many, having a new one that may provide further evidence in topics already explored may also be considered valuable.

III. THE SURVEY

A. Research Method

We carried out an exploratory study using a qualitative research approach [20]. Qualitative research is especially indicated when the purpose is to explore the subject of interest with the aim of improving the knowledge available. The general goal of investigating how software architects deal with NFRs was decomposed into several research

questions shown in Table II. Although the focus is on NFR-related issues, we added a preliminary research question about the responsibilities that software architects have assigned in their organizations to help understanding and interpreting the results. The other research questions focus on the perspective of the software architect on elicitation, documentation, validation and tool support, as well as terminology issues and the importance of NFR types.

TABLE II. RESEARCH QUESTIONS OF OUR STUDY

RQ1	What is the role of the software architect?
RQ2	Are there terminological confusions on NFRs?
RQ3	What types of NFRs are relevant to software architects?
RQ4	How are NFRs elicited?
RQ5	How are NFRs documented?
RQ6	How are NFRs validated?
RQ7	What type of tool support for NFRs is used?

We used semi-structured interviews for gathering information about the pre-established topics, but at the same this allows to gain deeper knowledge when required. The interview guide was carefully designed following the guidelines stated by Oates [21]. In general, the guide focused on a single software development project in which the respondents participated as architects. The interview guide used in the study is available in www.essi.upc.edu/~gessi/papers/RE12-appendix.pdf.

B. Sampling

The target population of the study (see Table III) was professionals that covered the role of architect in at least one project in the organization. Under McBride's perspective [22], a software architect is the person who makes design and technological decisions in a software development project. It is important to remark, though, that we didn't provide this or any other definition to interviewees, on the contrary RQ1 was precisely intended to find out the view that they had on software architect's responsibilities.

Participating organizations were chosen from our industrial collaboration network. We sent an invitation letter to 21 software-intensive organizations located in Spain and asked for their willingness to participate in the study. We finally recruited 12 organizations covering a varied spectrum of business areas and application domains. At one of these organizations, we were able to interview two software architects, bringing the total number of interviews to 13. The respondents held different positions in the organizations and were in charge of architectural tasks in at least the project they based their answers on. Most respondents had an education background related to computer science (with just two cases of academic background related to telecommunications and industrial engineering). 11 of the respondents had a bachelor's degree.

The selected projects themselves were also diverse in terms of functionality, size and involvement of the project staff. In some projects the team was involved just on the development tasks while in other projects the team was involved also in maintenance activities.

Although all organizations were based on Spain, some of the projects involved clients from abroad.

C. Data Collection and Analysis

Interviews were conducted face-to-face by the two first authors in the respondents' mother tongue. Each interview took about one hour and was audio-taped and prepared for analysis through the manual transcription of the audio records into text documents (made by an external company and reviewed by the researchers). Data analysis was conducted in a series of steps (based on [20]). First, the two first authors coded the data independently, using the interview transcripts and individual notes taken during the interviews. Both researchers used the tabulation technique [20] to analyze the answers of each question of the interview guide. This made it possible to get an overview of the responses and ease the process of categories generation. Depending on the granularity of the questions, some of them got a higher number of categories. NVivo Software [23] was used to support this process. Once the two researchers processed the answers, we compared our results. Most of the categories generated by the two researchers were semantically similar, but some others needed further discussion. Thus, the whole team held several meetings to analyze and discuss the categories and the evidence. Whenever we had a disagreement, we discussed the issues until we reached an agreement. As a result, some categories were split, modified, discarded or added to ensure that all answers were well-represented. It was a thorough process and some meetings lasted about 3 hours until agreement was reached. Finally, for displaying the results shown in this paper, we used the counting technique [25] to enable the reader to "see" the findings by counting frequency of occurrences, or recurrent categories of events (see Table IV). Our interpretation of the results is tackled in section V.

D. Limitations of the Study

Like all software engineering empirical studies, ours faces certain validity threats. This section discusses them in terms of construct, internal and external validity as well as reliability, as proposed by Yin [24] and also emphasizes the mitigation actions used.

Construct validity. This aspect of validity reflects to what extent the operational measures really represent what is investigated according to the research questions [24]. This study was supported by 2 main principles: rigorous planning of the study according to Oates [21], and establishment of protocols for data collection and data analysis. Our protocol included specific mitigation actions for evaluation apprehension by ensuring the confidentiality of the interviews and also by emphasizing the exploratory nature of the study. In addition, the interview guide used as an instrument to gather data, was piloted with 2 academic and 2 industrial people in order to improve its understandability. As a result, some changes were done to enhance the elicitation process (e.g., we added a glossary to homogenize key terms that could cause some confusion).

Internal validity. It refers to the confidence that we can place in the cause and effect relationship in a study [24]. We took relevant decisions for approaching a further understanding of the approached research questions. One of the main relevant decisions was to focus the questions of the

interview guide on a single software development project. Considering a single project instead of a general perception of the architects' rationale allows for better interpretation and assessment of contextual information [25]. It would otherwise have been very difficult to interpret certain decisions or influential factors related to the nature of the projects. We are aware that some possible biases may be related to this strategy, for instance the fact that some time passed since the project was completed, so it could be difficult for the respondents to remember some project details. To reduce the possible side effects of this, we sent the interview guide in advance to the respondents so they could become familiar with the topic, and asked them to choose the project beforehand. Thus, when performing the study, we rarely experienced respondents having difficulty in remembering project details. Another factor raised was that the projects were selected by the participants. They may have selected the most successful project to base their answers on, although we asked them to use the most familiar one. To mitigate this, we explained that our study was not focused on analysing "best practices" but on learning "how things are done." There is always the possibility that the respondent forgets something or does not explicitly state it when s/he is asked about it [26]. To reduce this issue, we approached two strategies: 1) we discussed some potential topics that might be omitted by the respondents, and paid particular attention to them during the interviews in order to ask for clarifications if necessary; 2) once the interviews were transcribed, the documents were validated by the respondents, so they had the chance to add or modify any comment.

We tried to be rigorous with respect to the data analysis strategy, and put forward several mitigation strategies. First, recording all interviews (and later on transcribing them) contributed to a better understanding and assessment of the data gathered. Second, to reduce the potential researcher bias, two different researchers assessed the data individually and generated their own categories. Then, the generated categories were analysed, discussed and reviewed by all researchers of the team to ensure their accuracy, understanding and agreement. Categories were also checked with respect to the data gathered to confirm that none of the categories refuted any of the conclusions.

External validity. It is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case [24]. As our study was exploratory, we do not attempt to make universal generalizations. Thus our observations should be interpreted not as a universal view of the field status but as a starting point for a universal discussion and analysis [27].

Moreover, we did not randomly select the organizations that participated in the study but got them from our industrial collaboration network. However, we tried to strengthen the external validity by having no control over the projects chosen by the respondents. It is important to mention that most of the participating companies were small or medium-sized, in addition, most of the studied projects dealt with non-critical domains (except for D and I, which dealt with aerospace and banking, respectively). We are aware that both factors may have an impact on how NFRs are dealt with, and so we highlight that our findings should be considered with caution.

TABLE III. OVERVIEW OF THE ORGANIZATIONS

Business Area (*)	Main Domain	Respon-dent Id	Project Description	Project Staff Size	Team Involvement	Duration (&)
SCC	Lottery management	A	Web application for managing transactions over mobile phones	15	Software development	6
ITD	Management of academic activities and IT resources	B	Web application for managing the activities of organization members	3	Software development and maintenance	48
SCC	Information systems	C	System for the management and logistics of a growing fast-food chain	5	Software development and maintenance	120
SCC	Aerospace information systems	D	Geographic information system to manage aerospace launch bases	≈10	Software development and maintenance	180
SCC	Information systems	E	Application to manage the processes and documents of a public-sector body	6	Software development	30
SCC	Web information systems	F	E-commerce system for a company selling motorcycle items	5	Software development and maintenance	12
SCC	Geographic information systems	G	Web system to support shipping logistics	1	Software development	3
SCC	Web information systems	H	Web system for personal data management	≈20	Software development	36
SCC	Information systems for document digitization	I	System to manage accounting activities at a bank	8	Software development	18
SH	Support systems for insurance companies	J	Integral system to support insurance company tasks	50	Software development	30
ITD	Information systems for staff management and interactions	K	System to manage staff research activities	8	Software development	36
ITD	IT support for a university department	L1	Web application to manage students and teaching activities	5	Software development and maintenance	144
		L2	Web collaboration system	8	Software development and maintenance	5

(*). SCC: Software Consultancy Company that performs software development tasks for different clients as its primary business; ITD: IT department in public or tertiary organizations that usually perform or outsource some software development tasks for covering the internal demands of the organization; SH: Software house that develops and commercializes specific proprietary solutions.

(&) It states the number of months required to perform the tasks in the team involvement column.

TABLE IV. SUMMARY OF RESPONSES

Research Question	Observations
RQ1. Architect role	– 13 interviewees performed the tasks assigned to "software architects" in the project based on their experience or knowledge rather than their possible skills as architects – 0 interviewees held a "software architect" position at the company – 12 interviewees played other roles in the project in addition to the role of software architect, specifically: project manager (3), developer (5), and project manager and developer (4)
RQ2. NFR terminology	– Confusion was reported around the terminology for designating NFR types
RQ3. NFR type ranking	– 49 references were made to technical NFRs (see Fig. 1)
	– 33 references were made to non-technical NFRs (see Fig. 2)
RQ4. NFR elicitation	– In 10 projects, the NFRs were elicited solely by the architect – In 3 projects, the NFRs were elicited by the client with the participation of the architect
	– 13 architects considered elicitation as a gradual process
RQ5. NFR documentation	– 9 architects did not document the NFRs at all – 4 architects documented the NFRs: 3 used templates (1 only for initial NFRs), 1 used plain text (only for initial NFRs)
RQ6. NFR validation	– 11 architects claimed that the NFRs had been met by the end of the project – 2 architects did not claim that all NFRs had been met by the end of the project
	– 1 architect validated three types of NFRs (reliability, efficiency, and accuracy) – 3 architects validated two types of NFRs (efficiency and accuracy; efficiency and usability; efficiency and reliability) – 4 architects validated one type of NFR (efficiency twice; accuracy; usability) – 1 architect did not validate any NFRs at all – 4 architects did not provide details on this point
RQ7. Tool support for NFRs	– Architects did not use any specific tool support for NFR management

Reliability. This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers. In order to strengthen this aspect we considered the validity of the study from the very beginning. So, as stated in the previous paragraphs, we put forward several strategies. In addition, we maintained a detailed protocol, the collected data and obtained results were reviewed by the participants; we have spent sufficient time with the study, and gave sufficient concern to the analysis of all responses.

IV. OBSERVATIONS

We present next the most relevant observations resulting from the analysis of the interviewees' responses, summarized in Table IV. We include quotations from the interviews stating respondents' ID in bold and enclosed in parenthesis.

A. RQ1: What is the Role of the Software Architect?

The analysis of the interviewees' responses in our study shows that at their companies, the role of architect did not exist as a job position as such. Given this situation, we tried to understand more in depth how architects were nominated for this role and how the boundaries of this role were set.

How were software architects nominated? The nomination of the respondents as software architects was made according to the nature of the project. In other words, it was not based on the usual architects' skills defined in the literature [22], but on technical knowledge ("[The architect] is whoever knows the technologies used in the development best" **(E)**) or experience ("Decisions affecting the whole system are made by the most experienced people" **(H)**).

How was their role scoped? Respondents found it difficult to define the exact nature of their work as an architect since it overlapped with other activities they performed in the project, primarily, project management (7 respondents) and development (9). Some even played two other roles apart from architect. Only one respondent said that the only role he played in the project was that of architect.

B. RQ2: Are there Terminological Confusions on NFRs?

In our interviews we encountered certain communication problems concerning the meanings of words, especially with regard to the definition of types of NFRs. In fact, we found two related problems: the problem of meaning itself, and the problem of translating English terms into another language, Spanish in our case. The main problematic situations were[1]:

- Inability to interpret some particular term, e.g., "availability", "accuracy", and "sustainability", requiring additional explanations from the interviewers. **(E, F, G, I)**
- Use of a term that could lead to real confusion, e.g., some said "ergonomic", "comfortable," or "friendly," when they meant "usable" (in the context of "usability"). **(B, E)**
- Use of a term with an incorrect definition. We found a serious confusion in the answer, e.g., "Maintainability is very important, because when something is working, we can't make changes" **(D)**.

C. RQ3: What Types of NFRs are Relevant to Software Architects?

We asked the respondents what were the NFR types that they took into account when making architectural decisions. We consolidated their answers, e.g., to reconcile different names for the same concept (see the terminology problem above) using the ISO/IEC 9126-1 quality standard [28] as unifying framework. Some respondents had problems to directly answer the question. In those cases, we provided them with a list of 15 terms that was consolidated when piloting the survey design and clarified their meaning. Fig. 1 shows the result of this part of the interview.

[1] In this subsection, we use English terms for the discussion but the problems appeared in their Spanish use.

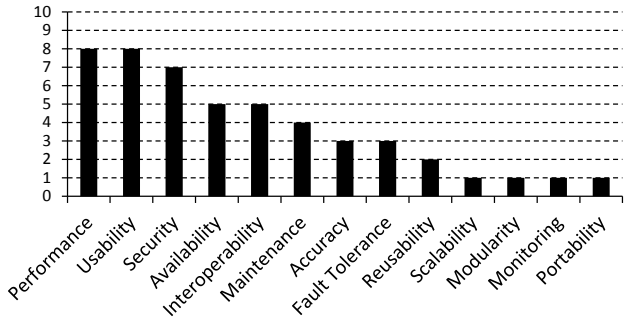

Figure 1. Importance of NFR types.

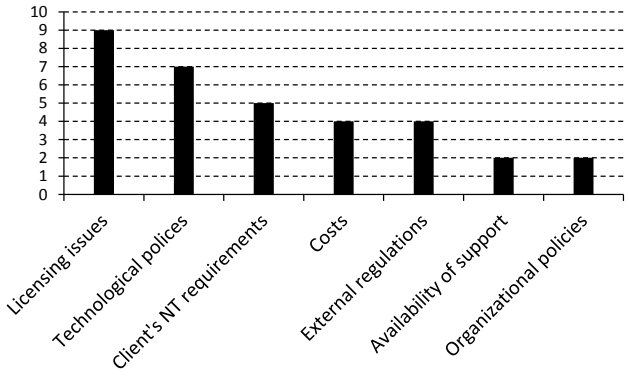

Figure 2. Non-technical NFRs.

If we observe the bar chart, we may see a graduation of the mentioned types. This aligns with the information given by the architects that considered some types as common sense characteristics, e.g., "I consider Performance and Security as default requirements of any project" **(B)**, "I would never think on a system that it is not Secure" **(I)**. Apart from these dominant types, we found other situations:

- NFRs that were considered because they represented an explicit need of the client, e.g., "one of the contractual requirements was that the system could interoperate with other systems that were already deployed in the client's environment" **(D)**.
- NFRs that were particularly important for the development team, e.g., "We were the ones that would maintain the system, so, it was important for us to ensure its maintainability" **(B)**.
- Last, four of the respondents mentioned that some NFRs were not important to them because they rely on the technologies and the underling platform, e.g., "We didn't thought about the security of the documents because it is done by the management system of SharePoint" **(E)**. The perception was that the maturity level of many technological solutions was enough to ensure the satisfaction of NFRs.

Moreover, about 40% of the NFRs considered by respondents in their projects were non-technical [29], i.e. referring to issues not directly related to the quality of the product itself but to some contextual information. In fact, some respondents explicitly mentioned that some types of non-technical NFRs took precedence over all others ("Money rules and everything has to be adapted to it" **(J)**).

The types of non-technical NFRs most often mentioned were (see Fig. 2): licensing issues, 9 times ("the client's organization limited the type of OSS licenses to be used in the software solution" **(J)**); technological constraints, 7 ("we prefer to use technologies we have already mastered" **(L1)**; "we had some limitations from the client, e.g., architecture based on OSS and Java" **(H)**); organizational issues, 5 ("we needed to adapt our solution to the organization's strategic vision" **(I)**); cost, 4 ("we preferred JBoss to an IBM solution because of cost constraints" **(F)**); external regulations, 4 ("as we are a public organization, we had to comply with certain public regulations and make our system accessible for people with certain disabilities" **(L1)**); availability of support, 2 ("the choice of technology was influenced by the support that Oracle offered" **(A)**); and development team policies, 2 ("we preferred to use our own human resources instead of subcontracting someone else" **(J)**).

D. RQ4: How are NFRs Elicited?

Our interviews show that in 10 out of the 13 projects considered, the software architect was the main source of the NFRs. Clients either never mentioned NFRs ("[the client] never mentioned that web pages could not take more than 2 seconds to load, but he complained about it afterwards" **(E)**) or provided only very broad indications, usually in the form of cost or efficiency constraints ("the client mentioned a basic [NFR], and we added others based on our experience" **(L2)**). The main explanation seems to be that architects consider themselves to be the real experts when it comes to defining efficiency, reliability, and other similar aspects.

Respondents **(D)**, **(H)**, and **(I)** were the only three cases with client-led NFR elicitation process. Interestingly, they were also the only cases in the study in which the interviewee was working on an outsourced project (managed by an aerospace company **(D)**, a software company **(H)**, and a bank **(I)**). Even in these cases, however, the architects played an active role in completing the definition of the NFRs ("Our client was an aerospace system department. Therefore, all the NFRs were very well defined. We also added other NFRs based on our experience" **(D)**).

All respondents agreed that deciding NFRs is a gradual and iterative process throughout the system lifecycle. A first set of NFRs were decided early in the project as a result of gaining knowledge about the client organization. E.g., "We determined first some relevant NFRs (e.g., compatibility with other systems) and then developed a prototype and analysed alternatives" **(J)**. However, the interviewees emphasized that the list of NFRs of the project could never be considered complete even after the development tasks had finished, instead, this list is under extension and negotiation during all development and maintenance phases of the project, e.g., "In relation to efficiency we had to make changes because the necessary level of service was not specified at the beginning of the project" **(K)**.

E. RQ5: How are NFRs Documented?

9 out of the 13 interviewees acknowledged that they had not documented the NFRs at all ("[functional requirements] came in UML, using conceptual models and use cases, but there was no mention of NFRs" **(H)**). In some cases, the lack of documentation was intentional ("I rarely appropriately document my projects, basically because it costs money" **(C)**). Some interviewees emphasized that documentation is only necessary if the client or the critical nature of the domain so requires.

The 4 respondents who did explicitly document their NFRs used different methods to do so:

- Volere templates **(B)** [30].
- Grouping of the NFRs using the ISO/IEC 9126 quality classification **(K)** [28].
- Ad-hoc formalization based on domain needs. ("Since we work in the field of aerospace, our NFRs had to be clearly stated and verifiable. We have special templates, and we used different techniques from other engineering disciplines, such as risk models, failure trees, etc." **(D)**).
- Simply drawing up a plain text document **(J)**.

Out of these four, two (**(J)** and **(K)**) only documented the initial NFRs ("At first, we wrote down some initial ideas for NFRs in natural language [...], but afterwards we did not keep track of any of them or of any other NFRs arising during the design process" **(K)**).

F. RQ6: How are NFRs Validated?

In our study, most of the interviewed architects (11 out of 13) claimed that all NFRs had been satisfied by the end of the project. However, when asked how they had validated them, their answers were vague. The following comment is illustrative: "compliance with some [not all] NFRs is only informally discussed with the client, since it is not easy to test". The only exception was interviewee **(D)**, who used formal techniques based on statistical analysis and simulation to check the system's reliability.

Eight interviewees performed some validation, but each one validated only one to three NFRs. Few types of NFRs were considered: efficiency ("we ran load and stress tests to evaluate performance" **(H)**); accuracy ("for each hour of coding we spent one hour testing for bugs" **(A)**); usability ("we made a prototype just to ensure client satisfaction with the interface" **(K)**); and reliability ("we have forced some errors to see what happens and control loss of data" **(J)**). Notably, one highly relevant type of NFR, security, was not mentioned by any of the respondents.

One respondent **(F)** was an extreme case of non-validation, noting: "We wait for the client to complain. He will notice when something goes wrong."

G. RQ7: What Type of Tool Support for NFRs is Used?

All the architects declared that no specific tools were used for NFR management. Taking the chance of the exploratory nature of semi-structured interviews, we asked the interviewees if they would be willing to accept some help in the form of a decision support tool to assist them in architectural decision-making. The main motivation to explore this issue is our vision on the use of NFRs in the model-driven development process as presented in [31].

We found a very strong reaction (e.g., "I do not believe in automatic things" **(B)**, or "I would not trust" **(F)**) against an automated decision-making tool from 5 of the respondents. The others were not so reluctant but expressed several concerns. 4 of them expressed their opinion that such a decision-making tool is simply too difficult to build ("it is hard for me to imagine that this can be done" **(I)**). A way to fight against this effect mentioned by 2 of the respondents was that the tool suggested alternatives instead of making final decisions ("the tool could show you possibilities that you have not envisaged" **(C)**). Also some worried about the amount of information that the architect should provide to such a tool for getting informed decisions ("all the time that I would need for thinking and introducing all the necessary information, would not pay" **(F)**). If such a tool would exist, architects would require a clear justification of decisions ("the critical point is the accuracy of the tool and the answer that it could give" **(C)**).

V. Discussion

In this section we discuss the possible answers to the proposed research questions based on the observations summarized in the previous section, and establish whenever possible links to the findings of previous studies.

A. RQ1: What is the Role of the Software Architect?

> Software architects did not exist as a differentiated role and performed other duties in the projects.

Most software engineering literature concurs that software companies have a specific position, known as the "software architect," whose mission is to design an architectural solution in a software development project by making architectural decisions that are compliant with the elicited requirements. Some authors as McBride [22] and Clemens [32] support this statement. However, our results show that the role of architect did not exist as a job position in the organizations and that their tasks were very diverse. This also concurs with other studies not specifically reporting on the software architect role but related to RE, e.g. Sadraei et al. reporting on project managers to take on RE activities [33].

On the one hand, the respondents were nominated as architects of the assessed projects mainly based on their technical knowledge. This finding aligns with the stated opinions of other professionals, e.g., "an architect should only be responsible for a single project/application and not the architect for all projects within a software company" [34].

On the other hand, it was difficult to enumerate the architect' tasks as these overlapped other roles' tasks. This fact aligns with the observation made by Tang et al. [18] who state that architects work on a variety of tasks (such as requirements analysis, tender analysis, architecture design and software design) and management responsibilities.

B. RQ2: Are there Terminological Confusions on NFRs?

> Architects did not share a common vocabulary for types of NFRs and showed some misunderstandings.

The problem of gathering data from interviewees was challenging due to the terminological discrepancies and misunderstandings about concepts related to NFRs. It is not the practitioners the (only) ones to blame, their confusion just reflected the lack of consensus that exist in the community, e.g., in the use of "performance" and "efficiency".

This problem has been also highlighted by other researchers. E.g., Anh et al. [17] reported confusion among maintainability and reliability ("OSS components are more reliable because the code is available and then it is easier to fix the bug"). Also, Svensson et al. reported that the concept of "compliance" as used by some interviewees was fairly different from the ISO/IEC 9126 standard's, e.g., some respondent said that compliance is important because "we must be compliant with the requirement document" [10]. Last, the problem of using English terminology by non-English professionals was reported also in [14] where the majority of practitioners was native Swedish speakers and had troubles when documenting the requirements in English.

C. RQ3: What Types of NFRs are Relevant to Software Architects?

> The two most important types of technical NFRs for architects were performance and usability. On the other hand, architects considered non-technical NFRs to be as relevant as technical NFRs.

If there is a topic that has been documented in existing empirical studies with respect our research questions, is the perception of the importance of NFR types. However, since these studies did not focus on the software architect role (we are just aware of [35]), it was a good opportunity to complement these findings with our observations.

The higher importance of performance and usability was also reported in two previous studies, [10] and [15], in the last case together with maintainability. In [17] performance was important, but usability was not among the most important quality attributes. It is worth to mention that it is not easy to align the results of these studies since often they use different classification for NFRs, therefore we have not tried to make an exhaustive alignment of the results.

In spite of these similarities, we have observed too that the results are still dependent on the domain (e.g., aerospace domain **(D)**, gave much importance to safety of people, whilst this type of NFR was not mentioned by the other participants). We also mentioned other differences in Section IV (e.g., the expertise of the development team and the technologies used). These facts could be a factor influencing the partially divergent results with previous studies. Similar opinions appear in other empirical works (e.g., "[NFRs] importance can vary depending on stakeholders' roles, types of project, orders of magnitude of requirements and application domains" [15]; "NFR types that are typical for traditional telecommunication systems gain more attention than others" [14]). Also, in [7] and [15] it is mentioned that the role of the stakeholders may influence on the perception of importance for the NFRs, but this difference could not be observed in this study because all our participants played the architect role. On the contrary, we found one work stating that there are NFR types that are always important, e.g., "some quality requirements (security) are always important for everyone" [11]. Similar statements were made by the architects interviewed in our study (see Section IV), but it is worth to mention that security was also mentioned as example of NFR type whose satisfaction is delegated onto the technologies used (from the architects perspective), and in consequence not considered important.

In our study we found out that non-technical NFRs are considered by the architects as important as technical NFRs. As far as we know, no other empirical study made this differentiation, even though that some of the non-technical NFRs are recurrently mentioned (e.g., cost [18]).

D. RQ4: How are NFRs Elicited?

> NFRs were mainly elicited by the architects themselves following an iterative process.

Numerous techniques (interviews, role playing, etc.) have been developed for requirements elicitation. They usually assume that the client, as the domain expert, is the main source of requirements. In fact, some respondent acknowledged that when referring to functional requirements: "[Business analyst] writes a detailed document reflecting all the [functional] requirements specified by the customer" **(A)**.

However, we couldn't corroborate this assumption in our work. The finding that NFRs were elicited by architects is one of our most relevant observations. The only empirical work covering this aspect is reported by Borg et al. [14], but with a non-conclusive result: from the two cases reported, in one it is said that requirements elicited directly from end users are very rare, whilst in the other, most of the requirements are elicited directly from customers and end users.

The iterative nature of NFR elicitation has not been explicitly stated by other studies. Some weakly related statement may be found by Doerr et al., who argue that the elicitation of NFRs, functional requirements and the architecture must be intertwined [36], which seems to imply that NFRs cannot be elicited upfront. Also the finding stated by Svensson et al. in [10] about NFR dismissal is somehow related: a total average mean 22.5% of NFRs were reported to be dismissed whilst the projects evolved.

E. RQ5: How are NFRs Documented?

> NFRs were not often documented, and even when documented, the documentation was not always precise and usually become desynchronized.

In spite of the plethora of proposals made by academics on requirements documentation, we finally may conclude that the participants of this study did not produce high-quality documentation or even no documentation at all.

It is not easy to compare our results with others, since most of existing reports on NFR documentation focus on their degree of quantification. NFRs are often described in non-measurable terms and with vague wordings [14]. Sabaliauskaite et al. reported that NFRs tend to be badly structured or vague [13]. Svensson et al. reported different situations in their case studies [10]. Remarkably, 60% of the interviewees stated that NFRs are *never*, or just *sometimes*, specified in a measurable manner. Interestingly enough, discrepancies between the two types of roles involved in the interviews arose (even if each project manager and project leader pair worked in the same project). Olsson et al. reported that about half of NFRs considered in a case study were quantified [37]. At this respect, our study reports just 2 out of 13 respondents (**(B)** and **(D)**) providing some quantification level, which is far from the 60% mentioned above.

F. RQ6: How are NFRs Validated?

> NFRs were claimed to be mostly satisfied at the end of the project although just a few types were validated.

The 85% (11 out of 13) of interviewees that claimed satisfaction of all NFRs is a high percentage, much higher than the 60% reported in [10]. One could argue that this observation we got in our study contradicts the statement by Borg et al. saying that most NFR types are difficult to test due to their nature [14] but in fact it is not the case. On the contrary, it indicates the need to distinguish between the perception of NFR satisfactibility (85%) and the real validation (8 out of 13, i.e. 61%, and not for all types of NFRs).

Three of the four types of NFRs mentioned by interviewees as validated, belong to what Borg et al. name "system characteristic types", which means NFRs directly related to the characteristics of the systems per se; they report that in their study, these system characteristics are considered properly tested most of the cases, whilst others like usability are often poorly tested [14].

One of the findings of our study that may align with previous results is the link between documentation and validation. Borg et al. that said: "when expressed in non-measurable terms testing is time-consuming or even impossible" [14]. Since we had just 2 respondents expressing the NFRs in a measurable form, this may be one of the reasons behind the low level of validation performed.

Last, Ali Babar et al. reported that participants in their survey suggested that the approach to evaluation also depends on the evaluation goals [19]. The evidence we can provide in this direction is the respondent that reported rigorous validation was **(D)**, whose project was in a critical domain.

G. RQ7: What Type of Tool Support for NFRs is Used?

> Software architects did not use any specific tool for NFR management.

This was one of the most extreme results of the survey. Even tool support as reported in [7] about dependency management (one important issue when it comes to NFRs), was missing. For sure the answer to this research question uncovers an important challenge to be addressed jointly by researchers and practitioners.

Concerning tool support for decision-making, this issue was mentioned by Ali Babar et al. in relation to some industrial cases that use tool support for generation of design option by exploiting some architectural knowledge [38]. Our observation about the type of tool practitioners may adopt aligns with the position reported by Hoorn et al. [39]: architects do not fancy proactive or automated support; instead, we share the view by Borg et al. [14] that methods and tools supporting NFRs throughout the entire development process are needed.

VI. Conclusions and Future Work

In this paper we have presented an empirical study about how software architects deal with NFRs in practice. We have focused our research questions on three activities: elicitation, documentation and validation; and on three other issues: terminology, ranking of types and tool support.

In conclusion, the presented results of this study enhance previous industrial surveys on the topic by:

- Finding previously unreported observations. We mention: NFRs were mostly elicited by architects; software architects considered non-technical NFRs as relevant as technical NFRs; software architects were happy with NFR fulfilment (independently of the poor validation performed); software architects did not use any specific tool for NFR management.
- Corroborating (totally or partially) previously found empirical evidence, even if for different technical roles: software architects performed other duties in the projects; interviewed software architects did not share a common vocabulary and had some terminological misunderstandings; NFR elicitation is iterative; NFRs were not often documented; just a few types of NFRs were validated; the two most important types of technical NFRs were performance and usability; software architects didn't want automatic NFR-based decision-making tools but accepted architect-driven tools.
- Finding results that do not align with other studies, or are related to contradictory results from former studies: software architects did not exist as a differentiated role; quantification of NFRs was poor.

As happens with all qualitative research, we have not aimed at obtaining generalizable results, as the qualitative research approach is intended to characterise and to find variation rather than similarity.

About future work, we concur with different authors (e.g. [7], [8]) about the need of conducting more empirical studies on this topic. Consolidation of results coming from qualitative studies is far more difficult than in the case of quantitative ones, but the knowledge gathered is very rich and a good input for both researchers and practices.

Another future work is the connection of the topic of this study with others. Remarkably we can mention the influence that the existence of a starting architecture may have on requirements in general and NFRs in particular (e.g. [40]).

One outcome of this study is an indication of how the methods and techniques coming from the research community have been rarely adopted by practitioners. From this perspective, the study has shown that a gap exists between both communities. Therefore, other possible, more visionary stream of future work has to do with bridging this gap. Some possible actions are enumerated below.

First, dealing with NFRs in software design demands for a cost-benefit analysis. In our study, an observation was the lack of proper documentation of NFRs in most projects. This is one of the many situations that arise because practitioners live in a high-pressure world that prevents to adopt tools that do not provide short-term benefits.

Second, the different profile of organizations calls for highly customizable NFR management. For instance, the companies in our study do not have a software architect position, while other companies acknowledge the importance of such a position. It is thus unrealistic to expect methods and techniques to be one-size-fits-all.

Third, professional software architects and academics should start to share communication channels. The use of blogs, twitters and e-zines by practitioners is still not commonplace in the academy, probably because the advantages for researchers are not evident. On the other hand, practitioners are usually reluctant to participate in empirical studies so that researchers in the end need to work with their local network which limits the number of such studies.

ACKNOWLEDGMENTS

We would like to thank all the participants of this study for their time and valuable contributions. This work has been supported by the Spanish project TIN2010-19130-C02-01.

REFERENCES

[1] M. Glinz, "On Non-Functional Requirements," RE, 2007.

[2] F. Buschmann, "Value-Focused System Quality," *IEEE Software*, 27(6), 2010.

[3] L. Chung, J. C. Leite, "On Non-Functional Requirements in Software Engineering," in *Conceptual Modeling: Foundations and Applications*, Springer-Verlag, 2009.

[4] K. Pohl, C. Rupp, *Requirements Engineering Fundamentals*, Rocky Nook Inc, 2011.

[5] L. Zhu, I. Gorton, "UML Profiles for Design Decisions and Non-Functional Requirements," SHARK, 2007.

[6] I. Ozkaya, L. Bass, R. Sangwan, R. Nord, "Making Practical Use of Quality Attribute Information," *IEEE Software*, 25(2), 2008.

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

[8] D. Falessi, P. Kruchten, G. Cantone, "Issues in Applying Empirical Software Engineering to Software Architecture," ECSA, 2007.

[9] I. Gorton, *Essential Software Architecture*, Springer-Verlag, 2011.

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

[11] R.B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, A. Aurum, "Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems," RE, 2011.

[12] 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.

[13] 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.

[14] A. Borg, A. Yong, P. Carlshamre, K. Sandahl, "The Bad Conscience of Requirements Engineering: An Investigation in Real-World Treatment of Non-Functional Requirements," SERP, 2003.

[15] J.L. de la Vara, K. Wnuk, R.B. Svensson, J. Sánchez, B. Regnell, "An Empirical Study on the Importance of Quality Requirements in Industry," SEKE, 2011.

[16] M. Haigh, "Software Quality, Non-functional Software Requirements and IT-business Alignment," *Software Quality Journal* 18, 2010.

[17] N.D. Anh, D.S. Cruzes, R. Conradi, M. Höst, X. Franch, C. Ayala, "Collaborative Resolution of Requirements Mismatches when adopting Open Source Components," REFSQ, 2012.

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

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

[20] M.B. Miles, A.M. Huberman. *Qualitative Data Analysis- An Expanded Source Book*. Second Edition. SAGE Publications, 1994.

[21] B.J. Oates, *Researching Information Systems and Computing*, Sage Publications, 2006.

[22] M. McBride, "The Software Architect," *Communications of the ACM*, 50(5), 2007.

[23] www.qsrinternational.com

[24] R.K. Yin. *Case Study Reseach: Design and Methods*. Fourth edition SAGE Publications, 2009.

[25] C. Robson, *Real World Research: A Resource for Social Scientists and Practitioner-researchers*, 2nd Edition, Blackwell Pub. Inc., 2002.

[26] C. Seaman, "Methods in Empirical Studies of Software Engineering," *IEEE Transactions on Software Engineering* 25(4), 1999.

[27] P. Runeson, "A Survey of Unit Testing Practices," *IEEE Software*, 22(4), 2006.

[28] *ISO/IEC 9126-1 Quality Standard*, 2001.

[29] J.P. Carvallo, X. Franch, C. Quer, "Managing Non-Technical Requirements in COTS Components Selection," RE, 2006.

[30] S. Robertson, J. Robertson, *Mastering the Requirements Process—Second Edition*, Addison-Wesley, 2006.

[31] D. Ameller, X. Franch, J. Cabot, "Dealing with Non-Functional Requirements in Model-Driven Development," RE, 2010.

[32] P. Clemens, "Certified Software Architects," *IEEE Software*, 27, 2010.

[33] E. Sadraei, A. Aurum, G. Beydoun, B. Paech, "A Field Study of the Requirements Engineering Practice in Australian Software Industry,", *Requirements Engineering Journal*, 12(3), 2007.

[34] J. Cahill, "The Role of a Software Architect," available at joncahill.zero41.com/2009/04/role-of-software-architect.html, 2009.

[35] D. Ameller, X. Franch, "How Do Software Architects Consider Non-Functional Requirements: A Survey," REFSQ, 2010.

[36] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, T. Suzuki, "Non-functional Requirements in Industry – Three Case Studies adopting an Experience-based NFR Method," RE, 2005.

[37] T. Olsson, R.B. Svensson, B. Regnell, "Non-functional Requirements Metrics in Practice – An Empirical Document Analysis," MeReP, 2007.

[38] M.A. Babar et al., "Introducing Tool Support for Managing Architectural Knowledge: An Experience Report," ECBS, 2008.

[39] J.F. Hoorn, R. Farenhorst, P. Lago, H. van Vliet, "The Lonesome Architect," *Journal of Systems and Software* 84(9), 2010.

[40] R. Ferrari, J.A. Miller, N.H. Madhavji, "A Controlled Experiment to assess the Impact of System Architectures on new System Requirements,". Requirements Engineering Journal 15(2), 2010.