How Architects See Non-Functional Requirements: Beware of Modifiability

Eltjo R. Poort[1], Nick Martens[2], Inge van de Weerd[2], and Hans van Vliet[3]

[1] Logica, Amstelveen, The Netherlands
eltjo.poort@logica.com
[2] Utrecht University, The Netherlands
namartens@gmail.com, i.vandeweerd@cs.uu.nl
[3] VU University, Amsterdam, The Netherlands
hans@cs.vu.nl

Abstract. This paper presents the analysis and key findings of a survey about dealing with non-functional requirements (NFRs) among architects. We find that, as long as the architect is aware of the importance of NFRs, they do not adversely affect project success, with one exception: highly business critical modifiability tends to be detrimental to project success, even when the architect is aware of it. IT projects where modifiability is perceived to have low business criticality lead to consistently high customer satisfaction. Our conclusion is that modifiability deserves more attention than it is getting now, especially because in general it is quantified and verified considerably less than other NFRs. Furthermore, IT projects that applied NFR verification techniques relatively early in development were more successful on average than IT projects that did not apply verification techniques (or applied it relatively late in development).

Keywords: Software Architecture, Requirements Management, Software Project Management, NFR, Modifiability, Empirical Software Engineering.

1 Introduction

Organizations are investing heavily in Information Technology (IT) in order to stay competitive [3]. For many of those organizations, improving IT project success rates is critical for their survival. Failure of IT projects is often linked to shortcomings in the requirements phase [12, 19]. Especially dealing with non-functional requirements[1] (NFRs), requirements that represent quality characteristics, is a promising area for improvement, because dealing with NFRs is viewed as a particularly difficult part of requirements engineering [2]. Not properly taking NFRs into account is considered to be among the most expensive and difficult of errors to correct once an information system is completed [16] and it is rated as one of the ten biggest risks in requirements engineering [11]. NFRs are widely seen as the driving force for shaping IT systems'

[1] The term "non-functional requirements" is widely disparaged, many prefer "quality attribute requirements" or "extra-functional requirements". However, because in the survey target audience the term is much better established and understood than its alternatives, we have chosen to maintain it throughout the survey and in this paper.

B. Regnell and D. Damian (Eds.): REFSQ 2012, LNCS 7195, pp. 37–51, 2012.

architectures [1, 4, 15, 17]. According to [8], "there is a unanimous consensus that non-functional requirements are important and can be critical for the success of a project".

One could say that architects are responsible for facilitating and realizing NFRs during software development; they are the population that has to "deal" with NFRs. Knowledge about how architects perceive and address NFRs can help IT organizations improve their architecting practices and project success rates. Therefore, we set up a survey among the members of the architecture community of practice in a major Dutch IT services company[2] to gather such knowledge. The survey was aimed at investigating how architects perceive the importance of NFRs, and which approaches they use to deal with them. We were also interested to see whether we could link these findings with IT project success.

1.1 Conceptual Model

The context of this study is bespoke software development in ABC, a major Dutch IT services company. More specifically, it is about IT Development Projects, defined as *a project where an IT system (application, software, infrastructure or other IT system) is designed, constructed and implemented.*

The focus of the survey is on investigating the two relationships depicted in the conceptual model, shown in Fig. 1, within the context of bespoke software development, and from the perspective of the architects. On the one hand, the more important non-functional requirements are, the greater the implied risk to IT project success if they are not fulfilled. On the other hand, several NFR approaches could help an IT project deal with NFRs. To put it another way, the assumption is that IT project success depends on the importance of the NFRs and the application of approaches for dealing with NFRs. We are interested in the following questions:

1. How do architects perceive the importance of non-functional requirements?
2. Is there a significant relationship between the perceived importance of non-functional requirements and IT project success?
3. What approaches for dealing with non-functional requirements do practitioners apply?
4. Is there a significant relationship between applying approaches for dealing with non-functional requirements and IT project success?

A complicating factor in this model is the fact that we are by necessity looking at all this through the architect's eyes. Since the measuring instrument is a survey among architects, we are not actually measuring the importance of NFRs, but rather the *architect's awareness* of their importance. Architecture is a risk driven discipline [7]. Awareness of a risk is a prerequisite to dealing with it. The more an architect is aware of the importance of a requirement and its implicit risk of not being fulfilled, the better he is able to address it. This mechanism works against the expected negative impact of NFR importance on project success; it can even completely negate it when the architect is fully successful in addressing the NFRs he is aware of.

[2] In this paper, this company will be identified as ABC.

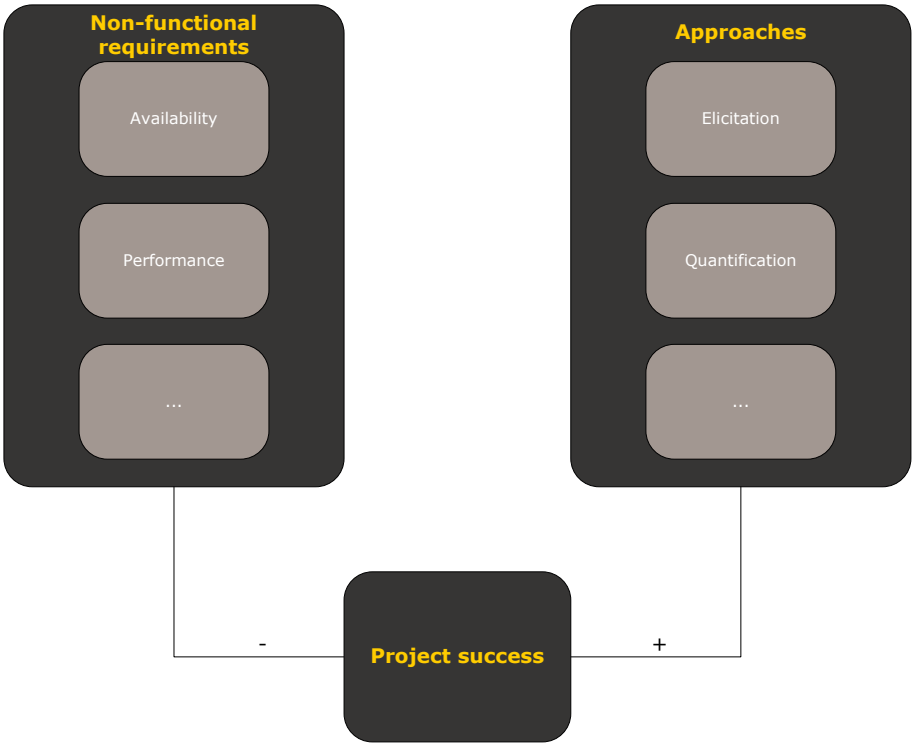

Fig. 1. Conceptual model

2 Survey Description

The core of this study is an on-line survey that was conducted in 2010 among practicing architects. In addition to the survey itself, we organized two expert workshops, consisting of a guided discussion with a select group of architecture experts in the ABC company. One workshop was held prior to the survey itself, and its prime objective was to align the survey's contents with the vocabulary and way of working within ABC. The second workshop was held after the survey, and its purpose was to enrich the initial quantitative analysis results with qualitative knowledge from practicing architects.

The invitation to participate in the survey was sent out by e-mail to around 350 members of the Netherlands (NL) Architecture Community of Practice (ACoP) of the ABC company. The ACoP consists of experienced professionals practicing architecture at various levels (business, enterprise, IT, software, and systems architecture) in project or consultancy assignments. The survey was closed after 16 days. By that time, 133 responses were collected. After elimination of duplicates (1), incomplete responses (51) and responses from respondents that indicated they had not fulfilled the role of architect on their latest project (41), 39 responses remained.

The survey consists of 23 questions divided over four sections. The first section consists of questions that are related to the general characteristics of the latest completed

project of the respondent. The second section asks the respondent to evaluate the success of his or her latest completed project from a number of perspectives. Respondents were asked to characterize their latest completed project in terms of NFRs in the third section of the survey. The fourth section evaluates the approaches deployed for managing and dealing with NFRs in their latest completed project. The survey concludes by presenting a number of statements about NFRs to the respondent. Examples of what the survey questions looked like are shown in Fig. 2.

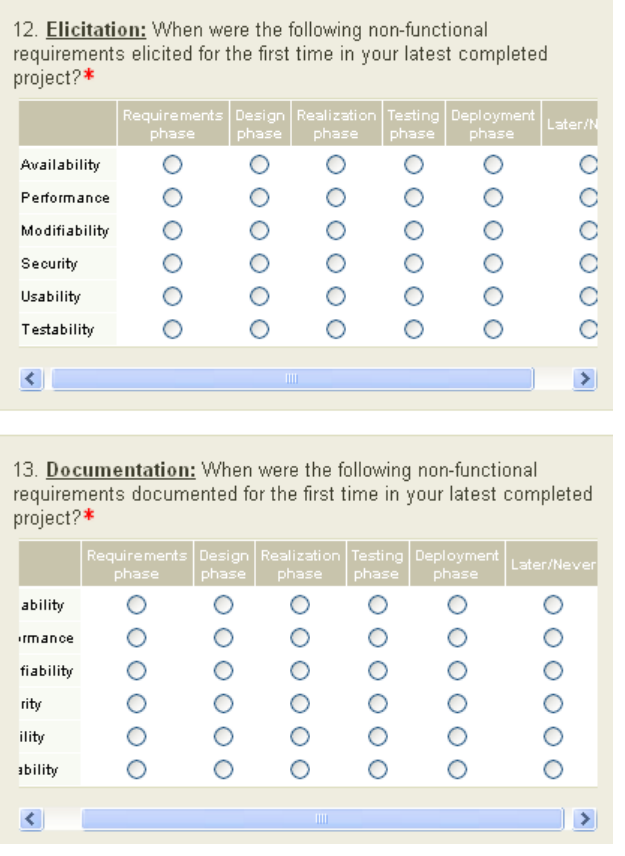

Fig. 2. Example survey questions

2.1 Constructs

Considerable time and effort was spent on translating the key concepts of the conceptual model into operationalized constructs for use in the survey. The four key concepts were *Non-Functional Requirements*, *NFR importance*, *project success* and *NFR approach*. Each of these concepts was first operationalized by looking for useful descriptions and

classifications in literature, which resulted in a draft survey. The draft survey was then the subject of an expert workshop, in which it was discussed by eight architecture experts from ABC's central technical unit (a kind of architecture board). The constructs were the main topic of the workshop discussion - especially the use of terms and models that would be commonly understood by the ABC company's architecture community. The workshop outcome led to a modified, final version of the survey.

Non-Functional Requirements The Non-Functional Requirements concept had to be made more specific. To be able to analyze the impact of different NFRs, the NFR concept had to be classified into subtypes. The problem of choosing a specific scheme to sub-classify NFRs lies in the observation that even well-known classification schemes are terminologically and categorically inconsistent with each other [4]. Many of the published classifications and definitions of NFRs have their own communities in science and practice [1]. Since a significant number of architects of ABC had been trained in the software architecture practices of the Software Engineering Institute, the six most common and important types of NFRs distinguished by those practices were used in the survey. Their basic descriptions were taken from [1], and were slightly enhanced with examples by the pre-survey expert workshop to increase understandability in the ABC architecture community context:

Availability concerns system failure and its associated consequences. A system failure occurs when the system no longer delivers a service consistent with its specification. Such a failure is observable by the system's users (either humans or other systems). Reliability and recoverability are examples that belong to this type.

Performance events (interrupts, messages, requests from users, or the passage of time) occur, and the system must respond to them. Performance is concerned with how long it takes the system to respond when an event occurs. Efficiency and throughput are examples that belong to performance.

Modifiability considers how the system can accommodate anticipated and unanticipated changes and is largely a measure of how changes can be made locally, with little ripple effect on the system at large. Adaptability, maintainability and compatibility are examples that belong to this type.

Security is a measure of the system's ability to resist unauthorized usage while still providing its services to legitimate users. An attempt to breach security is called an attack and can take a number of forms. It may be an unauthorized attempt to access data or services or to modify data, or it may be intended to deny services to legitimate users.

Usability is concerned with how easy it is for the user to accomplish a desired task and the kind of user support the system provides. It can be broken down into the following areas: learning system features, using a system efficiently, minimizing the impact of errors, adapting the system to user needs, increasing confidence and satisfaction.

Testability refers to the ease with which software can be made to demonstrate its faults through (typically execution-based) testing.

NFR Importance. How does one measure the importance of each type of NFR for a project? The experts in the 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. Furthermore, when you measure the number of requirements for each type of NFR, you are only measuring NFRs that were documented or elicited. The problem with NFRs often is that certain NFRs are *not* documented or elicited. Therefore, the suggestion of the experts was to use the concept of *business criticality*: a certain type of NFR is more important if it is relatively more critical for the system and the business of the customer. This is a concept that can be judged by the respondent in hindsight and is more valid than a simple requirement count. An NFR is considered business critical when it is vital to the customer's business. The measure in which highly business critical NFRs are fulfilled has a high impact on the system's business value, and vice versa. Respondents were asked to rate the business criticality of each of the six types of NFRs on a 5-point Likert-scale (very low, low, medium, high, very high).

Project Success. The project success construct consists of five dimensions, that are designed to reflect the interests of the three main stakeholders (cf. [6]). 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. Respondents are asked to rate the success of their latest completed project in terms of these dimensions on a 5-point Likert-scale (very unsuccessful, unsuccessful, neutral, successful, very successful). The overall project success parameter is the sum of the responses for the 5 values. Cronbach's α [5] was used as a reliability test to assess internal consistency of this construct; at $\alpha = .858$, the construct proves to be valid ($> .8$).

NFR Approach. The survey asks the respondents to indicate what approaches were applied for dealing with NFRs during their latest completed IT project. Practitioners find dealing with NFRs the most difficult part of requirements engineering [2]. 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*.

Svensson [2] and Paech [18] both provide classifications of activities aimed at dealing with NFRs. After merging these two classifications and discussing the result in the pre-survey expert workshop, the following approaches were included in the survey:

Elicitation interacting with stakeholders (customers, users) of a system to discover, reveal, articulate, and understand their requirements.

Documentation requirements are written down in order to communicate them to stakeholders (designers, developers, testers, customers).

Quantification NFRs are made explicit by giving them numbers on a measurable scale. This makes the NFRs verifiable.

Prioritization assigning priorities among the different NFRs on the basis of their relative importance.
Conflict analysis identifying the interdependencies and conflicts among the NFRs.
Verification verifying that a system fulfills requirements, e.g. by prototyping, simulation, analysis, testing or other means.

For a full operationalization of the NFR Approach construct, we not only need a classification of sub-types, but also a way to measure their usage in the projects. The simplest way to determine which of the approaches were applied would be to ask respondents using a yes/no format. However, this is not sufficient. We want to be able to distinguish between situations where the approaches were used early on in the project ("on time") and late in the project ("after the fact"). Several studies [9, 20] have pointed out that the relative costs of correcting (requirements) errors increases during the development life cycle. In line with these findings, one may expect that applying an approach later in the development life cycle is less effective; in other words, the earlier an approach for dealing with NFRs is applied, the stronger its positive impact on project success is expected to be. Therefore, respondents are asked to indicate *when* the approaches were applied during the development life cycle for each type of NFR on a 6-point Likert-scale. The Likert-scale represents five phases of a generic systems development life cycle (requirements phase, design phase, realization phase, testing phase, deployment phase) and a later/never option.

3 Analysis

In this section, we present the most interesting results of the quantitative analysis of the survey responses. The outcome of this quantitative analysis was discussed by a post-survey workshop with architecture experts in the ABC company. The results of this post-survey workshop will be presented in the Discussion section of this paper.

In Fig. 3, an overview is given of how the software architects rated the business criticalities of the NFRs.

Availability and (to a slightly lesser degree) usability are generally considered highly business critical, while modifiability and testability score relatively low. Performance and security are somewhere in the middle.

Overall, the types of NFRs are almost never unimportant: very few respondents rated the business criticality of any type of NFR as very low or low. This suggests that each type of NFR has at least some basic level of business criticality in every project. Therefore, each project involves dealing with every type of NFR at least to some degree.

Figure 4 shows how many of the 39 architects applied each of the approaches, differentiated per NFR. Again, modifiability scores low: almost all approaches are applied less for modification than for other NFRs, especially quantification and verification.

3.1 Non-Functional Requirements and Project Success

Based on the theory described earlier, the expectation is that the business criticality of NFRs is negatively correlated with IT project success, but that this effect may be

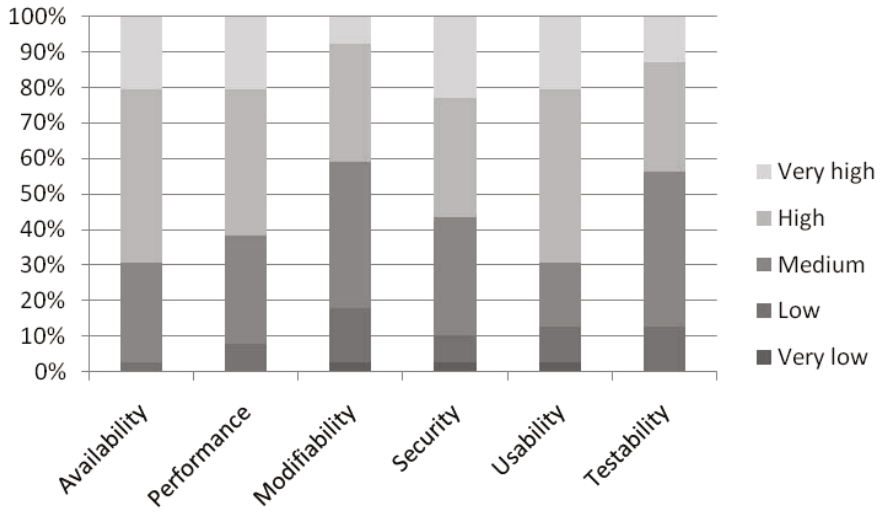

Fig. 3. Perceived business criticality of NFRs

	availability	performance	modifiability	security	usability	testability	Total
elicitation	37	37	32	37	35	34	212
documentation	32	30	25	32	25	26	170
quantification	32	29	16	30	20	21	148
prioritization	26	27	18	24	19	17	131
conflict analysis	22	24	15	20	18	16	115
verification	29	33	18	33	28	30	171
TOTAL	**178**	**180**	**124**	**176**	**145**	**144**	

Fig. 4. Application of approaches per NFR

dampened by the architect's awareness bias. For each NFR category, this hypothesis is tested using Kendall's τ (one-tailed) and the level of statistical significance is .05 (α = .05). The value of Kendall's τ ranges between -1 (perfect negative correlation) and +1 (perfect positive correlation).

A summary of the results is presented in Table 1. Statistically, we should ignore correlation coefficients where the significance $Sig. > .05$, which are indicated by "ns" (not significant) in the table. Only Modifiability shows a significant correlation between its perceived business criticality and project success. In other words, *projects where modifiability is highly business critical tend to be less successful than projects where modifiability is less important.*

Further analysis in Table 2 shows that this correlation can be attributed largely to one project success factor: customer satisfaction. This result is visualized in Fig. 5. The figure shows a remarkably consistent level of customer satisfaction for all projects

Table 1. NFRs, correlation coefficient with IT project success

Type of NFR	Kendall's τ	Sig. (1-tailed)
Availability	.086	ns
Performance	-.181	ns
Modifiability	-.257	.023
Security	.078	ns
Usability	-.102	ns
Testability	.095	ns

Table 2. IT project success factors, correlation with perceived business criticality of modifiability

Success Factor	Kendall's τ	Sig. (1-tailed)
Time	-.212	ns
Budget	-.219	ns
Efficient use of resources	-.207	ns
Customer satisfaction	-.324	.010
Solution quality	-.233	ns

where the architect judged business criticality of modifiability to be low or very low. As business criticality of modifiability grows, customer satisfaction ratings are spread over a wider range, and decrease on average.

3.2 Approaches and Project Success

The six requirements engineering approaches we consolidated from literature are expected to have a positive correlation with IT project success. For each identified approach, respondents had to indicate if it was applied and when it was applied during their latest completed project. The earlier the application of an approach in the systems development life cycle the higher the score, measured on a 6-point Likert-scale where each rating represents a project phase (requirements phase, design phase, realization phase, testing phase, deployment phase, later/never). The rationale behind this argument was described earlier. Statistical techniques are used to test the hypotheses and the results are presented in this section.

A summary of the results is presented in Table 3.

As seen from the table, only applying verification is positively correlated with IT project success.

Count		Criticality modifiability				
		Very Low	Low	Medium	High	Very high
Customer satisfaction	Very Successful			3		1
	Successful	1	6	6	6	
	Neutral			6	4	
	Unsuccessful			1	2	
	Very Unsuccessful				1	2

Fig. 5. Cross-table of business criticality of modifiability and customer satisfaction

Table 3. NFR Approaches and their correlation coefficient with IT project success

NFR Approach	Kendall's τ	Sig. (1-tailed)
Elicitation	.054	ns
Documentation	.065	ns
Quantification	.024	ns
Prioritization	.057	ns
Conflict analysis	-.128	ns
Verification	.256	.014

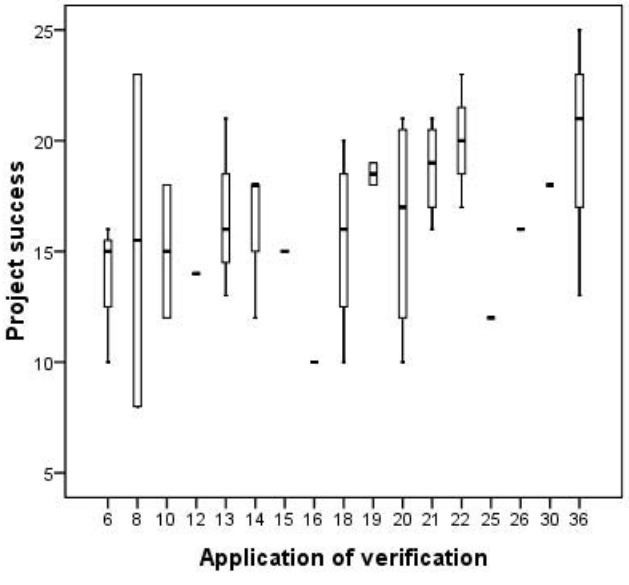

Fig. 6. Boxplot of the correlation between the application of verification and project success

The correlation between verification and project success is visualized in Fig. 6. The horizontal axis in this figure represents a score based on when verification was applied, accumulated for all NFRs listed in 2.1: the higher the score, the earlier in the project verification was applied. There is a significant positive relationship between applying verification and IT project success, $\tau = .256$, p (one-tailed) $< .05$. In other words, we find that *projects where NFRs are verified in an early stage tend to be more successful than projects where NFRs are not verified or only at a later stage in the project.*

4 Discussion and Related Work

In this section, we further discuss the results found above, and share the key contributions from the post-survey analysis expert workshop. We will also discuss threats to validity, and relate our work to additional material found in literature.

4.1 Availability Most Business Critical

In the perception of architects, on average the business criticality of availability is highest. Earlier studies found similar results. For instance, in [10] reliability was identified as the most important type of NFR in software platform development. Furthermore, in [13] reliability was ranked as the most important NFR and availability was ranked as the most important sub-characteristic for intranet applications. These studies used the six quality characteristics from the ISO/IEC 9126 standard as types of NFRs, where availability is a sub-characteristic of reliability. Furthermore, their definition of reliability is very similar to the definition of availability used in this research.

4.2 Non-Functional Requirements and Project Success

The results show that the perceived business criticality of modifiability is negatively correlated with IT project success. In other words: on average, IT projects where modifiability is seen as relatively important are significantly less successful than IT projects where modifiability is considered to be relatively unimportant. This correlation is largely due to the level of customer satisfaction.

The following three possible explanations for this phenomenon were generated by the post-survey workshop with architecture experts:

1. A high demand for modifiability might be an indication that the **customer does not know what he wants**. This means that a customer that demands high modifiability, is a customer that is more likely to change his requirements later on. A development team is trying to hit a moving target in such a situation. This explanation is in line with the leading role of customer satisfaction in the correlation.
2. Modifiability **leads to complexity**. Known techniques to realize high modifiability (such as layering, late binding and parameterizing) quickly lead to increasing complexity, with an adverse effect on budget and timescale. If this were the case, projects where modifiability is highly business critical would be expected not only to be less successful, but also larger and more prone to budget and schedule overruns. Thus, one would expect significant correlations between modifiability and project size, time and budget success factors. None of these correlations were found; in fact, some of the respondents that indicated low criticality for modifiability were working in some of the larger projects compared to other respondents. Thus, the survey yields no evidence supporting this theory.
3. Modifiability gets **too little attention**. This explanation appears to be confirmed by the relatively low scoring of modifiability in terms of perceived business criticality and application of techniques reported above. Expert workshop members experienced multiple reasons for "underappreciation" of modifiability:
   - modifiability is harder to quantify or measure, less "mathematical" than other NFRs; even though there are well known modifiability related code analysis metrics like cyclomatic complexity [14], such metrics are seen as only indirectly related to the actual modifiability business goals, and easily "cheated"

– other NFRs have a more direct effect on the project's business stakeholders (end-users, managers), while modifiability is sometimes perceived to become important only after the project is over - a dangerous view in light of the research presented here

No correlation is found between the business criticality of the other types of NFRs (availability, performance, security, usability and testability) and IT project success. This can either mean that the negative impact of NFRs is too small to be measured in a population this size, or that the dampening effect discussed before is in play: architects can only respond that NFRs are highly business critical if they are *aware* of this business criticality at the time of the survey. If an architect is aware of an NFR's business criticality at the time of creating the architecture, this awareness normally leads to addressing of the NFR in the architecture, thus reducing the risk to project success. The expert workshop produced anecdotal evidence confirming the second theory. For example, the ABC company has a project unit that is specialized in highly reliable system construction. Projects where availability is highly business critical get assigned to this unit. This leads to economies of learning and thus more successful projects.

All this leads to the following conclusion regarding the link between NFRs and project success:

As long as the architect is aware of the business criticality of NFRs, they do not adversely affect project success, with one exception: highly business critical modifiability tends to be detrimental to project success, even when the architect is aware of it.

4.3 Approaches and Project Success

The application of verification is positively correlated with IT project success. More specifically: IT projects that apply verification early in the development life cycle are significantly more successful than IT projects that apply verification late in the development life cycle. Verification was defined earlier as: verifying that a system fulfills NFRs, e.g. by prototyping, simulation, analysis, testing or other means. Although it is quite trivial that verification techniques reduce errors, there are apparently obstacles that prevent early verification of NFRs. This result indicates that practitioners should spend effort to overcome those obstacles.

It is surprising that none of the other approaches were found to have a significant effect on project success. After all, to be able to apply verification, shouldn't one at least have elicited and quantified the NFRs first? When evaluating the operationalization of the questions, some limitations come to mind. First, it might be more meaningful to measure *how* a certain approach was applied instead of measuring *when* it was applied. In the current situation, IT projects that very carefully elicited NFRs with multiple stakeholders using a formal method are not necessarily discriminated from IT projects where elicitation is informally applied in an ad-hoc fashion by a single stakeholder; moreover, the approaches are not really orthogonal with respect to the development phases. Second, the 6-point Likert-scale used is based on a general waterfall systems development life cycle and does not map very well unto iterative development methodologies. During the validation session, the experts judged that they were sufficiently

aligned with the majority of the projects carried out by ABC. However, at least one respondent had trouble answering the questions about the application of the approaches, because his projects always use iterative development. These limitations mean we have to be careful interpreting this result, beyond that it is good to have some statistical evidence that early NFR verification is correlated with successful projects in at least one company.

4.4 Threats to Validity and Opportunities for Further Research

A few important limitations of this survey have to do with generalizability. First, the context of the research is architecture, since it has such a strong link with dealing with NFRs. This was a conscious choice, but it does mean that all results are subject to the perception of the projects' architects. It would be interesting to also investigate the impact of NFRs from other perspectives and compare the results. In particular, a study that would be able to distinguish between NFRs' business criticality and the architect's awareness of that criticality might shed more light on the material.

Second, the data was collected using respondents from a single organization. A cross-organizational approach would have been preferred, but this was not feasible due to practical limitations. Strictly speaking, the results are valid only in the context of this single organization. However, the IT services company where this research was carried out has many similarities with other similar companies. Moreover, from other surveys we know that over half of the ACoP architects fulfil their roles on-site in customer organizations; so the results represent a mix of experiences in ABC and its customer base in the government, utilities, financial and other industrial sectors. Nevertheless, some results could be specific to the ABC company, and cannot be generalized without further research.

The measurement of the applied approaches was already mentioned as a limitation of this study. This could be a reason why no significant relationships were found between applying the approaches and IT project success except for verification. A study that focuses on measuring maturity of the applied approaches might be better capable to differentiate successful IT projects from unsuccessful ones. Another recommendation for future research would be to use a different kind of measurement for project success, e.g. including the actual customer and his evaluation of a project's success.

Other suggested extensions to future versions of this research are:

- extend the definition of business criticality (see Section 2.1) to the company developing the software, rather than only its customers, which might yield a more balanced view on e.g. testability
- include *Designing for NFRs* in the list of approaches; this key activity of architects is left implicit in this survey, but making it explicit may yield additional interesting results
- ask the architects *when* they became aware of the business criticality of NFRs, to validate the conclusion at the end of Section 4.2.

5 Conclusions

We set out on this survey with the goal to investigate the awareness and handling of non-functional requirements among architects, and their effect on IT project success.

The first part focused on trying to identify if certain types of NFRs have a relationship with IT project success. In other words, are there under-performing IT projects based on the types of NFRs they deal with? A significant negative relationship between the business criticality of modifiability and IT project success was found. Therefore, it can be concluded that IT projects where modifiability is relatively business critical perform significantly worse on average. Even though this result might be local to the ABC company, it provides a warning to all practitioners dealing with IT projects with a strong focus on modifiability. Aspects like quantification, verification and managing customer expectations around modifiability might require additional attention, because it seems that customer satisfaction especially is significantly lower on average in this type of IT projects.

The second part views the research question from another perspective: do approaches for dealing with NFRs have a positive influence on IT project success? From the results it can be concluded that the application of verification (starting as early as possible during the software development life cycle) has a positive influence on IT project success. In other words: IT projects that applied verification techniques relatively early in development were more successful on average, than IT projects that did not apply verification techniques (or applied it relatively late in development). As said earlier, practitioners should be aware that the long term benefits of verification outweigh the short term extra costs.

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison Wesley (2003)
2. Berntsson Svensson, R.: Managing Quality Requirements in Software Product Development. PhD thesis, Department of Computer Science, Lund University (2009)
3. Centraal Bureau voor de Statistiek. Nationale rekeningen 2006 (2007)
4. Chung, L., Nixon, B., Yu, E.S., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic (1999)
5. Cronbach, L.J.: Coefficient alpha and the internal structure of tests. Psychometrika 16(3), 297–334 (1951)
6. 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)
7. Fairbanks, G.: Just Enough Architecture: The Risk-Driven Model. Crosstalk (November/December 2010)
8. Glinz, M.: On non-functional requirements. In: 15th IEEE International Requirements Engineering Conference RE 2007, pp. 21–26. IEEE (2007)
9. Grady, R.B.: An economic release decision model: Insights into software project management. In: Proceedings of the Applications of Software Measurement Conference, Orange Park, Software Quality Engineering, pp. 227–239 (1999)

10. Johansson, E., Wesslén, A., Bratthall, L., Höst, M.: The importance of quality requirements in software platform development - a survey. In: HICSS 2001: Proceedings of the 34th Annual Hawaii International Conference on System Sciences, vol. 9, p. 9057. IEEE Computer Society, Washington, DC (2001)
11. Lawrence, B., Wiegers, K., Ebert, C.: The top risks of requirements engineering. IEEE Softw. 18(6), 62–63 (2001)
12. Leffingwell, D.: Calculating your return on investment from more effective requirements management. American Programmer 10(4), 13–16 (1997)
13. Leung, H.K.N.: Quality metrics for intranet applications. Information and Management 38(3), 137–152 (2001)
14. McCabe, T.: A complexity measure. IEEE Transactions on Software Engineering 2, 308–320 (1976)
15. Mylopoulos, J.: Goal-oriented requirements engineering, part ii. In: RE 2006: Proceedings of the 14th IEEE International Requirements Engineering Conference, IEEE Computer Society, Washington, DC (2006)
16. Mylopoulos, J., Chung, L., Nixon, B.: Representing and using nonfunctional requirements: A process-oriented approach. IEEE Trans. Softw. Eng. 18(6), 483–497 (1992)
17. Paech, B., Detroit, A., Kerkow, D., von Knethen, A.: Functional requirements, non-functional requirements, and architecture should not be separated - a position paper. In: REFSQ, Essen, Germany (September 2002)
18. Paech, B., Kerkow, D.: Non-functional requirements engineering - quality is essential. In: 10th Anniversary International Workshop on Requirements Engineering: Foundation for Software Quality (2004)
19. Sheldon, F.T., Kavi, K.M., Tausworth, R.C., Yu, J.T., Brettschneider, R., Everett, W.W.: Reliability measurement: From theory to practice. IEEE Software 9(4), 13–20 (1992)
20. Westland, J.C.: The cost of errors in software development: evidence from industry. Journal of Systems and Software 62(1), 1–9 (2002)