The impact of non-functional requirements on project success

Nick Martens
April, 2011

IKU-0255300
Summary

Organizations are investing heavily in Information Technology (IT) in order to stay competitive. At the same time, several studies have reported IT project success rates that are unsatisfactory. For many of those organizations, improving IT project success rates is critical for their survival. The failure of IT projects is oftentimes linked to shortcomings in the requirements phase. Especially dealing with non-functional requirements (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.

Knowledge about what type of IT projects are more likely to be less successful is very useful for organizations that want to improve their IT project success rates. Therefore, this quantitative research aims to evaluate if certain types of NFRs can be linked to IT project success. On the other hand, this research will evaluate whether applying specific approaches for dealing with NFRs during the IT project have a positive effect on its success.

The results show that IT projects where modifiability is highly business critical are significantly less successful than IT projects where modifiability is not highly business critical. Practitioners dealing with IT projects characterized by a focus on modifiability should be careful. Managing customer expectations might require additional attention, since the results indicate that customers of these types of projects are significantly less satisfied. Furthermore, 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). Practitioners should be aware that the long term benefits of verification most likely outweigh the short term extra costs.
Acknowledgements

I would like to take this opportunity to thank several people that helped me finish writing my master thesis. First of all, I would like to thank Inge van de Weerd and Eltjo Poort for their effort, excellent supervision and patience. During my research project I got distracted a number of times, but they were always available for useful advice. Second, I would like to thank Hans de Vreeze, Marco Stikkelorum and Edwin Essenius of Logica/Working Tomorrow for providing me the opportunity to write my thesis at Logica in the first place and for their daily supervision. Together with my fellow students they made working at Logica a lot of fun. Third, I would like to thank Slinger Jansen for the useful feedback on my work. And last but not least, I want to thank my family and friends for supporting me in every possible way.
Contents

Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . i
Acknowledgements . . . . . . . . . . . . . . . . . . . . . . . . . . ii

1 Introduction .............................................. 3
  1.1 Problem definition . . . . . . . . . . . . . . . . . . . . . . . 3
  1.2 Research questions . . . . . . . . . . . . . . . . . . . . . . . 4
  1.3 Conceptual model . . . . . . . . . . . . . . . . . . . . . . . . 5
  1.4 Relevance . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
  1.5 Thesis structure . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2 Research method ........................................ 8
  2.1 Theory-building . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  2.2 Theory-testing . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
    2.2.1 Population and respondents . . . . . . . . . . . . . . . 11
    2.2.2 Sampling procedure and data collection . . . . . . . 12
  2.3 Expert workshop . . . . . . . . . . . . . . . . . . . . . . . . . . 13

3 Related literature ....................................... 14
  3.1 Project success . . . . . . . . . . . . . . . . . . . . . . . . . . 14
  3.2 Non-functional requirements . . . . . . . . . . . . . . . . . . 16
    3.2.1 Classification . . . . . . . . . . . . . . . . . . . . . . . 17
  3.3 NFR Approaches . . . . . . . . . . . . . . . . . . . . . . . . . 23
    3.3.1 Elicitation and documentation . . . . . . . . . . . . . . 24
    3.3.2 Dependencies and conflict analysis . . . . . . . . . . . 28
    3.3.3 Prioritization . . . . . . . . . . . . . . . . . . . . . . . . 31
    3.3.4 Quantification . . . . . . . . . . . . . . . . . . . . . . . 34

4 Survey operationalization ............................. 37
  4.1 Constructs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
    4.1.1 Project success . . . . . . . . . . . . . . . . . . . . . . 38
    4.1.2 Non-functional requirements . . . . . . . . . . . . . . . 39
    4.1.3 NFR Approaches . . . . . . . . . . . . . . . . . . . . . . 40
  4.2 Expert workshop on the survey . . . . . . . . . . . . . . . . 41
  4.3 Reliability and validity . . . . . . . . . . . . . . . . . . . . . 43
4.3.1 Reliability .............................................. 43
4.3.2 Internal validity ................................. 43
4.3.3 External validity ................................. 44
4.4 Refining the hypotheses ......................... 44

5 Results ................................................. 46
5.1 Data preparation ................................. 46
5.2 Responses ........................................... 47
5.3 Descriptive statistics ......................... 47
5.4 Perception .......................................... 53
5.5 Non-functional requirements and project success .......... 56
5.6 Approaches and project success .............. 59
5.7 Comparison with other functions ........... 61
5.8 Summary ............................................. 64

6 Discussion ............................................. 65
6.1 Descriptive statistics ........................ 65
6.2 Perception .......................................... 66
6.3 Non-functional requirements and project success .......... 66
6.4 Approaches and project success .............. 67
6.5 Other limitations .................................. 68

7 Conclusion ............................................. 69
7.1 Future research ................................. 70

A Survey ................................................. 79

B Figures .................................................. 92
Chapter 1

Introduction

In this chapter, the context of the research is shaped by describing the general problem. This problem definition leads to the central research question that guides the research. Next, the research is justified by explaining its relevance. An overview of the remaining chapters of the thesis concludes this chapter.

1.1 Problem definition

Increased competition has forced many organizations to improve their business in order to stay competitive. Consequently, organizations are investing heavily in Information Technology (IT), because they believe it can help them achieve goals such as reducing costs, increasing the speed of service and improving their quality. Figures provided by Statistics Netherlands\(^1\) indicate that the Dutch industry and government invested a total of 12.6 billion euro in IT in 2004 alone [1]. However, despite a lot of scientific effort on improving success rates, several studies have shown that IT project success rates are unsatisfactory. Research by The Standish Group shows that more than 30% of IT projects gets cancelled before completion and another 50% significantly exceeds originally estimated costs [2]. A more recent report among dutch executives and IT managers shows that almost half of the IT projects partly fails [3]. It is critical to improve these success rates, especially for software development companies that have IT projects as their core business.

The failure of IT projects is often linked to shortcomings in the requirements phase. For example, in a study of a US Air Force project it was found that more than 40% of discovered errors could be traced back to errors in

\(^1\)Statistics Netherlands (Centraal Bureau voor de Statistiek) is a Dutch governmental institution that is responsible for collecting and processing data in order to publish statistics to be used in practice, by policymakers and for scientific research.
the requirements [4]. Furthermore, it is estimated that finding and fixing requirements errors account for 70 to 85% of IT project rework costs [5]. A particular difficult part of requirements engineering is dealing with and the management of non-functional requirements [6].

Practitioners and scholars often distinguish two kinds of requirements that together determine the complexity of an information system: functional requirements and non-functional requirements (NFRs). Functional requirements represent desired functionality in the system (i.e., what the system does). On the other hand, NFRs represent quality characteristics that the system must possess such as performance, reliability, security and usability [7, 8]. NFRs and quality are actually closely intertwined concepts [9]. Therefore, NFRs are also referred to as quality attributes [10, 11] or quality requirements [6].

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 [7] and it is rated as one of the ten biggest risks in requirements engineering [12]. At the same time, scholars have pointed out the importance of NFRs in IT projects [7, 9, 10, 13]. Serving as selection criteria for a number of decisions, among which architectural decisions [14, 8, 15, 11, 13], NFRs play a crucial role in software development [7]. According to [9], "there is a unanimous consensus that non-functional requirements are important and can be critical for the success of a project".

Despite these acknowledgements, the emphasis in scientific literature and practice has been on functional requirements, which is shown by the superficial treatment of NFRs in requirements engineering literature [16]. The discipline of dealing with functional requirements has matured from focusing on just specifying formal specifications to supporting the complete IT development process. Dealing with NFRs, on the other hand, has not been given much attention and as a result NFRs are still poorly understood [7, 16]. Obviously, there exists both a scientific and practical need to better understand NFRs and the impact that they might have on IT projects. Therefore, the main question that will guide the research is the following:

What is the impact of non-functional requirements on IT project success?

1.2 Research questions

The central research question will be answered by decomposing it in to three different sub questions which are introduced below.
The first sub question is concerned with IT project success. What is it exactly in the context of this research?

Dealing with NFRs presents practitioners with a lot of difficulties, but not all NFRs are the same because there are many different types of NFRs. Are there certain types of IT projects (based on their NFRs) that perform worse than others? The second sub question aims to address this question.

As a result of the difficulties in dealing with NFRs, specific approaches that can help practitioners in dealing with NFRs have been proposed in scientific literature. What approaches are applied in practice and do they have a positive influence on project success? This is addressed by the third sub question.

The sub questions introduced above can be summarized as follows:

1. What is IT project success in the context of bespoke software development?
2. Is there a significant relationship between the importance of non-functional requirements and IT project success?
   (a) What are non-functional requirements?
   (b) How do practitioners perceive the importance of non-functional requirements?
3. Is there a significant relationship between applying approaches for dealing with non-functional requirements and IT project success?
   (a) What approaches exist for dealing with non-functional requirements?
   (b) What approaches for dealing with non-functional requirements do practitioners apply?

1.3 Conceptual model

The focus of the present research is on investigating the two relationships depicted in the conceptual model, shown in figure 1.1, within the context of bespoke software development. On the one hand, the inclusion of dealing with NFRs poses a possible threat for the success of IT projects. 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.
The following preliminary hypotheses can be constructed from the conceptual model:

**H1** The importance of NFRs is negatively correlated with IT project success

**H2** Applying approaches for dealing with NFRs is positively correlated with IT project success

### 1.4 Relevance

When describing the relevance of a research it can be useful to distinguish between scientific and practical relevance. This section will explain why this research is relevant by describing scientific and practical contributions.

A scientific contribution of the present research is the investigation of the relationship between the importance of NFRs and IT project success. By answering the research question new knowledge is provided to the understanding of NFRs, which is particularly relevant since they are poorly understood [7, 16]. The identification of types of NFRs that are a threat to IT project success can be a trigger for future research as well, since it can serve as a starting point for the development of approaches tailored for dealing with a specific type of NFR. Furthermore, the relationship between
approaches for dealing with NFRs and IT project success is explored. A number of approaches have been proposed in scientific literature that could help to deal with NFRs. However, empirical evidence that support these approaches is lacking. Therefore, validating some of the underlying concepts of these approaches is important.

Organizations are struggling with IT projects, which is shown by the several studies cited earlier. The amount of IT projects failing is significant and NFRs are seen as critical for the success of IT projects [9]. It could be very helpful to know exactly what type of projects, in terms of the importance of NFRs, are less successful than others. It would allow practitioners to take appropriate measures for these underperforming projects. Additionally, an overview of the current practices in relation to types of NFRs of the case company is provided as a practical contribution.

A last contribution of this research can be found in the societal relevance. Governments are spending large amounts of money on IT projects [17] and there is no reason to assume success rates are any better than the statistics provided earlier. An important difference with industry is that public money is wasted if software projects fail in the public sector. Therefore, society benefits when a better understanding of NFRs and how to deal with them results in less project failures. More importantly, IT projects undertaken by the government are frequently concerned with the well-being of citizens in its broadest sense. For example, the safety of citizens is increased by the the Eastern Scheldt storm surge barrier (Oosterscheldekering) that needs adequate software to control the barriers to prevent a flood. It turns out NFRs are highly important in such IT projects. Other IT projects by the government aim to make life easier for citizens by providing a higher quality of service.

1.5 Thesis structure

The rest of this thesis aims to answer the main research question and the related sub questions. It is structured as follows. Chapter two describes the design of the research approach and the main research method that is used. Chapter three will provide an overview of related scientific literature on NFRs, project success and methods for dealing with NFRs. The survey operationalization is discussed in chapter four. The results of the survey are presented in chapter five. In chapter six the results are analyzed and limitations of the research will be discussed. The main conclusions and ideas for further research are presented in chapter seven, which concludes the thesis.
Chapter 2

Research method

This research roughly follows two approaches: exploratory research and theory-testing research. Therefore, it uses a combination of different methods (Figure 2.1). In the exploratory part, a literature research will investigate relevant theories and studies in order to come up with a set of hypotheses. These hypotheses will be used in the second part of the research where they will be tested based on quantitative analysis. Although it would perhaps be best to test causality between the independent variables and the dependent variable of this research, both practical and methodological limitations make it impossible to perform an experiment. A survey among architects is chosen as quantitative research method, because the required data for quantitative analysis was not readily available.

Figure 2.1: Overview of research approach
2.1 Theory-building

The theory-building research phase consisted of investigating relevant theories and research by literature research. Various scientific journals and published books on topics closely related to project success, NFRs and approaches for dealing with NFRs are studied in order to come up with relevant hypotheses that can be tested in the quantitative part of the research.

2.2 Theory-testing

Järvinen [18] provides a taxonomy of different studies that can help to map research questions to research methods which is shown in figure 2.2. Using his taxonomy, this research can be categorized as an empirical study on reality, because the influence of certain variables on the success of real IT projects is investigated. Furthermore, following his taxonomy it can be considered a theory-testing study where consensus is assumed, because common belief in scientific literature will be empirically tested. Commonly used research methods in this context are case studies, surveys and experiments.

A frequently used research method is the case study. In contrast to experiments and surveys, the case study refers to a group of methods that emphasize qualitative analysis [19]. It involves the investigation of some phenomenon in its natural setting which implies the researcher can only control the time and scope [20]. Methods such as interviews and observations are often used to collect data. The emphasis of a case study is on understanding the problem under investigation and allows for in-depth questioning or discussions. Disadvantages associated with the case study are mainly related to the lack of: controllability, deductibility, repeatability and generalizability [21]. There are also advantages compared to other research methods. In the domain of IS, three strengths of the case study are identified: "(1) the researcher can study information systems in a natural setting, learn about the state of the art, and generate theories from practice; (2) the method allows the researcher to understand the nature and complexity of the process taking place; and (3) valuable insights can be gained into new topics emerging in the rapidly changing information systems field." [22]. Therefore, case studies are most appropriate when contemporary events are studied, when it is not possible or necessary to control variables [19] and when theory is at an early stage [22].

A survey can be used to study a phenomenon in a variety of natural settings. In general, there is an emphasis on quantitative analysis. By collecting and analyzing data on a relatively large number of different objects, this approach tries to discover or verify relationships that are common across
Figure 2.2: Järvinen’s taxonomy of research approaches

these objects [23]. Most of the time, a sample has to be drawn from a much
larger population. Data is often collected through systematic methods such
as questionnaires, structured interviews and sometimes published statistics.
It is a relatively inflexible method: once the data collection has begun, little
can be changed when it turns out that some questions are not understood
by the respondents or something crucial is missing. This means that the
independent variables and dependent variables, as well as the model of the
expected relationships that will be tested, should be clearly defined up front
[20]. On top of that, the measurement of some variables of interest might
be very hard or impossible.

The last research approach that will be discussed is the experiment. An
experiment is a quantitative research method that involves the investigation
of some phenomenon in a controlled setting. The researcher is able to manip-
ulate the independent variables, while holding all other variables constant,
and observes the effects on the dependent variable [20]. An example would
be a drug trial, where one group of patients receive a certain drug and the
other group of patients receive a placebo. The researcher can test whether
the drug has an effect on the illness of the patients by comparing the results
from both groups. Baarda and de Goede [24] mention that experiments are
very well suited for testing causality between variables.

A survey was chosen as the theory-testing research method, because of
practical constraints and its emphasis on quantitative analysis. This re-
search focuses on Logica Netherlands (NL)\(^1\), which is a large technology
service company. The internal community of software architects was ad-
dressed to participate in the survey. The survey was designed by defining a
number of aspects which are described below.

2.2.1 Population and respondents

The unit of analysis is the unit about which statements are being made.
Essentially, it can be anything the researcher decides. Examples are an
individual, a certain group, an organization or a project. It must be carefully
chosen and defined to prevent a weak link between the unit of analysis and
the respondents [20]. The unit of analysis in this survey research is the latest
completed IT projects by architects of Logica NL, since we want to find out
which IT projects (in terms of the importance of different types of NFRs)
perform worse or subpar. Logica NL operates in different sectors which
includes energy, utilities and telecom, industry, distribution and transport,
finance, public and others. Others have pointed out that certain sectors
might have a bias towards particular types of NFRs:

"In the mobile phone domain size and power consumption are
central along with limited computer capacity, which can explain
why performance requirements are very common. Furthermore,
mobile phones are not what are usually considered to be safety
critical systems, such as trains or aeroplanes. This explains the
limited emphasis on reliability requirements. If a different do-
main was analyzed, the results would probably have looked dif-
ferent, suggesting that different domains require different solu-
tions." [25]

Although observations with regard to this bias will be addressed later,
it is not the primary objective to link sectors to types of NFRs. To keep
the research as general as possible and avoid a bias towards certain types
of NFRs, IT projects from all sectors that Logica operates in are included.
Therefore, the population consists of all the latest completed IT projects by
architects of Logica NL.

\(^1\)http://www.logica.nl/
This brings up the issue of choosing a suitable respondent, because the unit of analysis is an abstract object (IT projects) that can not be surveyed. It has been frequently argued that NFRs serve as selection criteria for a number of decisions, among which architectural decisions [14, 8, 15, 11, 13]. You could say that software architects are responsible for facilitating and realizing NFRs during software development. The relationship between architecture and NFRs is shown in figure 2.3. This makes software architects reasonable respondents in the context of this research.

Figure 2.3: Relationship between non-functional requirements and architectural decisions

2.2.2 Sampling procedure and data collection

The following procedure was used to collect data. All the members of the Architecture Community of Practice (ACoP) of Logica NL, which consists of professionals practicing architecture, were send an e-mail invitation to participate in the online survey. Among the architects are software architects, system architects and infrastructure architects. In the survey the respondents had to answer a number of questions about their latest completed IT project. An assumption is that every IT project has at least one architect working on it, if there are NFRs involved. If there are no NFRs involved in the IT project or there are no architects working on the IT project, it is of no interest for this research.

According to [20], "Sampling is concerned with drawing individuals or entities in a population in such a way as to permit generalization about
the phenomena of interest from the sample to the population.”. The most important concern in sampling procedures is representativeness. Using the procedure described above, significant overlap between collected responses (architects that worked on the same IT project) could limit the representativeness, which will be discussed later. Furthermore, the entire population of ACoP members was sent an invitation. However, not every Logica architect is a member of the ACoP. Although nothing is known about the non-response bias, it is assumed to be negligible.

Respondents were contacted through e-mail by sending an invitation to all members of the Architecture Community of Practice (ACoP) of Logica NL. The initial invitation was sent on 12-03-2010 and the survey remained online until 28-03-2010. A reminder was sent to the respondents a few days before the closing of the survey. The invitation and reminder was sent by one of the founders of the ACoP and member of the Central Technical Unit (CTU) of Logica NL. The CTU trains software architects on a regular basis, organizes workshops and consists of the most experienced software architects. The CTU also serves as ‘technical conscience’. Using a high entry point into the hierarchy of a company is recommended to improve response rates, which is important since they are generally very low [20]. Furthermore, a gift worth approximately 330\$\textsuperscript{2} (at the time of writing) was promised to be given away in a raffle among respondents that completed the survey.

The finalized survey was made available online by using a web survey tool by SurveyGizmo\textsuperscript{3}. SurveyGizmo provides extensive options to design surveys and format the questions and text. This was particularly important for readability purposes, because a number of definitions were provided to the respondents during the survey. Another feature of SurveyGizmo is that the dataset can be directly exported to a variety of file formats.

\section{2.3 Expert workshop}

An expert workshop will be held to validate the constructed survey. In particular, it aims to minimize possible threats to reliability and validity of the survey. The expert workshop is discussed in section 4.2.

\textsuperscript{2}A third generation iPod touch 32GB
\textsuperscript{3}http://www.surveygizmo.com
Chapter 3

Related literature

The previous chapters described the goal and the relevance of the research, followed by the research method(s) that are used. In this chapter, the independent and dependent variables of this research will be discussed by examining related literature.

3.1 Project success

Although project success has been an active topic in scientific research for many years, there is still no accepted standard approach. More traditional definitions of project success include meeting time schedule, meeting budget and meeting an acceptable performance level [26]. However, it has been acknowledged that these criteria do not give a complete view. Consider a project that met the planning objectives but has not met important end-user requirements. Clearly, from a client perspective the project is unsuccessful while the traditional criteria, that take on a managerial perspective, view the project as successful. Freeman and Beale [27] point out that success means different things to different people, which they illustrated by the following example:

"An architect may consider success in terms of aesthetic appearance, an engineer in terms of technical competence, an accountant in terms of dollars spent under budget, a human resources manager in terms of employee satisfaction. Chief executive officers rate their success in the stock market." [27].

These insights have led to multi-dimensional frameworks for the evaluation of project success that aim to reflect different perspectives. For instance, Pinto and Mantel [28] extended the traditional criteria by identifying three types of project performance that can be used to measure success: the implementation process, the perceived value of the project, and client satisfaction with the end-product of the project.
Baccarini [29] proposes a conceptual framework for project success. The logical framework method is used for the definition of two components of project success: project management success and product success. Project management success is process oriented and focuses on project success from a managerial point of view. It includes the traditional criteria mentioned above (time, cost and quality), quality of the project management process and stakeholder satisfaction. The product success components add criteria related to the product, like product goal and satisfaction.

Shenhar et al. [30] analyzed 13 success measures from previous research and grouped them into three dimensions: (1) meeting time, budget and requirements, (2) impact on the customer and (3) benefit to the performing organization. A two-staged study was performed in which qualitative and quantitative methods were combined. In the first stage, a multiple case study approach was applied to 15 projects to find major success dimensions across projects of different levels of technological uncertainty. Their findings indicate the emergence of a fourth dimension: future potential. The second stage of the study involved quantitative analysis of data on 127 projects from 76 companies. Project managers that participated were asked to evaluate the success of their project on several of the initial measures and to characterize the level of technological uncertainty. The authors conclude that the level of technological uncertainty of the project affects the importance of the success dimensions. Furthermore, the relative importance of the success dimensions is time dependent.

Lipovetsky et al. [31] studied the relative importance of a number of success dimensions in defense projects. Four success dimensions were defined based on previous research: meeting design goals, benefits to the customer, benefits to the developing organization, and benefits to the defense and national infrastructure. Three different stakeholders (customer, developing organization and the coordinating office) had to judge the relative importance of those dimensions for each of the 110 defense projects that they used. The results show that benefits to the customer is by far the most important success dimension followed by meeting design goals. The authors argue that the other two dimensions are relatively unimportant.

In later work by Dvir et al. [32], it is also stated that not every success dimension is of the same importance. Therefore, they determined success measures by performing a stakeholder approach in which they looked at success from the perspectives of the main stakeholders. In the context of their research they came up with three criteria to measure project success in a survey: meeting planning goals, end-user benefits and contractor benefits.
In the context of this research, it is also important to look at software project success as perceived by developers. Linberg [33] used structured interviews, documentation reviews and survey instruments to gather information from the perspective of software developers about a software development project that failed. The results of his study indicate that software developers’ perception of project success deviates from the general definition of project success in the software industry. It was also found that the job satisfaction of the software developers was not associated with the organizational goals or success measures. For instance, the team of software developers were able to maintain a high-level of job satisfaction while failing to meet time schedules and cost goals. Additional evidence is found in an exploratory study by Procaccino et al. [34]. They used a survey to discover important components of project success as perceived by software practitioners. What do practitioners mean when they talk about a successful project? Their results indicate that ”practitioners consider software projects successful if they provide intrinsic, internally motivating work to develop software systems that both meet customer/user needs and are easy to use.” [34].

3.2 Non-functional requirements

To get a better understanding of NFRs and requirements in general, it is useful to take a look at scientific literature and existing definitions. In almost every requirements classification, requirements concerned with the functionality of the system are distinguished from other requirements. There seems to be a general consensus on the definition of these so called functional requirements (FRs). An overview of a number of definitions of FRs found in literature is shown in table 3.1. It is clear that FRs are concerned with the functionality or behavior of the system (i.e., what the system does).
IEEE 610.12 [35] “a function that a system (…) must be able to perform”

Robertson and Robertson [36] “what the product must do”

Sommerville [37] “what the system should do”

Wiegers [38] “A statement of a piece of required functionality or a behavior that a system will exhibit under specific conditions.”

Jacobson, Booch, and Rumbaugh [39] “A requirement that specifies an action that a system must be able to perform, without considering physical constraints; a requirement that specifies input/output behavior of a system.”

Antón [40] “describe the behavioral aspects of a system”

Table 3.1: Overview of definitions for functional requirements

The other requirements are sometimes called non-functional requirements, but they are also referred to as qualities, attributes, constraints, characteristics or properties. Table 3.2 shows an overview of a number of definitions [9]. It is clear that a general definition of NFRs is non-existent.

The following examples are important differences between FRs and NFRs. First, FRs are related to specific functions while NFRs are related to architecture and affect several functions. This implies that a change in an NFR is potentially more complicated. Second, when implemented, FRs either work or not while NFRs have a sliding scale of good and bad. Third, NFRs can be in conflict with each other, because they fight for the same available resources. Trade-offs must be made in such situations.

### 3.2.1 Classification

There is also a wide variety of concepts used for sub-classifying NFRs. However, one frequently used concept for sub-classifying NFRs is quality. Perhaps one of the earliest attempts at establishing a conceptual framework of software quality is Boehm’s software quality tree [42]. His approach was as follows. As a starting point, a set of important software characteristics that was reasonably exhaustive and non-overlapping was defined. Candidate metrics were then developed that have the purpose of assessing the degree to which a system has these characteristics. Each candidate metric was then evaluated according to a number of criteria and with respect to other candidate metrics. Based on that evaluation, the set of software characteristics was refined into a more exhaustive and mutually exclusive set which resulted in a refinement of the candidate metrics as well. Analysis of the software
<table>
<thead>
<tr>
<th>Source</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antón [40]</td>
<td>&quot;Describe the nonbehavioral aspects of a system, capturing the properties and constraints under which a system must operate.”</td>
</tr>
<tr>
<td>Davis [41]</td>
<td>&quot;The required overall attributes of the system, including portability, reliability, efficiency, human engineering, testability, understandability, and modifiability.”</td>
</tr>
<tr>
<td>Robertson and Robertson [36]</td>
<td>&quot;A property, or quality, that the product must have, such as an appearance, or a speed or accuracy property.”</td>
</tr>
<tr>
<td>Wiegers [38]</td>
<td>&quot;A description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behavior.”</td>
</tr>
<tr>
<td>Jacobson, Booch, and Rumbaugh [39]</td>
<td>&quot;A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability. A requirement that specifies physical constraints on a functional requirement.”</td>
</tr>
<tr>
<td>Mylopoulos, Chung and Nixon [7]</td>
<td>&quot;global requirements on its development or operational cost, performance, reliability, maintainability, portability, robustness, and the like. (...) There is not a formal definition or a complete list of nonfunctional requirements.”</td>
</tr>
</tbody>
</table>

Table 3.2: Overview of definitions for non-functional requirements

characteristics and the metrics led to the insight that some of the software characteristics were related. For instance, any metric of understandability was also a measure of maintainability (two of their initially defined software characteristics). However, not every metric of maintainability had a relationship with understandability. This insight made it possible to model the software characteristics in a tree structure, which is shown in figure 3.1. The direction of the arrow indicates a logical implication. For software to be maintainable, it must be both understandable and testable or as the authors put it: "a high degree of maintainability implies a high degree of understandability and testability”. They also realized that there could be an even lower level of software characteristics below the level of understandability. After generating additional, lower level software characteristics, the whole set was represented as a tree which can be seen in figure 3.2.
Ultimately, all the characteristics determine the utility of the software which is why general utility is at the root of the tree. Furthermore, the general utility of software is decomposed into a number of higher level uses of the software. According to the authors, the following three questions are generally most important in evaluating these uses:

- How well can the software be used as it is?
- How easy is it to maintain the software?
- Can the software still be used when the environment changes?

Therefore, the general utility of the software is decomposed into as-is utility, maintainability and portability. The complete tree depicts a hierarchy between software characteristics, in which higher level characteristics represent different uses of the software and lower level characteristics are closely related to metrics that can be used to perform evaluations.
Figure 3.2: Boehm’s software quality tree
Another early model was McCall’s quality model [43]. The model, somewhat similar to the software quality framework described above and shown in figure 3.3 (adapted from [44]), defines software product qualities as a hierarchy of factors, criteria and metrics. The highest level is the management-oriented view of software product quality and it identifies the most important aspects of software quality. In McCall’s model they are called factors, but essentially they are NFRs. Below the quality factors are criteria which, if realized, provide the system with the characteristics that are represented by the quality factors. For each criteria a metric could be established that is an actual quantified measure of that criteria.

![Figure 3.3: McCall's software quality framework](image)

The quality factors are further defined by evaluating how a manager views the end product of an IT project. Three management-oriented viewpoints are distinguished that are related to life cycle activities: product operations, product revision and product transition. Each of these viewpoints has a number of associated quality factors.

Another quality model is established in the international standard ISO/IEC 9126 [45] for software product quality. A software product can also be an architectural description or some kind of system. The standard consists of the quality model and three types of metrics. Just like the models described above, it can be used to convert abstract goals and priorities of the project to values that can be measured and evaluated. This will help develop a common understanding of the objectives of the project. In the
quality model, software quality is decomposed as a set of characteristics and sub-characteristics. The highest level of characteristics are:

- Functionality: attributes that are related to the functions of the software product
- Reliability: attributes that have to do with the capability of the software product to stay operational
- Usability: attributes that have to do with the effort required by users to use the software product
- Efficiency: attributes that are concerned with the level of performance and the usage of resources of the software product
- Maintainability: attributes that relate to the required effort to make modification in the software product
- Portability: attributes that concern the software product’s ability to be transferred to a different environment

Each characteristic is divided into sub-characteristics. For instance, usability is divided into understandability, learnability, operability, attractiveness and usability compliance. The standard defines three types of metrics: internal metrics, external metrics and quality in use metrics. Internal metrics are static measures that do not need any execution of the software or system. External metrics apply to running software or systems. Quality in use metrics are measures when the software or system is being used in real conditions. The assumption is that internal quality influences external quality and external quality is related to quality in use.

The Architecture Tradeoff Analysis Method [46, 47] was developed by the Carnegie Mellon Software Engineering Institute. It is a method for evaluating software architecture decisions with respect to desired quality goals. In the context of this method, a distinction is made between runtime qualities (availability, performance and security) and development time qualities (modifiability and integration).

FURPS, or FURPS+, can also be used to classify NFRs. The model was developed by Hewlett-Packard. IBM suggests using it to capture architectural requirements [48]. FURPS is an acronym of Functionality, Usability, Reliability, Performance and Supportability, where the ”URPS” part represent the NFRs that are assumed architecturally relevant. The ”+” represents concerns that pose constraints, such as design requirements, implementation requirements, interface requirements and physical requirements.
3.3 NFR Approaches

It is acknowledged that NFRs play a critical role in software development. 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 [7] and, in worst case, can lead to project failure. On top of that, it is claimed that practitioners find dealing with NFRs the most difficult part of requirements engineering [6]. This makes explicit the need for ways to manage NFRs and 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.

Paech [16] reviews the state of the art NFRs engineering and groups activities into the following categories, or NFR approaches:

- Identifying NFRs from several perspectives and different levels of detail
- Supporting the discovery of dependencies and conflicts among NFRs, in order to prioritize them accordingly
- Documenting NFRs and evaluating this documentation
- Supporting the identification and evaluation of means to realize NFRs, in order to make trade-off decisions accordingly
- Supporting change and project management

Similarly, Berntsson Svensson [6] systematically reviews literature on managing NFRs. Activities are grouped into the following NFR approaches, that are associated with the software product management domain:

- Elicitation
- Quantification
- Prioritization
- Dependencies
- Cost estimation
- Software product management in general
The software product management domain refers to market-driven software development in which a supplier develops a product for a specific market. In contrast, the scope of this research is bespoke software development: the supplier develops a custom-made product or service for a single customer. There are a number of differences between market-driven software development and bespoke software development. For instance, time-to-market pressure in market-driven software development might make planning of subsequent releases and prioritization of requirements more important compared to bespoke software development [49]. For completeness purposes it is assumed that most of the NFR approaches identified by [6] are valid in the context of this research, even though the context of this research is bespoke software development.

Not surprisingly, both groupings are very similar and contain some overlap. An overview of the overlap is shown in table 3.3. For instance, the identification approach has similarities with elicitation and quantification. Elicitation, in essence, is identifying NFRs from different viewpoints and quantification can be considered making NFRs more detailed. Furthermore, analyzing conflicts among NFRs and prioritizing NFRs is literally included in both studies. There are also some differences between the groupings. Documentation is only recognized as an approach by Paech, while cost estimation is only recognized as a separate approach by Berntsson Svensson. Paech does mention cost estimation as a way to perform trade-off analysis, but includes it in the evaluation of realizations approach. Related literature to the approaches will be discussed next. However, cost estimation is omitted, because only one related paper is found [6].

<table>
<thead>
<tr>
<th>Paech et al. [16]</th>
<th>Svensson et al. [6]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Elicitation</td>
</tr>
<tr>
<td>Supporting prioritization by uncovering</td>
<td>Dependencies</td>
</tr>
<tr>
<td>dependencies and conflicts</td>
<td>Prioritization</td>
</tr>
<tr>
<td>Documentation</td>
<td>Not included</td>
</tr>
<tr>
<td>Evaluation of realizations</td>
<td>Cost estimation</td>
</tr>
<tr>
<td>Supporting change and project manage-</td>
<td>Software product management in general</td>
</tr>
<tr>
<td>ment</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.3: Overlap between approaches

### 3.3.1 Elicitation and documentation

Requirements elicitation is concerned with revealing and discovering requirements, for the product that will be developed, from the main stakeholders. Compared to eliciting FRs, the elicitation of NFRs faces additional
challenges: there is a lack of understanding of NFRs and NFRs are generally stated informally [50]. A number of studies that proposed techniques or methods for the elicitation of NFRs will be discussed in this section.

Doerr et al. [13] developed the NFR method, which was systematically evaluated in three industrial case studies performed with companies in different domains using three different systems. As a result, different types of NFRs are considered in the case studies. The NFR method provides guidance in requirements elicitation by using workshops to capture and elicit NFRs. The goal of the method is to arrive at a minimal and sufficient set of NFRs that is both traceable and measurable. It has the following features:

- A process for the common treatment of types of NFRs based on the ISO 9126 classification (maintainability, efficiency, portability, usability, security and reliability)
- Experiences with the characteristics of NFRs, metrics that can be used to measure these NFRs and realizations are captured in experience-based Quality Models
- Different types of NFRs are distinguished which helps eliciting and documenting them
- Checklists and a prioritization questionnaires to provide detailed guidance
- Documentation guidelines provided by templates
- Rationales to justify NFRs
- Integrated treatment of FRs, NFRs and architecture
- Requirements management support

The first case study was performed on a distributed, embedded system (a wireless plant control) together with a large German company. Efficiency, reliability and maintainability were chosen for further discussion in workshops after filling out the prioritization questionnaire. The authors conclude that using their method new NFRs were discovered that were otherwise overlooked. Experts judged that these NFRs had an impact on the architecture, which would have led to a lot of rework. In addition, a high degree of measurable NFRs was achieved, because only five out of the 54 elicited NFRs were not measurable. According to the authors, this is a significant improvement compared to previous methods. The second case study was performed together with Ricoh, a Japanese manufacturer of copiers and
printers. The method was applied on another embedded system (a multifunctional printer). The report focused on efficiency since it is generally most important in this specific domain. In this case study, the checklists and Quality Models encouraged the elicitation of NFRs during the stakeholder workshops. Moreover, the collaborative nature of the workshop and the checklists made it easier for the requirements engineers and developers to reach agreement on the granularity of requirements. A disadvantage of the method is that the quality of NFRs that are discovered with the method depends on the quality of the FRs that serve as input. The development of a web-based geographical information system was considered in the third case study. Security was found to be the most important type of NFR in this context. The method, by means of the workshop and checklists, helped elicit new NFRs that were very important for the system, but had not been considered before. Furthermore, the elicited NFRs helped by providing a basis for evaluating architectural decisions. In general, experts argued that there was more time spend on NFRs than before, but the return on investment was positive.

Another technique that follows an integrated approach is the Structured Hierarchical Interview for Requirement Analysis (SHIRA) [51]. SHIRA is an interviewing technique that tries to capture qualities, design approaches that meet these qualities and the relationships among them. Abstract qualities are distinguished from concrete qualities. Concrete qualities lead to the desired abstract qualities and design approaches realize concrete qualities. An objective of SHIRA is to identify new and important concrete qualities. The interview consists of the following parts:

- Introduction: the interviewee is introduced to the system being designed and its intended use
- Choose abstract qualities: the interviewee has to choose a number of abstract qualities from a set that is predetermined
- List concrete qualities: the interviewee has to list concrete qualities of the system that would justify the chosen abstract qualities
- Design approaches: the interviewee has to provide ideas of how the concrete qualities can be addressed with designs

The SHIRA technique was evaluated by conducting 18 interviews to support the design of a home automation system. The participants were introduced to the system by means of illustrations of typical usage scenarios. Three abstract qualities were then chosen from a predetermined group of 15 abstract qualities. In the following steps, a total of 172 concrete qualities were listed by the participants and 474 design approaches were provided.
Therefore, on average, each abstract quality resulted in approximately three concrete qualities. Furthermore, each concrete quality resulted in approximately three design approaches, on average. The authors mention SHIRA is a fairly generative approach. Five experts are consulted to evaluate its usefulness. Although they generally found it useful, a number of disadvantages were pointed out: one expert pointed out the limited quality of the design approaches put forward by the interviewees and others criticized a "lack of information that helps defining usability requirements". A last attempt at evaluating SHIRA was made by comparing their resulting requirements with the results of a number of focus groups that addressed the same system. The fact that they found a strong overlap looks promising, because the focus groups used a total of 75 participants and they could benefit from the social interactions among the participants. Still, further evaluations of SHIRA are needed according to the authors.

JaeJoon et al. [52] encounter a number of difficulties during the scenario’s and requirements gathering process in the context of the ATAM method. In summary: there is no consensus on NFRs, the availability of various stakeholders is limited, the stakeholders have biased viewpoints, constructing scenarios can be difficult and unfamiliarity with quantitative evaluation. Therefore, a strategy for eliciting NFRs is proposed which consists of the following steps:

- Select the types of NFRs to be elicited
- For each selected type of NFR, reach a consensus on its meaning to guarantee a common understanding
- Develop scenario elicitation forms
- Determine appropriate metrics for each NFR
- Prioritize the NFRs

According to the authors, the strategy helps to trace components of the architecture to related NFRs. To evaluate the strategy, the maintenance costs of the existing architecture (using an older elicitation method) was compared to the maintenance costs of the new architecture (using the new strategy). The new architecture that was created using the proposed strategy resulted in an estimated maintenance costs reduction of about 10 to 50%.

The Quality Attribute Workshop (QAW) [53] is an important elicitation method in the context of this research, because it is used by Logica architects in practice. The QAW is a stakeholder engaging method that can be used
early in the development life cycle for the discovery of important NFRs. It is developed by the Software Engineering Institute to complement the Architecture Tradeoff Analysis Method [46]. The aim of the method is to identify and clarify NFRs before the design of the architecture. The QAW consists of the following steps:

- Presentation of the business and mission drivers of the system.
- Presentation of a high-level system description or architectural plan.
- Identification of architectural drivers.
- Scenario generation by brainstorming.
- Consolidation of similar scenarios.
- Prioritization of scenarios.
- Refinement of scenarios.

The result of the QAW yields a list of architectural drivers, scenarios, a prioritized list of scenarios and refined scenarios. These artifacts can be used to design the architecture. Additionally, the scenarios can be used to evaluate the architecture after it has been created if the complementary Architecture Tradeoff Analysis Method is used as evaluation method.

### 3.3.2 Dependencies and conflict analysis

It has been observed that the different types of NFRs can have dependencies, or conflicts, among them. For example, Zulzalil [54] used three different web application domains to investigate the interdependencies among four types of NFRs. Strong correlations were found between some types and the author concludes that there are indeed dependencies among NFRs which can be either negative or positive. For example, increased security generally leads to decreased performance. On the other hand, increased performance could increase usability as perceived by users. Several methods or tools to identify or resolve dependencies among NFRs will be discussed below.

The WinWin system is discussed by Boehm et al. [10]. It is a manual method that alleviates group decisions and can be used to identify and resolve conflicts among NFRs. The main step of the WinWin system consists of the stakeholders formulating their win conditions using a schema that is provided to them. This makes it possible to identify conflicts based on the win conditions. If there is a conflict, an issue schema is provided which summarizes the conflict and the win conditions it involves. The stakeholders then prepare an option schema for each issue, in which the issue is addressed.
The different options are then evaluated which allows the stakeholders to converge to a satisfactory option. Finally, after checking that this option covers the stakeholders’ win conditions, it is agreed upon in an agreement schema. Figure 3.4 shows the relationships between the different schemas.

![WinWin schema overview](image)

The authors also discuss the Quality Attribute Risk and Conflict Consultant (QARCC) tool. QARCC is a knowledge-based tool that can be used to identify potential conflicts early in the development life cycle. NFRs and related architecture tradeoffs are considered by QARCC: "It may tell you, for example, that a layered architecture will improve portability, but usually at some cost in performance." [10]. QARCC works with the same artifacts as the WinWin system. For instance, it takes the win conditions schemas of the stakeholders as an main input. The QARCC tool is evaluated in an experiment with a ground satellite systems by comparing it to the WinWin system. The WinWin system was able to successfully identify two significant conflicts and supported the negotiation of a new solution. The same conflicts were identified by QARCC. Moreover, it found eight more possible conflicts that were not identified by the WinWin system. Five of these conflicts were considered significant for the development of the ground satellite system. The main conclusion is that QARCC can be helpful for developers and other stakeholders by pointing out conflicts and potential conflicts among NFRs. QARCC needs further development to prevent an overload of insignificant conflict suggestions.
Poort et al. [55] propose the Non-Functional Decomposition (NFD) process, an iterative divide-and-conquer strategy for requirements conflict resolution with an emphasis on NFRs. It is based on the following important experience-based observations: similar supplementary requirements are grouped in the same subsystem of good architectures and if a subsystem deals with conflicting requirements it is better to separate the parts that cause the conflict. The NFD process consists of the following steps. First, requirements are gathered and prioritized using any elicitation or prioritization technique. It is important that primary functional requirements (the main functionality of the system) are mapped to the supplementary requirements (other requirements that usually constrain the implementation of the primary functional requirements, for instance NFRs). According to one of the observations, the second step groups the functions of the system based on the supplementary requirements. In-group and grouping conflicts among the supplementary requirements are identified. Grouping conflicts are conflicts between two different function groups and in-group conflicts occur when there is a conflict within a function group. If a group has an in-group conflict it is split accordingly. Finally, when there are no more in-group conflicts the resulting set of grouping conflicts serves as the basis for the architectural decomposition. The NFD process is evaluated by applying it in two case studies. The result was a good and documented traceability between the requirements and architectural design decisions. The authors conclude that this traceability helped communicating to stakeholders how design decisions are related to their requirements.

The NFR framework is a process-oriented approach in the context of dealing with NFRs [56]. A unique feature of this process is the fact NFRs are treated as an integral part during development. NFRs are modeled as conflicting or synergistic goals in a softgoal interdependency graph (SIG). Consequently, design alternatives that realize the NFRs can be evaluated using tradeoff analysis. Figure 3.5 shows an example SIG for performance and security in a credit card system [57].
3.3.3 Prioritization

Prioritization of requirements or NFRs in particular was mentioned a number of times in the discussions of the earlier approaches. For instance, NFRs are prioritized in the last step of JaeJoon’s strategy for elicitation and the first step of the NFD process consists of prioritizing requirements. However, no particular method or technique is proposed for performing NFR prioritization. This section will discuss a number of methods to prioritize requirements. Because of the many methods available to prioritize requirements, only a few methods of different complexity are described. They assume a priority value or level is associated with every requirement. Several aspects can be used to evaluate the priority of a requirements: importance, cost, time and risk are some examples [58]. It is important to stress that there are a number of larger methods or decision making frameworks that use the techniques discussed below.

**Ranking** Perhaps the easiest technique to prioritize requirements is to let a stakeholder rank requirements, without ties, from most important to least important. The resulting ranking is based on an ordinal scale: it is possible to see which requirements are more important than others, but not by how much more important. Therefore, it is not possible to see the relative differences. Berander [58] reports that this technique is more suitable
for situation where there is only a single stakeholder like in bespoke development, because combining several rankings from different stakeholders could be difficult. Two algorithms that can be used to obtain the ranking are bubble sort and binary search tree. Bubble sort starts out by putting the requirements in a list and repeatedly moving through it, comparing the priority of each pair of requirements adjacent to each other and swapping them if they are not in the right order. This process of passing through the list repeated until there are no swaps needed in a single pass, because that means the list is sorted. The total number of comparisons for a project with \( n \) requirements is \( n \times \frac{n-1}{2} \).

**Priority groups**  Priority groups are the most common prioritization technique according to [58]. It involves grouping requirements into groups of different priorities. Although the number of different groups (and thus priorities) can vary, it is reported that three is common in practice. Using descriptive words that describe the priority of each group is important for a reliable classification. It is not advised to use terms such as high, medium or low, because it will confuse the stakeholders. Furthermore, stakeholders might have different perspectives which means they have a different view on what high, medium or low is. Rather, the terms should have a real meaning to the stakeholders like in the MoSCoW technique described below. Another potential problem with prioritizing requirements using priority groups is that stakeholders might view all the requirements as critical. It is reported that customers generally classify 85% of the requirements as critical, 10% as standard and 5% as optional if all requirements are grouped in to one of three groups [58]. Restrictions on the number of requirements that are allowed in each group can be used, but it might decrease the usefulness of the priorities. The result is a set of requirements groups that are prioritized on an ordinal scale. Requirements within a group are all of the same priority.

A particular popular technique is the acronym MoSCoW. The technique was developed by Dai Clegg and is now part of the Dynamic Systems Development Method (DSDM) Consortium. Its capital letters stand for:

- Must have this
- Should have this if possible
- Could have this if it does not affect anything else
- Won’t have at this time but would like to have in future

All requirements, functional and non-functional, are put in one of the four categories which can help to discuss what is really important. The requirements classified as "Must have" are non-negotiable, because the system will
not work without them. Requirements that are nice to have and add extra value are classified as "Should have" or "Could have". The inclusion of the fourth category, "Won’t have", distinguishes MoSCoW from other techniques. Discussing not only what the customer wants, but also what he does not want, provides a different perspective and can help define a better and prioritized requirements list.

100-dollar test  Regnell et al. [49] use a variant of the 100-dollar test to prioritize 17 groups of requirements. The original 100-dollar test is very simple: stakeholders are given an imaginary amount of 100 units to distribute between the requirements. However, distributing only 100 units between a large number of requirements is problematic. For instance, prioritizing 25 requirements means there are on average only four points to distribute for each requirement. A larger fictitious amount of 100000 units is used in the study by Regnell et al. to alleviate this problem. The authors mention that the subjects were positive of this variant, especially the use of the larger amount to distribute. Another important observation is that the prioritization should only be performed once on the same set of requirements, because participants could bias their evaluation the second time if their favorite requirements were not ranked high enough.

Analytical Hierarchy Process  The Analytical Hierarchy Process (AHP) is a decision-making method which can be used to prioritize requirements. Prioritizing with AHP entails comparing all unique pairs of requirements and determining which has a higher priority and to what extent. The total number of comparisons for a project with \( n \) requirements is \( n \times \frac{n-1}{2} \). Thus, the number of required comparisons dramatically increases as the number of requirements increases which makes AHP not suitable if there are large numbers of requirements. Variants have been developed that are able to reduce the number of comparisons significantly. Consider the following example: requirement A has a higher priority than requirement B and requirement B has a higher priority than requirement C. By the transitive property, requirement A should also have a higher priority than requirement C. However, the last comparison (requirement A compared with requirement C) is still performed if AHP is used. The redundancy can be removed by creating a minimal spanning tree in a directed graph. In this case, only \( n-1 \) comparisons are enough to determine relative priorities. Although this technique is much faster than regular AHP, errors in judgment are more likely to occur since all redundancy is removed.

Overview  Table 3.4 presents an overview and compares the discussed techniques.
<table>
<thead>
<tr>
<th>Technique</th>
<th>Scale</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranking</td>
<td>Ordinal</td>
<td>Easy</td>
</tr>
<tr>
<td>Priority groups</td>
<td>Ordinal</td>
<td>Very easy</td>
</tr>
<tr>
<td>100-dollar test</td>
<td>Ratio</td>
<td>Complex</td>
</tr>
<tr>
<td>AHP</td>
<td>Ratio</td>
<td>Very complex</td>
</tr>
</tbody>
</table>

Table 3.4: Comparison of prioritization techniques

### 3.3.4 Quantification

Olsson et al. [59] empirically analyzed how NFRs were quantified and what metrics are used in the mobile phone domain. About 40% of the requirements in the requirements specification were NFRs. Different scales were used to quantify the NFRs. The most frequently used scale was absolute values: 58% of all quantified NFRs used an absolute number without an interval. Other scales that were found include an interval using a minimum and a maximum or a one-sided interval where only an upper or lower bound is specified. The authors concluded that different development areas treated NFRs differently. Therefore, it is argued that different areas (both with respect to technological domain and types of NFRs) would benefit from tailored methods, tools and support.

Jacobs [60] presents a method called Gilb-style to improve requirements engineering which is evaluated in a case study. The focus of the method is on NFRs and quantification. Gilb-style uses several concepts from Planguage [61] to make NFRs measurable:

- Gist: a rough summary of the requirement
- Scale: the unit in which the requirement will be measured
- Meter: how to perform the measurement
- Past: benchmark value typically achieved
- Record: highest value achieved
- Must: represents minimal goal level
- Plan: represents desired goal level

The use of Gilb-style was introduced to a software development department which led to improved understanding of requirements, a focus on NFRs and a change of behavior.
The quality performance model (QUPER) [62, 63, 25] supports reasoning about and quantification of NFRs by combining cost and benefit views. QUPER is developed for the software product management domain and can be used to align the quantification of NFRs with the market (or competitors). The main observation underlying QUPER is that in contradiction to FRs, quality and NFRs are often continuous and nonlinear in nature instead of either "good" or "bad". For example, significant investments might be required for only a small increase in performance. QUPER uses two basic concepts: breakpoints and barriers. A breakpoint is a quality level point where a significant shift in benefit occurs. For example, when the start-up time of a mobile phone shifts from a level that is expected to a level that outperforms most competitors. On the other hand, a quality level point where a significant shift in cost occurs is called a barrier. QUPER combines three views:

- The benefit view is concerned with the relationship between quality and benefit in terms of breakpoints.
- The cost view is concerned with the relationship between quality and cost in terms of barriers.
- The roadmap view combines the benefit and cost views to assess the current situation and targets.

The benefit view of the model is of special interest for quantification of NFRs and can be seen in figure 3.6. It includes three main breakpoints. The utility breakpoint represents a border between a quality level that is perceived as useless and a quality level that is becoming useful. The shift in benefit from a quality level that is useful to a quality level that outperforms most competitors is called the differentiation breakpoint. The last breakpoint is called the saturation breakpoint and it indicates the border between a quality level that is competitive and a quality level that is excessive.

![Figure 3.6: The benefit view of QUPER [62]](image-url)
Berntsson Svensson [25] evaluated QUPER by interviewing experts that used the model for three months. The inclusion of quality level intervals in relation to the market was seen as a useful feature by the experts. They found that adjusting the quality level interval of NFRs (determining the breakpoints) based on the market or competitors provides a better basis for the actual quantification.
Chapter 4

Survey operationalization

The rationale behind the choice for a survey approach and the design of the research method was described in the previous chapter. This chapter will elaborate on the construction of the survey itself and the operationalization of the constructs. Finally, the validation of the survey constructs is discussed as well as threats to reliability and validity of the survey.

The survey starts with a short introduction to the objective of the research. The survey consists of four sections. The first section consists of questions that are related to the general characteristics of the latest completed project of the respondent. Examples are "How many months ago was your latest project completed?" and "In which sector did your latest completed project take place?". The second section asks the respondent to evaluate the success of his or her latest completed project from a number of perspectives. Respondents are 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. The survey starts out with easy questions. Gradually, the questions become more difficult before concluding with some easier questions. It is generally accepted that such a sequence is pleasant for the respondent [64]. The complete survey consists of 23 questions and can be found in Appendix A. It took the respondents 17 minutes, on average, to complete.

4.1 Constructs

The different sections of the survey are roughly described above. Not yet explained is how the variables of interest are measured. This section will explain the different constructs of this research in order of appearance in the survey.
4.1.1 Project success

In the second section of the survey, respondents are asked to evaluate the success of their latest completed IT project. Project success is a concept that has been subject of discussion for a long time and it has been used many times as a construct. An overview of project success in scientific literature can be found in chapter 3. The stakeholder approach proposed by [32] is followed in this research, because it helps defining a success construct for a specific situation. The main stakeholders are analyzed to determine project success measures that reflect their different perspectives. The main stakeholders in the projects to be analyzed are: the project manager(s), the customer and the architect(s). From the perspective of the project manager, project success can be measured with the traditional criteria of project success that are included in all of the references above. More specifically, the criteria included to reflect the managerial perspective in this research are meeting time schedule and meeting budget. The overview above also points out the importance of the customer perspective. For instance, it was found that benefit to the customer was considered to be the most important success criterion by all stakeholders [31]. Additionally, the context of this research is bespoke software development where a single customer is usually the principal stakeholder [65]. Customer satisfaction is therefore included as a success measure. Finally, a criterion that reflects the perspective of the software architect should be included as a success measure. Several researchers have mentioned the deviation of software developers’ perception of project success from the traditional criteria. For instance, Linberg [33] found that the definition of project success of a team of software developers was different from traditional definitions. Also, the team of developers still maintained a high level of job satisfaction even though they failed to meet time and cost schedules. More recently, Procaccino [34] confirmed these observations. Poort et al. [66] have similar experiences: ”Developers (including architects) tend to judge success by criteria that extend beyond the project, sometimes even to the extent that even canceled projects can be successful in their eyes.”. In their study, which also uses software architects as respondents, solution quality is used as a measure of the architects’ perception of project success. Therefore, it is also used in this research.

The project success construct consists of four dimensions, that are supposed to reflect the interests of the three main stakeholders. Meeting time and budget corresponds to project success from a managerial perspective, 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 a software architect. 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 suc-
cessful). Statistical techniques such as reliability analysis with Cronbach’s $\alpha$ \cite{67} will be used to see if these dimensions can be added together to represent a single project success construct.

4.1.2 Non-functional requirements

The third part of the survey requires the respondents to rate the importance of several types of NFRs. The operationalization of this construct consisted of two parts: the classification of NFRs into sub types and defining importance.

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 \cite{8}. The advice is the following: "No matter what classification scheme a software practitioner might choose to adopt, the most important thing to bear in mind is that s/he should know what s/he means by an NFR term, such as performance, so that the meaning of such an NFR can be communicated with the user as well as with system/software developers so that the end product will behave as expected." \cite{8}. Furthermore, many of the published classifications and definitions of NFRs have their own communities in science and practice \cite{11}. Therefore, since a number of architects of Logica NL is trained in the ATAM method of the Software Engineering Institute, the six most common and important types of NFRs distinguished by them are used in the present research \cite{11}:

- **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).

- **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.

- **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.

- **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.

The definition of the importance of types of NFRs and how it is measured was the primary objective of the validation session with architecture experts. Therefore, it is discussed in section 4.2.

4.1.3 **NFR Approaches**

The fourth part of the survey asks the respondents to indicate what approaches are applied for dealing with NFRs during their latest completed IT project. The approaches, that are linked to requirements engineering activities, are described in chapter 3. The following approaches are 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**: non-functional requirements are made explicit by giving them numbers on a measurable scale. This makes the non-functional requirements verifiable.

• **Prioritization**: assigning priorities among the different non-functional requirements on the basis of their relative importance.

• **Conflict analysis**: identifying the interdependencies and conflicts among the non-functional requirements.

The operationalization of the constructs for the approaches means defining how they are measured in the survey. First, we want to know which of the approaches were applied. The simplest way would be to ask respondents if each approach was applied using a yes/no format. However, this is not sufficient. Several studies have pointed out that the relative costs of correcting (requirements) errors increases during the development life cycle. In an empirical study it was concluded that the cost of errors that are not corrected increases exponentially with each phase they are not fixed [68]. Others found that it can cost up to 110 times more to correct a requirements
defect that is found when the system is operational compared to finding the
same defect during the requirements definition phase [69]. In line with these
findings, it is assumed that applying an approach later in the development
life cycle is less effective. 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 op-
tion.

4.2 Expert workshop on the survey

The operationalization described above resulted in a draft survey. This
draft survey was discussed during an expert workshop: a session with eight
architecture experts that are a member of the CTU of Logica NL. The
emphasis of the session was on the constructs. Additional important con-
siderations when designing survey questions were addressed as well [70]:

- The questions should be consistent
- The respondents should be able to answer the questions
- The respondents should be willing to give correct and valid answers
- The respondents should be able to understand the questions

During the discussion of the IT project success construct, the experts
judged that an extra dimension was wanted. From a managerial perspec-
tive, a project that is both on time and within budget can still be considered
unsuccessful if it utilized resources inefficiently. Therefore, efficient use of
resources was added. Furthermore, a number of practical terms that are
associated with the success dimensions were added to increase understand-
ability for the respondents.

The most important item to be discussed was how to operationalize the
NFR construct. How do you measure the importance of each type of NFR
for a project? It was mentioned that simply asking for the number of re-
quirements for each type of NFR is not valid. Intuitively, a project could
have only a few performance requirements that are critical for the system.
At the same time, 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. Most of the time it is the problem 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. 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. On the other hand, the measure in which NFRs that are not business critical are fulfilled has a low impact on the system’s business value. NFRs that are not within the scope of a project have a very low business criticality. Respondents are 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).

An observation made earlier on the inconsistencies among different NFR classification schemes became a point of discussion as well. For instance, to some experts reliability was different from availability, while others thought of them as different terms for the same thing and treated reliability as simply a sub type of availability. Likewise, recoverability was thought of a sub type of availability. Therefore, it was decided to include a number of sub types in the definitions of the six types of NFRs in the survey to increase understandability.

The last point of discussion was the approaches. Most experts recognized the approaches that are distinguished, but it was advised to define an additional approach that covers verification activities:

- Verification: verifying that a system fulfills requirements, e.g. by prototyping, simulation, analysis, testing or other means.

On the other hand, it was unclear whether many experts were familiar with actual formal methods apart from generic workshops. Explanations were added to indicate that any form of an approach would suffice, whether it was informal or formal. One expert thought that he had to choose one approach that was applied from the available approaches. Therefore, an explanation was added to prevent respondents from making the same assumption. With regard to the project phases, the experts judged that they were sufficiently aligned with the majority of the projects carried out by Logica. An explanation of the project phases was added the survey to increase understandability. It was indicated that the project phases used are meant as generic names for any systems development life cycle and respondents should attempt to map them to the most appropriate phase in their project.
4.3 Reliability and validity

This section describes reliability and validity concerns. A survey that is reliable measures the constructs consistently, which makes it repeatable: it will yield similar results when performed again. Internal validity is concerned with whether the constructs in the survey actually measure what they are supposed to measure. Finally, external validity has to do with the ability to generalize the findings beyond the boundaries of the present research.

4.3.1 Reliability

Cronbach’s $\alpha$ [67] is used as a reliability test to assess internal consistency of the multi-item scales (constructs) described above. As shown by the results (table 4.1), the criticalities of the types of NFRs can not be added together to form one NFRs criticality construct, because the internal consistency is low ($\text{Cronbach} \alpha = .398$). However, all the other constructs can be considered reliable ($\text{Cronbach} \alpha > .8$).

<table>
<thead>
<tr>
<th>Construct</th>
<th>Number of items</th>
<th>Cronbach’s $\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT project success</td>
<td>5</td>
<td>.858</td>
</tr>
<tr>
<td>Criticality NFRs</td>
<td>6</td>
<td>.398</td>
</tr>
<tr>
<td>Applying elicitation</td>
<td>6</td>
<td>.801</td>
</tr>
<tr>
<td>Applying documentation</td>
<td>6</td>
<td>.873</td>
</tr>
<tr>
<td>Applying quantification</td>
<td>6</td>
<td>.860</td>
</tr>
<tr>
<td>Applying prioritization</td>
<td>6</td>
<td>.923</td>
</tr>
<tr>
<td>Applying conflict analysis</td>
<td>6</td>
<td>.938</td>
</tr>
<tr>
<td>Applying verification</td>
<td>6</td>
<td>.909</td>
</tr>
</tbody>
</table>

Table 4.1: Reliability analysis of constructs

4.3.2 Internal validity

According to [71], internal validity is about whether valid conclusions can be drawn or not and has to do with the level of control on the study. For a survey the level of control is usually low: social desireability might cause respondents to answer untruthfully or other factors might bias the results in some way like history effects.

Social desireability is the tendency for respondents to provide answers that will make them look better or will avoid making them look bad [70]. In the context of this survey, the questions about IT project success and the approaches pose a risk in the context of social desireability. For instance, a respondent might distort the answers on IT project success because he or
she thinks that low success makes him look bad. A slightly different example is the following: answers on the applied approaches might be distorted, because the accurate answer is not aligned with the way the respondent wants to think about him- or herself. The effect of this distortion is addressed by allowing respondents to stay anonymous and explicitly stating the answers will be treated confidential. Consistency of responses is addressed by making every question mandatory. A respondent can not continue to the next page of the survey until every question is filled out. On the other hand, a disadvantage of this approach is that it might force some respondents to fill in an answer when they actually can not provide an accurate answer. For instance, the project might be too long ago for the respondent to remember the details. This is partly addressed by only considering the latest completed projects of the respondents. Finally, there is a significant amount of people that did not respond and they might be different from those who did respond. As mentioned before, nothing is known about the non-response bias.

4.3.3 External validity

A factor that limits the external validity of this research is the fact that all included IT projects are carried out by the same company. A cross-organizational approach would have been better, but due to practical limitations this was not possible. However, the software development company where this research was carried out has many similarities with other companies. Furthermore, a choice was made to include IT projects from all sectors to prevent a bias towards certain types of NFRs. The results of this research represent findings across all sectors, which make them general in nature. Lastly, it should be noted that the context of the research is architecture, since it has a strong link with NFRs. As will be pointed out later, the results do not generalize well to other groups of respondents such as project managers.

4.4 Refining the hypotheses

Now that the constructs are operationalized, the hypotheses H1 and H2 can be further refined by decomposing them. Hypothesis H1 is decomposed into:

H1a The business criticality of availability is negatively correlated with IT project success

H1b The business criticality of performance is negatively correlated with IT project success
H1c The business criticality of modifiability is negatively correlated with IT project success

H1d The business criticality of security is negatively correlated with IT project success

H1e The business criticality of usability is negatively correlated with IT project success

H1f The business criticality of testability is negatively correlated with IT project success

Hypothesis H2 is decomposed into:

H2a Applying elicitation is positively correlated with IT project success

H2b Applying documentation is positively correlated with IT project success

H2c Applying quantification is positively correlated with IT project success

H2d Applying prioritization is positively correlated with IT project success

H2e Applying conflict analysis is positively correlated with IT project success

H2f Applying verification is positively correlated with IT project success
Chapter 5

Results

This chapter presents the results of this research.

5.1 Data preparation

A number of precautions were taken to ensure the integrity of the collected data and the data was prepared for further statistical analysis. This section will describe how and why certain preparations were made.

All the responses were checked for duplicates by looking at the names of the respondents. Two responses were completed by someone with the same name and it turned out they were the exact same person. Therefore, one of the responses was deleted. Three respondents who completed the survey wished to remain anonymous and could only be checked by inspecting their answers. It is safe to assume these responses were no duplicates because their experience, business unit and latest completed project were different. When inspecting partially completed responses for duplicates it was found that two responses were made by the same person and one of them was deleted. One response that was labelled as partially complete was added to the completed responses, because it turned out to be fully complete.

Another important preparation was filtering responses on type of function that the respondents had during their latest completed project. As explained earlier, a choice was made to use software architects as respondents, because of the strong link between NFRs and architecture. However, not every member of the ACoP that received the invitation for the survey is necessarily a software architect. Respondents that were not a software architect during their latest completed project were filtered out. A comparison between the responses of software architects and other respondents will be described later.
5.2 Responses

When the survey was closed, 81 completed responses were collected (including non-architects) of which 39 came from software architects. Another 51 respondents had completed the survey partially. However, in section two of the survey a definition is given of an IT project. If the latest completed project of the respondent is not such a project, he or she is asked to leave the survey. As a result, 36 respondents left the survey at this point. The other 15 respondents that partially completed the survey left somewhere else in the survey: 2 left when they were asked about the success of the project, 3 left when they were asked about the criticality of the NFRs and 10 left in the section about the approaches.

The ACoP has a total of approximately 350 members that were sent an invitation through the mailing list. The dataset consisted of 81 completed responses after preparation (non-architects included), which means the response rate is 23%, but this is a very inaccurate and pessimistic percentage. First of all, 36 respondents left the survey because their latest completed project did not meet the requirements and they were asked to leave the survey. It would be inaccurate to include these observations as non-responses: they left because they were asked to and were not supposed to be surveyed in the first place. If we adjust the total number of members of the ACoP according to this observation, we get a more realistic response rate of approximately 26%. Simply adjusting the total members of the ACoP with this number could also be inaccurate, because we only know how many of the respondents had to leave the survey. We do not know how many of the non-respondents would have had to leave the survey. If we assume that the respondents are unbiased with regard to the total ACoP, approximately 27% ($\frac{36}{81+51}$) of the 350 members of the ACoP would have had to leave the survey because their latest project did not meet the requirements. If these observations are taken into consideration, a response rate of approximately 31% ($\frac{81}{0.73 \times 350}$) would be more accurate. These statistics are very similar to an earlier study that also used a survey with the members of the ACoP as respondents [66]. They collected a total of 142 responses (partially completed responses included) among the members of the ACoP. In this research, a total of 132 responses were collected if we include partially completed responses.

5.3 Descriptive statistics

The main features of the data set will be described by presenting some descriptive statistics. Only the 39 responses from software architects are used (N = 39). First, for the responses to be accurate the latest com-
pleted projects of the respondents had to have been completed recently. If a project is completed recently, it is to be expected the respondent would have less trouble remembering details accurately compared to a project that is completed a long time ago. Almost all of the responses that are collected are about projects that were completed at most two years ago (figure 5.1). Furthermore, on average the projects were completed approximately seven months ago (M = 7.74, SD = 9.55).

![Histogram of project completion](image)

Figure 5.1: Histogram of project completion

Closely related to when a project was completed is the total duration of the project in number of months (figure 5.2). On average, projects were of medium length lasting for just over a year (M = 13.44, SD = 12.05).

The distribution of project sizes is shown in figure 5.3. Most of the projects are small to medium in size (M = 26.67, SD = 29.76). A small number of relatively large sized projects are also included.
A choice was made to include projects from all sectors to avoid possible bias towards certain types of NFRs. Figure 5.4 shows the distribution of the projects over the sectors that Logica distinguishes. Although some sectors are better represented than others it is assumed that projects are sufficiently distributed to avoid a significant bias. The two projects that fall under the 'Other' category are an internal project and a project for Logica Middle East.
An overview of the project success construct is shown in figure 5.5. Success scores range from 5 to 25, because the construct is created by adding together the scores on the five individual success dimensions. Projects have medium success on average, because most of the respondents rated their latest completed project between medium and successful ($M = 16.49$, $SD = 4.15$).

Figure 5.4: Distribution of the projects over the sectors

Figure 5.5: Histogram of project success
Lastly, an overview is given of how the software architects rated the business criticalities of the NFRs. The respondents rated the business criticality of availability relatively high (M = 3.87, SD = 0.767), which can be seen in figure 5.6. There was only one project where availability was considered of low business criticality.

![Figure 5.6: Perceived business criticality of availability](image)

Shown in figure 5.7, respondents rated the business criticality of performance high on average (M = 3.74, SD = 0.88). There was not a single project where performance had a very low business criticality.

![Figure 5.7: Perceived business criticality of performance](image)

Figure 5.8 shows that modifiability was considered less business critical than availability and performance on average (M = 3.28, SD = 0.916).
Security was rated highly business critical on average (M = 3.67, SD = 1.009) which is shown in figure 5.9.

Shown in figure 5.10, the respondents rated usability highly business critical on average (M = 3.74, SD = 0.993).
5.4 Perception

In order to find out how the software architects perceive NFRs and the role that they play during IT projects, they had to rate their agreement with three statements on a 5-point Likert-scale (strongly disagree, disagree, neutral, agree, strongly agree) and estimate the frequency of one statement on a 5-point Likert-scale (never, sometimes, often, very often, always). In total 39 responses of software architects were collected (N = 39) to the statements.

**Statement 1** In general, non-functional requirements are critical for the success of IT projects
As described earlier, several researchers claim that NFRs are critical for the success of IT projects. Although the claim is open for more than one interpretation, it is interesting to see if practitioners agree. Figure 5.12 shows that more than half of the respondents strongly agreed with the statement (51.3%) and the rest agreed with the statement (48.7%). Not a single respondent was in disagreement with or neutral towards the statement.

**Statement 2** *In general, non-functional requirements are difficult to deal with*

Another claim that has been made in scientific literature is that practitioners are struggling to deal with NFRs. As a result, many approaches have been proposed in the requirements engineering domain for dealing with
NFRs. Do practitioners perceive dealing with NFRs as a difficult task? The results, shown in figure 5.13, indicate that the opinions are more divided than in the previous statement. More than half of the respondents at least agreed with the statement (46.2% agreed and 10.3% strongly agreed). Almost a quarter of the respondents was neutral (23.1%) and the rest disagreed (20.4%).

**Statement 3** *Non-functional requirements are the main reason for IT project failure*

![Figure 5.14: Frequencies of possible responses to statement 3: non-functional requirements are the main reason for IT project failure](image)

A significant amount of respondents indicated that NFRs are critical for the success of IT projects and difficult to deal with. There are even a number of examples found in literature, where neglecting NFRs or not properly dealing with NFRs is the reason for IT project failure. Do practitioners recognize NFRs as a possible reason for IT project failure? Respondents had to rate how often NFRs are the main reason for IT project failure. Figure 5.14 shows the results. The majority of the respondents think that NFRs are often (38.4%) or very often (23.1%) the main reason for IT project failure. The rest thinks that NFRs are only sometimes the main reason for failure (38.5%).

**Statement 4** *In general, applying any of the approaches to deal with non-functional requirements has a positive influence on the success of IT projects*
As explained earlier, there are a lot of methods and techniques for dealing with NFRs (and requirements in general) in requirements engineering literature. Roughly, these methods and techniques belong to one of six approaches. The approaches are supposed to help practitioners deal with and manage NFRs, which in turn should increase the success of the IT project. Do practitioners perceive the approaches as potential ways to increase the success of their IT projects? The results are shown in figure 5.15. One respondent disagrees with the statement (2.6%) and four other respondents are neutral (10.3%). However, a big majority of the respondents thinks that the approaches have a positive influence on the success of IT projects (76.9% agreed and 10.2% strongly agreed).

### 5.5 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. Respondents (N = 39) rated the business criticality of six types of NFRs during their latest completed project on a 5-point Likert-scale (very low, low, medium, high, very high). Furthermore, they rated the success of 5 aspects of their latest completed project on a 5-point Likert-scale (very unsuccessful, unsuccessful, neutral, successful, very successful). A single success construct was created by adding together the scores of the different aspects (Cronbach α = .858). Significant findings are further explored by also looking at the correlation with the separate success aspects. Since six types of NFRs are distinguished, six hypotheses were constructed and the results of the statistical tests are presented in this section.
There are a number of statistical correlation coefficients to investigate the correlation between two variables. The value of these coefficients range between -1 (perfect negative correlation) and +1 (perfect positive correlation), and a value of 0 indicates that there is no correlation. A well known correlation coefficient is Pearson’s product-moment correlation. Although it is frequently used, it is not suited for this particular situation, because it assumes the data is measured at interval ratio and is normally distributed. Spearman’s $\rho$ or Kendall’s $\tau$, so called rank correlation coefficients, can be used when the parametric assumptions are violated. While Spearman’s statistic is more popular, it is suggested that Kendall’s statistic is a better estimate of the correlation in the population and it is better suited when you have a small dataset with a lot of tied ranks [72]. Therefore, each hypothesis is tested using Kendall’s $\tau$ (one-tailed) and the level of statistical significance is .05 ($\alpha = .05$). Significant correlations will be visualized by boxplots. The boxplots of correlations that are not significant can be found in Appendix B.

**H1a** The business criticality of availability is negatively correlated with IT project success

Figure B.1 shows a boxplot of the correlation between the business criticality of availability and project success. The experimental hypothesis **H1a** has to be rejected. The correlation between the business criticality of availability and IT project success is not significant, $\tau = .114$, $p$ (one-tailed) > .05.

**H1b** The business criticality of performance is negatively correlated with IT project success

Kendall’s $\tau$ is again used to investigate the correlation between performance and IT project success. A boxplot of the correlation is shown in figure B.2. Hypothesis **H1b** is rejected, because the correlation between performance and IT project success is not significant, $\tau = -.181$, $p$ (one-tailed) > .05.

**H1c** The business criticality of modifiability is negatively correlated with IT project success
The results (visualized in figure 5.16) indicate that there is a significant negative relationship between the business criticality of modifiability and IT project success, $\tau = -.257$, $p$ (one-tailed) < .05. Hypothesis $H_{1c}$ is therefore not rejected.

**H1d** *The business criticality of security is negatively correlated with IT project success*

The correlation between the business criticality of security and IT project success is visualized in B.3 and is not significant, $\tau = .078$, $p$ (one-tailed) > .05. Therefore, hypothesis $H_{1d}$ has to be rejected.

**H1e** *The business criticality of usability is negatively correlated with IT project success*

No significant correlation is found between the business criticality of usability and IT project success, $\tau = -.102$, $p$ (one-tailed) > .05. Figure B.4 visualizes the correlation.
**H1f** The business criticality of testability is negatively correlated with IT project success

The experimental hypothesis **H1f** has to be rejected based on the results (figure B.5). There is no significant correlation between the business criticality of testability and IT project success, $\tau = .095$, $p$ (one-tailed) > 0.5. A summary of the results is presented in table 5.1.

<table>
<thead>
<tr>
<th>Type of NFR</th>
<th>Kendall’s $\tau$</th>
<th>Sig. (1-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>.086</td>
<td>ns</td>
</tr>
<tr>
<td>Performance</td>
<td>-.181</td>
<td>ns</td>
</tr>
<tr>
<td>Modifiability</td>
<td>-.257</td>
<td>.023</td>
</tr>
<tr>
<td>Security</td>
<td>.078</td>
<td>ns</td>
</tr>
<tr>
<td>Usability</td>
<td>-.102</td>
<td>ns</td>
</tr>
<tr>
<td>Testability</td>
<td>.095</td>
<td>ns</td>
</tr>
</tbody>
</table>

Table 5.1: NFRs and their correlation coefficient with IT project success

### 5.6 Approaches and project success

Six requirements engineering approaches proposed in 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 is described earlier. A hypothesis was constructed for each approach. Statistical techniques are used to test the hypotheses and the results are presented in this section. Again, each hypothesis is tested using Kendall’s $\tau$ (one-tailed) and the level of statistical significance is .05 ($\alpha = .05$). Significant correlations will be visualized by boxplots. The boxplots of correlations that are not significant can be found in Appendix B.

**H2a** Applying elicitation is positively correlated with IT project success

As shown in figure B.6, no significant correlation was found between applying elicitation and IT project success, $\tau = .054$, $p$ (one-tailed) > .05.

**H2b** Applying documentation is positively correlated with IT project success
The results (visualized in figure B.7) indicate that there is no significant correlation between documentation and IT project success, $\tau = .065$, $p$ (one-tailed) > .05.

**H2c Applying quantification is positively correlated with IT project success**

No significant relationship is found between applying quantification and IT project success, $\tau = .024$, $p$ (one-tailed) > .05. Figure B.8 shows the correlation in a boxplot.

**H2d Applying prioritization is positively correlated with IT project success**

Based on the results, shown in figure B.9, the experimental hypothesis H2d has to be rejected. The correlation between applying prioritization and IT project success is not significant, $\tau = .057$, $p$ (one-tailed) > .05.

**H2e Applying conflict analysis is positively correlated with IT project success**

The correlation between applying conflict analysis and IT project success is not significant, $\tau = -.128$, $p$ (one-tailed) > .05.

**H2f Applying verification is positively correlated with IT project success**

The correlation is visualized in figure 5.17. There is a significant positive relationship between applying verification and IT project success, $\tau = .256$, $p$ (one-tailed) < .05. Therefore, hypothesis H2f is not rejected.

A summary of the results is presented in table 5.2.

<table>
<thead>
<tr>
<th>NFR Approach</th>
<th>Kendall’s $\tau$</th>
<th>Sig. (1-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elicitation</td>
<td>.054</td>
<td>ns</td>
</tr>
<tr>
<td>Documentation</td>
<td>.065</td>
<td>ns</td>
</tr>
<tr>
<td>Quantification</td>
<td>.024</td>
<td>ns</td>
</tr>
<tr>
<td>Prioritization</td>
<td>.057</td>
<td>ns</td>
</tr>
<tr>
<td>Conflict analysis</td>
<td>-.128</td>
<td>ns</td>
</tr>
<tr>
<td>Verification</td>
<td>.256</td>
<td>.014</td>
</tr>
</tbody>
</table>

Table 5.2: NFR Approaches and their correlation coefficient with IT project success
5.7 Comparison with other functions

As explained earlier, a choice was made to include architects only, because architecture has a strong link with NFRs. Furthermore, it is argued that project success as perceived by architects can deviate from other functions such as project managers [33, 34, 66]. As an example, figure 5.18 shows the project success ratings for meeting time schedules for both the architects (N = 39) and the other functions (N = 42) which includes project managers, business consultants and test engineers.
To find out whether the success scores differ significantly, statistical analysis using the independent means $t$-test can be used. However, the collected data is measured on an ordinal scale which violates the assumptions of parametric tests [72]. Therefore, the non-parametric Mann-Whitney test is used to investigate whether the success scores differ significantly. Architects rated the success of their latest completed project in terms of meeting time schedules ($\text{Mdn} = 3$) significantly lower than respondents with other functions ($\text{Mdn} = 4$), $z = -2.431$, $p$ (two-tailed) $< .05$. Figure 5.19 shows the project success ratings for solution quality for both the architects ($N = 39$) and the other functions ($N = 42$).
Figure 5.19: Success of solution quality

Again, the non-parametric Mann-Whitney test is used to investigate whether the success scores differ significantly. The results indicate that architects rated the success of their latest completed project in terms of solution quality (Mdn = 3) significantly lower than respondents with other functions (Mdn = 4), $z = -2.083$, $p$ (two-tailed) $< .05$. In fact, for every success dimension architects gave lower scores on average. However, the scores did not differ significantly for any of the other dimensions. A summary is shown in table 5.3.

<table>
<thead>
<tr>
<th>Project success dimension</th>
<th>Z</th>
<th>Sig. (2-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meeting time schedule</td>
<td>-2.431</td>
<td>.015</td>
</tr>
<tr>
<td>Meeting budget</td>
<td>-.079</td>
<td>ns</td>
</tr>
<tr>
<td>Efficiency</td>
<td>-.884</td>
<td>ns</td>
</tr>
<tr>
<td>Customer satisfaction</td>
<td>-1.733</td>
<td>ns</td>
</tr>
<tr>
<td>Solution quality</td>
<td>-2.083</td>
<td>.037</td>
</tr>
</tbody>
</table>

Table 5.3: Success dimension rating differences between architects and other functions using the Mann-Whitney test
The results of the architects differ significantly compared to the other functions for two of the five success dimensions. Not surprisingly, the solution quality success dimension that is explicitly included to reflect the deviating perceptions of architects is rated significantly different by the other functions. Including the responses by other functions would have an effect on the correlation analysis described above. Therefore, it is argued that leaving the other functions out of the analysis is justified.

5.8 Summary

The key findings of the quantitative analysis in this chapter are:

- Software architects perceive availability as the most business critical type of NFR on average
- The business criticality of modifiability is negatively correlated with project success
- The application of verification is positively correlated with project success
- Software architects perceive project success differently compared to others such as project managers
Chapter 6

Discussion

Earlier chapters described how theory was translated to a number of hypotheses that became subject to statistical analysis. The findings, that were described in the previous chapter, indicate that most of the hypotheses have to be rejected. However, a few hypotheses were accepted as well. These results were discussed in a session with architecture experts as a form of validation. This chapter will discuss the results and their implications as well as the mentioned expert workshop. The conclusion of this research follows and recommendations for future research are given.

6.1 Descriptive statistics

The previous chapter showed descriptive statistics of the business criticalities of the types of NFRs. They indicate that on average the business criticality of availability is highest. Earlier studies found similar results. For instance, reliability was identified as the most important type of NFR in software platform development [73]. Furthermore, reliability was ranked as the most important NFR and availability was ranked as the most important sub-characteristic for intranet applications [74]. 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. Since this research most likely includes projects of a different domain, the results suggest that availability is the most important type of NFR regardless of the type of project (in terms of technological domain, sector and type of system that is developed). Another observation is that the types of NFRs are almost never unimportant, because very few respondents rated the business criticality of a 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.
6.2 Perception

The software architects in this research agree that NFRs are critical factors for the success of IT projects. However, the statement is ambiguous and open to different interpretations. Nevertheless, when developing a system, NFRs represent quality characteristics desired by the customer. That observation by itself makes NFRs important, because neglecting any requirement will most likely decrease customer satisfaction. However, it is argued that NFRs can have an impact on the managerial success dimension as well. NFRs are realized through architectural designs, which makes for instance a change relatively difficult and costly. Furthermore, NFRs are frequently in conflict with each other which could make realizing every NFR impossible.

Not surprisingly, the majority of the software architects agrees that dealing with NFRs can be difficult. What they find difficult about dealing with NFRs was not explicitly investigated. However, anecdotal evidence in the form of frequently cited comments by the respondents indicate the following difficulties regarding NFRs. A lot of NFRs remain implicit, because they have not been elicited or given enough attention. This results in a customer that expects some common qualities to be included at an acceptable level by default, while the developers don’t have any actual requirements that state these qualities. It seems a lot of customers think very differently about FRs and NFRs: FRs have to be paid for and certain NFRs are included by default. As mentioned by some respondents: managing customer expectations, validation of requirements and verification of requirements should be among the top priorities of requirements management or project management.

6.3 Non-functional requirements and project success

The results show that the business criticality of modifiability is negatively correlated with IT project success. In other words: on average, IT projects where modifiability is relatively important are significantly less successful than IT projects where modifiability is relatively unimportant.

The following possible explanations for this result were discussed in a session with software architecture experts. 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. Further investigation supports this explanation: the business criticality of modifiability is only significantly negatively correlated
with the customer satisfaction success dimension. Other possible explanations mentioned during the expert workshop are that techniques to realize high modifiability are not used (or available) or high modifiability could lead to too much complexity. However, evidence supporting these theories is not available.

No correlation is found between the business criticality of the other types of NFRs (availability, performance, security, usability and testability) and IT project success. Especially the lack of a significant correlation between the business criticality of availability and IT project success was surprising according to some architecture experts that attended the expert workshop. Given the relative high average business criticality of availability it was decided to discuss this further. A number of possible explanations were formulated during the session. First, it was mentioned by some experts that the results could be distorted by the fact that IT projects that are foremost concerned with availability are treated by a unit that is specialized on availability, which would lead to economies of learning and thus more successful projects. Second, it was thought that these projects are assigned with the most experienced architects on average, which would explain why they perform relatively well. However, this theory is not supported by the dataset. Finally, IT projects where availability is relatively business critical often have very technical customers, which would allow them to communicate much better. The lack of a significant correlation between the business criticality of performance and IT project success was surprising as well, but due to lack of available time it was not thoroughly discussed during the expert workshop.

6.4 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 reduces errors, it seems that it is still not standard practice to start verifying as early as possible. Practitioners should recognize the benefits eventually make up for the extra costs.

It is surprising that all the other hypotheses had to be rejected. 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. Second, the 6-point Likert-scale used is based on a general waterfall systems development life cycle and is therefore not suited for iterative development methodologies. During the validation session, the experts judged that they were sufficiently aligned with the majority of the projects carried out by Logica. However, at least one respondent had trouble answering the questions about the application of the approaches, because his projects always use iterative development.

6.5 Other limitations

A few important limitations of this research have to do with generalizability. First, in the previous chapter it was shown that the findings do not generalize well to other groups such as project managers. The context of the research is architecture, since it has such a strong link with NFRs. 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 software development company where this research was carried out has many similarities with other software development companies. Third, the sample size used in this research is relatively small despite taking a number of efforts to increase response rates: the survey was electronically distributed, the research was sponsored by someone with a high entry point in to the hierarchy of the organization and a gift was given away in a raffle as another incentive.

Instead of using a well known standard for sub classifying NFRs such as ISO/IEC 9126, the most common and important types of NFRs distinguished in the domain of software architecture were used [11]. The rationale behind this choice was that understandability was increased. However, it does limit the comparability of the results since a lot of studies used for instance the ISO/IEC 9126 standard [73, 74].
Chapter 7

Conclusion

The main research question that was formulated in the introduction was as follows:

What is the impact of non-functional requirements on IT project success?

The answer can be split in two parts. 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 underperforming 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. Practitioners dealing with IT projects characterized by a focus on modifiability should be careful. Managing customer expectations and such might require additional attention, because it seems that customer satisfaction especially is significantly lower on average in these type of IT projects. The business criticality of performance might have some influence on IT project success as well, even though no direct significant relationship was found. Further investigation using partial correlation (adjusting the relationship for the positive effect of applying verification), strictly speaking not allowed because the procedure makes use of Pearson’s correlation coefficient, does seem to reveal a possible significant relationship between the business criticality of performance and IT project success.

The second part of the answer 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.

7.1 Future research

This study investigated the impact of types of NFRs and applying approaches for dealing with NFRs on IT project success. If we combine the conclusions that were drawn with the discussion of the findings and the limitations of this research a number of recommendations for future research can be made.

The significant relationship that was found between the business criticality of modifiability and IT project success can be a research trigger. NFRs were sub classified using a high level classification scheme, which means the resulting classification is coarse. For instance, modifiability could be further sub classified in to adaptability, maintainability and compatibility. Which of these sub types the respondents exactly meant was not measured. Further research might be able to differentiate between these sub types of modifiability. More importantly, unsuccessful IT projects where modifiability is highly business critical could be studied to provide additional evidence that could help explain the relationship that was found in this research.

The measurement of the applied approached was mentioned as a limitation of this research. 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 IT projects. Another recommendation for future research would be to use a different kind of measurement for project success. Including the actual customer and his evaluation of a project’s success would be a great improvement.

Finally, anecdotal evidence shows that practitioners have trouble dealing with customers that treat NFRs differently from FRs. A customer might expect a certain level of quality, but does not realize these NFRs have to be made explicit. On top of that, they are reluctant to pay for NFRs, because they think they are standard or implicit. However, NFRs have to be realized by designing an appropriate architecture just like FRs have to be implemented by program code and the customer will have to pay for that. If a customer wants a higher level of quality it will cost him. The QUPER model might be very useful, because its underlying assumptions
can not only help quantifying NFRs that are otherwise left implicit, but it can also help explaining the relationship between quality level, benefit and costs. Further research might evaluate QUPER by applying it in the bespoke software development domain to overcome the mentioned troubles with customers.
Bibliography


Appendix A

Survey

Survey on the impact of non-functional requirements on IT project success

Figure A.1: Introduction to the survey

Dear member of the NL Logica Architecture Community of Practice,

My name is Nick Martens and I am a student at the Utrecht University. Currently, I am working on my master thesis within Logica’s Working Tomorrow program, closely co-operating with Eltjo Poort, NL Architecture Expert. The aim of my research is (a) to investigate the impact of non-functional requirements on the success of IT projects and (b) to validate practices for dealing with non-functional requirements. This is done by analyzing a number of IT projects by means of this survey. We consider you an expert on the field of non-functional requirements. Therefore, we would like you to share your view on a number of topics regarding your latest completed project. Definitions of key concepts are provided during the survey to ensure common understanding of the terms. If the precise answer is missing, please choose the answer that is most applicable.

Your input is highly valued, and the survey results will be used by Eltjo Poort and the NL Central Technical Unit to help improve the practice of architecture within Logica. Your answers will be treated confidential. As a token of my appreciation, participants will be rewarded with a final report containing the results of this survey. The survey consists of 23 questions, and takes approximately 15 minutes to complete.

Thank you,
Nick Martens
The following three questions are optional.

1. What is your name?

2. What is your practice unit?

3. How many years have you been working in the IT industry?

Figure A.2: Optional questions
In this part of the survey, you are asked to provide contextual information on your latest completed project.

**Project:** the questions in this survey are meant to obtain information from the perspective of your latest completed IT development project. An IT development project is a project where an IT system (application, software, infrastructure or other IT system) is designed, constructed and implemented. The work done by you for this project could either be spanning just a small part of the total duration of the project, or it could span the total duration of the project. If you have not been involved in such a project, please do not respond to the survey.

4. What was the number of project members in your latest completed project?*

5. What was the total duration in months of your latest completed project?*

6. How many months ago was your latest project completed?*

Figure A.3: Questions about context of the project
7. In which sector did your latest completed project take place?*
   - Energy, Utilities & Telecom
   - Industry, Distribution & Transport
   - Finance
   - Public
   - Other (please specify)

8. Where were you located for your latest completed project?*
   - 100% at Logica office
   - 75% at Logica office / 25% at customer
   - 50% at Logica office / 50% at customer
   - 25% at Logica office / 75% at customer
   - 100% at customer

9. Which of the following functions best describe your responsibilities in your latest completed project?*
   - Project manager
   - Architect
   - Designer
   - Developer
   - Test engineer
   - Reviewer/auditor
   - Business consultant
   - IT consultant
   - Other (please specify)

Figure A.4: Questions about context of the project
10. How do you rate your latest completed project in terms of the following aspects?

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Very Unsuccessful</th>
<th>Unsuccessful</th>
<th>Neutral</th>
<th>Successful</th>
<th>Very Successful</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meeting time schedule (business deadlines)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Meeting budget (business case)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Efficient use of resources (time, money, staff)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Customer satisfaction</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Solution quality</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

Figure A.5: Project success questions
Please rate the importance of a number of non-functional requirement types during your latest completed project. The types are based on the book "Software Architecture in Practice – Second Edition" by the Software Engineering Institute (SEI).

Non-functional requirements: desired quality characteristics that the end system must possess. They are also known as quality attributes or quality requirements. The most important distinguished types in this survey are: availability, performance, modifiability, security, usability and testability. If the precise answer is missing, please choose the one that is most applicable.

- **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 other 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.

Figure A.6: Questions about criticality of the NFRs
Business criticality: A non-functional requirement is considered business critical when it is vital to the customer's business. The measure in which highly business critical non-functional requirements are fulfilled has a high impact on the system's business value, and vice versa. On the other hand, the measure in which non-functional requirements that are not business critical are fulfilled has a low impact on the system's business value. Non-functional requirements that are not within the scope of your latest completed project have a very low business criticality.

11. How do you rate the business criticality of the following non-functional requirements during your latest completed project?

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modifiability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure A.7: Questions about criticality of the NFRs
In this part of the survey, you are asked about the ways that non-functional requirements were dealt with in your latest completed project.

**Practice:** A practice is defined as a way to deal with or manage non-functional requirements during IT projects. These practices can be applied using formal methods (i.e., using a Quality Attribute Workshop) or informally. Projects may use multiple practices to deal with non-functional requirements.

- **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:** non-functional requirements are made explicit by giving them numbers on a measurable scale. This makes the non-functional requirements verifiable.
- **Prioritization:** assigning priorities among the different non-functional requirements on the basis of their relative importance.
- **Conflict analysis:** identifying the interdependencies and conflicts among the non-functional requirements.
- **Verification:** verifying that a system fulfills requirements, e.g. by prototyping, simulation, analysis, testing or other means.

**Project phases:** The project phases used below are meant as generic names for any lifecycle used: please attempt to map them to the most appropriate phase in your project. The distinguished project phases are: requirements (inception, request for proposal), design (functional design, solution architecture), realization (coding, construction), testing (user acceptance testing, FAT, SAT, integration test) and deployment (transition, delivery, installation).

---

**Figure A.8: Questions about applied approaches**
12. **Elicitation:** When were the following non-functional requirements elicited for the first time in your latest completed project?*

<table>
<thead>
<tr>
<th>Requirements phase</th>
<th>Design phase</th>
<th>Realization phase</th>
<th>Testing phase</th>
<th>Deployment phase</th>
<th>Later/Neve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modifiability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13. **Documentation:** When were the following non-functional requirements documented for the first time in your latest completed project?*

<table>
<thead>
<tr>
<th>Requirements phase</th>
<th>Design phase</th>
<th>Realization phase</th>
<th>Testing phase</th>
<th>Deployment phase</th>
<th>Later/Neve</th>
</tr>
</thead>
<tbody>
<tr>
<td>availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reliability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>usability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure A.9: Questions about applied approaches
14. **Quantification**: When were the following non-functional requirements quantified for the first time in your latest completed project?*

<table>
<thead>
<tr>
<th></th>
<th>Requirements phase</th>
<th>Design phase</th>
<th>Realization phase</th>
<th>Testing phase</th>
<th>Deployment phase</th>
<th>Later/Never</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Performance</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Modifiability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Security</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Usability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Testability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

15. **Prioritization**: When were the following non-functional requirements prioritized for the first time in your latest completed project?*

<table>
<thead>
<tr>
<th></th>
<th>Requirements phase</th>
<th>Design phase</th>
<th>Realization phase</th>
<th>Testing phase</th>
<th>Deployment phase</th>
<th>Later/Never</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Performance</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Reliability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Usability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Testability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>
16. **Conflict analysis:** When were the following non-functional requirements analyzed for conflicts for the first time in your latest completed project?*

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Requirements phase</th>
<th>Design phase</th>
<th>Realization phase</th>
<th>Testing phase</th>
<th>Deployment phase</th>
<th>Later/Never</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Performance</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Modifiability</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Security</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Usability</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Testability</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

17. **Verification:** When were the following non-functional requirements verified for the first time in your latest completed project?*

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Requirements phase</th>
<th>Design phase</th>
<th>Realization phase</th>
<th>Testing phase</th>
<th>Deployment phase</th>
<th>Later/Never</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Performance</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Reliability</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Integrity</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td>Stability</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
<td>☐</td>
</tr>
</tbody>
</table>

Figure A.11: Questions about applied approaches
These last questions are more general and they do not refer to your latest completed project specifically. How much do you agree with the following statements?

18. In general, non-functional requirements are critical for the success of IT projects.*

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

19. In general, non-functional requirements are difficult to deal with.*

<table>
<thead>
<tr>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20. Non-functional requirements are the main reason for IT project failure:*

<table>
<thead>
<tr>
<th>Never</th>
<th>Sometimes</th>
<th>Often</th>
<th>Very Often</th>
<th>Always</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure A.12: Statements about NFRs
21. In general, applying any of the practices to deal with non-functional requirements has a positive influence on the success of IT projects.*

<table>
<thead>
<tr>
<th></th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

22. Rank the following practices from most important to least important for project success.*

1] Elicitation
2] Documentation
3] Quantification
4] Prioritization
5] Conflict analysis
6] Verification

23. Do you have any last comments regarding non-functional requirements or this survey?

Thank you for taking our survey. Your response is very important to us.

Figure A.13: Statements about NFRs
Appendix B

Figures

Figure B.1: Boxplot of the correlation between the business criticality of availability and project success
Figure B.2: Boxplot of the correlation between the business criticality of performance and project success
Figure B.3: Boxplot of the correlation between the business criticality of security and project success
Figure B.4: Boxplot of the correlation between the business criticality of usability and project success
Figure B.5: Boxplot of the correlation between the business criticality of testability and project success
Figure B.6: Boxplot of the correlation between the application of elicitation and project success
Figure B.7: Boxplot of the correlation between the application of documentation and project success
Figure B.8: Boxplot of the correlation between the application of quantification and project success
Figure B.9: Boxplot of the correlation between the application of prioritization and project success
Figure B.10: Boxplot of the correlation between the application of conflict analysis and project success