Evaluating risk management methods for software projects

Saskia L. Woortman
2572363

Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
s.l.woortman@student.vu.nl

Abstract. This study evaluates the NEN guideline from 2019 for risk management during the development and maintenance of custom software. An overview is made of the coverage of the ten risks from this standard in the existing literature in the field of software risk management. As well as an overview of other risks identified in other literature that may lead to the failure of software projects. Failure of projects is evident by cancellation, going over-budget, late delivery, or less quality. To prevent project failure, organizations could use control measures to mitigate project risks. The coverage of control measures from the NEN standard is studied in the literature. As well as other potential control measures identified in the field. A quantitative survey is held to draw a conclusion of the research question: the extent to which the guideline is a useful risk assessment framework for organizations to identify and mitigate risks. This study results in a ranking of the risks most likely to occur in software projects, and a ranking of which measures are most used in practice. Consequently, thirteen project risks most likely to occur are identified. Three of them originate from the NEN standard. Second, 24 of the measures are considered best practices that could be used to mitigate the risks. Ten of the measures originate from the Dutch guideline. It can be concluded that especially the control measures from the standard are considered useful. However, essential risks and control measures are missing from it as well.

Keywords: Software risk management · Software development · Project management · Project risks · Control measures

1 Introduction

Managing software projects could be a difficult practice. Projects are at risk of failure due to cancellation of the project, going over-budget, late delivery, or less quality (Kwak & Stoddard, 2004). Both commercial companies and government bodies are prone to these risks (NEN, 2019). The Standish Group (2014) concluded from their research that 16,2% of the IT projects were successful, 31,1% were canceled, and 52,7% were either over-budget, late, or offered fewer functionalities than predetermined (Figure 1). This number could be reduced if

companies and governments would pay more attention to identifying and analyzing the possible risks in software projects (Boehm, 1991; Kwak & Stoddard, 2004; Schmidt et al., 2001).

The Royal Dutch Standardization Institute (Stichting Koninklijk Nederlands Normalisatie Instituut, NEN) initiated a project to develop a standard for the identification and mitigation of risks in software projects (NEN, 2019). This led to the Dutch guideline for risk management during the development and maintenance of custom software. This document consists of common risks in software development and corresponding measures in order to mitigate these risks. It especially focuses on the development of custom software since this can entail even more risks (Beekman, 2019; NEN, 2019).

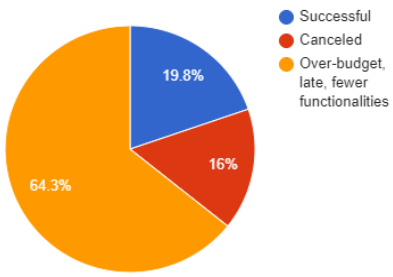


Fig. 1: Distribution of Successful Software Projects

This new standard should be analyzed and tested to determine its usefulness for companies and government organizations. The completeness of the assessment framework should be tested by identifying possible missing risks. Such a complete framework for risk assessment is important since the failure of software projects is an essential theme in project management literature. Moreover, the impact of failing projects can have a great impact on organizations. Mitigating the risks, and with that reducing failure of software projects, is essential in order to keep the faith of stakeholders and not to unnecessarily lose money.

2 Research Question

Thus, the standard has to be researched in order to test whether it is recommended to use the framework or if important aspects are missing. In order to research this problem, the following question will be studied:

To what extent is the guideline for risk management during development and maintenance of custom software a useful risk assessment framework for organizations to identify and mitigate risks?

The sub-questions established to help answer this main question are the following:

- How does the standard relate to the existing literature on risk management in software development projects?
- Are any important project risks missing from the standard?

– Which control measures are most used in practice?

By analyzing and testing the standard for software risk management, the guideline can be verified. The final result will consist of the most important risks within software projects, and the measures that are considered best practices to prevent or mitigate the risks. These results can be used by organizations to better identify risks in software development. Risk identification is the most important step in risk management (Roberts, 2001). However, it is often done poorly, and the project managers often overlook the greatest risk drivers. Hence why it is important to offer a guideline with the most occurring risks. When these risks are identified on time, organizations can act on it and mitigate them. In this way, long-term costs and software disasters could be prevented (Boehm, 1991).

3 Related Literature

3.1 Risk Management

Before defining what project risk management is, it is necessary to define the term risk itself. A risk is the product of uncertainty on the achievement of objectives, which can turn out to be negative, positive, or neutral (NEN, 2019). Although risks are often only conceived as negative. By applying control measures to identified risks, the chance the risk will occur or the outcome of the risk could be changed, or both. The identification and treatment of risks are both part of project risk management. ISO established the risk management process for general risk management, including the ongoing activities of Communication & Consultation, Monitoring & Reviewing, and Recording & Reporting (NEN-ISO, 2018; Purdy, 2010). While simultaneously executing the iterative process of 1) Establishing the Context, 2) Risk Assessment including Risk Identification, Risk Analysis and Risk Evaluation, and 3) Risk Treatment (Figure 2).

3.2 Software Risk Management

Risks specific to software projects can be defined as the uncertainty of risk factors and the potential loss due to the failure of the project (Schmidt et al., 2001). Software development itself has a risky nature (Boehm & DeMarco, 1997). Part of this problem is that the danger is not necessarily major calamities, but rather small day-by-day slippages (Brooks, 1995). From which it follows that the actual events deviate from the expected events as in the project plan (Abdel-Hamid, 1999). Thus, project managers should continuously stay on top of the current situation and adjust their staffing, estimates, resources, etc. accordingly. However, the concept of failure is often avoided in the field of information systems since acknowledging the risks in the project can be seen as defeatism (Boehm & DeMarco, 1997). Moreover, the strategies of risk management are often kept a secret as well since it can expose a company to legal liabilities (Boehm & DeMarco, 1997).

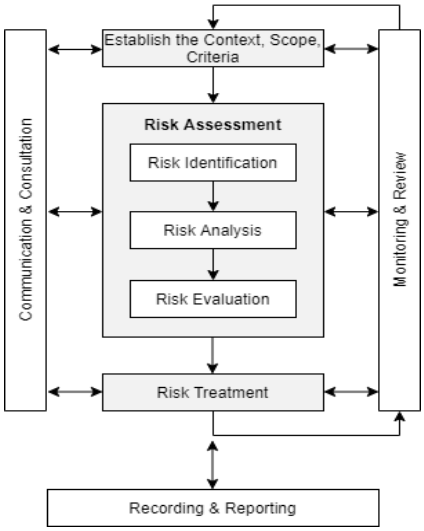


Fig. 2: Risk Management Process

Project managers have seven choices to deal with project risks (NEN-ISO, 2018). First, a risk can be prevented by not executing the activity that leads to the risk. Second, the risk can be accepted and nothing will be done, which is useful when the costs of other mitigation measures will be higher than the value. Third, the risk can be increased to pursue an opportunity. Fourth, a risk can be prevented by taking action so the risk will not occur. Fifth, the probability that the risk will occur can be reduced. Sixth, a risk can be controlled by reducing the impact of the risk. Last, the risk can be transferred to another party by outsourcing it or by buying insurance.

3.3 Project Risks

The Dutch guideline for risk management during the development and maintenance of custom software consists of ten potential risks. The coverage of these risks in the existing literature is shown in table 1. Other potential risks that are not covered in the guideline are shown in table 2. They are classified into six categories: project management, top-management, project team, users, development, and external risks.

Table 1: Coverage of Risks

ID	Risk	Coverage
Risk 01	The software changed, resulting in deteriorated quality of the software	Brooks, 1995; Schmidt et al., 2001
Risk 02	The software environment changed, resulting in deteriorated quality of the software	Brooks, 1995; Schmidt et al., 2001
Risk 03	The amount of work was not correctly estimated causing the planned functionality not to be completed on time	Abdel-Hamid et al., 1999; Addison & Vallabh, 2002; Boehm, 1991; Brooks, 1995; Han & Huang, 2007; Hoermann et al., 2012; Jones, 1998; Lehder et al., 1988; Sangaiah et al., 2017; Schmidt et al., 2001; Wallace et al., 2004a; Wallace et al., 2004b
Risk 04	Due to scope extensions, the product is not delivered on time and within budget	Boehm, 1991; Han & Huang, 2007; Hoermann et al., 2012; Keil et al., 1998; Paré et al., 2008; Sangaiah et al., 2017; Schmidt et al., 2001; Wallace et al., 2004b
Risk 05	The team does not have the right expertise, causing the software not to meet the requirements	Addison & Vallabh, 2002; Barki et al., 1993; Boehm, 1991; Hoermann et al., 2012; Keil et al., 1998; Paré et al., 2008; Schmidt et al., 2001; Sangaiah et al., 2017; Wallace et al., 2004a; Wallace et al., 2004b
Risk 06	Inadequate management of the work, causing the product to not offer the correct functionality	Barki, 1993; Brooks, 1995; Han & Huang, 2007; Hoermann et al., 2012; Schmidt et al., 2001; Sangaiah et al., 2017; Wallace et al., 2004a; Wallace et al., 2004b
Risk 07	Functional requirements are given too much priority so that the product lacks the correct non-functional properties	Boehm & In, 1996
Risk 08	The communication between stakeholders is suboptimal, causing misunderstandings	Brooks, 1995; Wallace et al., 2004a; Wallace et al., 2004b
Risk 09	Insufficient traceability of the development, use and management of customized software leads to no(non-demonstrable) compliance with obligations	Brooks, 1995; Sangaiah et al., 2017; Schmidt et al., 2001; Wallace et al., 2004a

Risk 10	Because a lot of time is required to meet the preconditions for software development, the product is not delivered on time	Barki et al., 1993

Table 2: Risks in the Existing Literature

ID	Risk	Coverage
Project management		
Risk 11	Lack of post-go-live approach	Hoermann et al., 2012; NEN, 2018
Risk 12	Poor relationship management	Addison & Vallabh, 2002; Keil et al., 1998; NEN, 2018; Schmidt et al., 2001
Risk 13	Gold plating: additional or more polished features than what was asked for or expected	Addison & Vallabh, 2002; Boehm, 1991; Lubelczyk & Parra, 1992
Risk 14	Project not based on a sound business case	Schmidt et al., 2001
Top-management		
Risk 15	Unstable corporate environment due to for instance organizational issues or changes in organizational management/environment	Brooks, 1995; Barki et al., 1993; Han & Huang, 2007; Paré et al., 2008; Schmidt et al., 2001; Wallace et al., 2004a
Risk 16	Mismatch between company culture and required business process needed for a new system	Schmidt et al., 2001
Risk 17	Bad estimated budget or underfunding	Abdel-Hamid et al., 1999; Addison & Vallabh, 2002; Hoermann et al., 2012; Sengupta & Swett, 1999; Schmidt et al., 2001
Risk 18	Insufficient resources	Addison & Vallabh, 2002; Barki et al., 1993; Paré et al., 2008; Ropponen & Lyytinen, 2000; Wallace et al., 2004a
Risk 19	Lack of top management commitment to the project	Addison & Vallabh, 2002; Barki et al., 1993; Gioia, 1996; Keil et al., 1998; Paré et al., 2008; Schmidt et al., 2001

Risk 20	Low project priority	Hoermann et al., 2012; Schmidt et al., 2001
Risk 21	Insufficient staffing	Barki et al., 1993; Brooks, 1995; Hoermann et al., 2012; Keil et al., 1998; Lehder et al., 1988; NEN, 2018; Sangaiah et al., 2017; Schmidt et al., 2001
Project team		
Risk 22	Dependency on a few key people	Barki et al., 1993; Hoermann et al., 2012
Risk 23	Low morale in the team, which manifests in for instance the lack of project member's commitment to the project plan or not reporting possible risks	Sangaiah et al., 2017
Risk 24	Incorrect or optimistic status reporting	Jones, 1998
Users		
Risk 25	Unrealistic expectations of the users	Hoermann et al., 2012; Paré et al., 2008; Schmidt et al., 2001
Risk 26	Lack of user acceptance and therefore cooperation	Hoermann et al., 2012; Paré et al., 2008; Schmidt et al., 2001; Wallace et al., 2004a
Risk 27	Conflict between users or user departments	Keil et al., 1998; Schmidt et al., 2001; Wallace et al., 2004a
Risk 28	Poor perceived usability	Paré et al., 2008
Development		
Risk 29	Availability of testing infrastructure	Hoermann et al., 2012
Risk 30	Relative large project size, compared to other internal projects	Barki et al., 1993; Hoermann et al., 2012; Paré et al., 2008
Risk 31	Technical complexity which may be due to the use of a new technology or straining computer-science capabilities	Addison & Vallabh, 2002; Barki et al., 1993; Boehm, 1991; Han & Huang, 2007; Hoermann et al., 2012; Keil et al., 1998; Mursu et al., 1996; Paré et al., 2008; Schmidt et al., 2001; Wallace et al., 2004a

Risk 32	Lack of effective development process/methodology	Schmidt et al., 2001
Risk 33	The project wants to release new versions of the software more often than the management organization/users want or can handle	NEN, 2018
Risk 34	Dependencies on other projects	Barki et al., 1993; Hoermann et al., 2012
Risk 35	Number of organizational units involved or affected	Barki et al., 1993; Hoermann et al., 2012; Schmidt et al., 2001
External risks		
Risk 36	Subcontracting and external partners	Addison & Vallabh, 2002; Barki et al., 1993; Boehm, 1991; Hoermann et al., 2012; Paré et al., 2008; Ropponen & Lyytinen, 2000; Schmidt et al., 2001
Risk 37	Legal changes	Hoermann et al., 2012

3.4 Measures

The probability project risks occur, or the impact they have can be reduced by using control measures. The Dutch guideline proposes seventeen measures that can be used against the ten risks. The coverage of these risks in the existing literature in the field of risk management and project evaluations, are shown in table 3. Other measures that are suggested in the existing literature are shown in table 4. These measures are categorized into the following groups: organizational, planning and vision, communication, and development.

Table 3: Coverage of Control Measures

ID	Control measure	Coverage
Measure 01	Identify and involve stakeholders	Addison & Vallabh, 2002; Audit Schotland, 2017; Project team C2000, 2006; Standish Group, 2014
Measure 02	Identify important non-functional requirements	Addison & Vallabh, 2002; Boehm & Ross, 1989; Project team C2000, 2006; Standish Group, 2014

Measure 03	Identify important functional requirements	Addison & Vallabh; Boehm & Ross, 1989; Project team C2000, 2006; Standish Group, 2014
Measure 04	Product decomposition in incrementally deliverable parts with business value	Addison & Vallabh, 2002; Lubelczyk & Parra, 1992; Project team C2000, 2006
Measure 05	Identify technical debt, provide insights and solve it according to plan	
Measure 06	Explore possible solutions, which includes prototyping	Boehm & Ross, 1989; Gemmer, 1997
Measure 07	Incremental delivery of the product	Boehm & Ross, 1989; McLeod & Smith, 1996; Project team C2000, 2006
Measure 08	Iterative development	Schwaber & Sutherland, 2017
Measure 09	Set up an automated development pipeline	
Measure 10	Constantly meet the requirements using regression tests	
Measure 11	Monitor progress using burndown charts	Audit Schotland, 2017; Lubelczyk & Parra, 1992; Project team C2000, 2006; Schwaber & Sutherland, 2017
Measure 12	An official product owner with a mandate	Schwaber & Sutherland, 2017; Standish Group, 2014
Measure 13	Apply a quality-driven development method	
Measure 14	Archiving the documents and code after the development or the project is finished	
Measure 15	Sound transfer to the customers or another team	Audit Schotland, 2017; Boehm & Ross, 1989
Measure 16	Support the teams by providing specialist knowledge and resources	Boehm & DeMarco, 1997; Standish Group, 2014
Measure 17	Continuous risk management	Addison & Vallabh, 2002; Audit Schotland, 2017; Higuera et al., 1994

Table 4: Control Measures in the Existing Literature

ID	Control Measure	Coverage
Organizational		
Measure 18	Recognise the role of culture and tone at the top	Audit Schotland, 2017
Measure 19	Maintain stability in leadership and develop succession planning	Audit Schotland, 2017
Measure 20	Teambuilding	Boehm & DeMarco, 1997; Boehm & Ross, 1989
Measure 21	Recognizing outstanding efforts	Boehm & DeMarco, 1997
Measure 22	Spread knowledge of product components among various people	Boehm & DeMarco, 1997
Measure 23	Develop contingency plans to cope with staffing problems	Addison & Vallabh, 2002; Boehm & Ross, 1989
Measure 24	Key-personnel agreements	Boehm & Ross, 1989
Measure 25	Structuring career paths around an organizations product lines	Boehm & DeMarco, 1997
Measure 26	Cross-training: teach (some) skills or knowledge required to perform other job function	Boehm & Ross, 1989
Measure 27	Organization analysis	Boehm & Ross, 1989
Measure 28	Inspections at external companies	Boehm & Ross, 1989
Planning and vision		
Measure 29	Clear and consistent distribution of responsibilities	Audit Schotland, 2017; Lubelczyk & Parra, 1992; Project team C2000, 2006
Measure 30	Shared product vision	Higuera et al., 1994; Standish Group, 2014
Measure 31	Clearly define the needs and benefits by for example carrying out user surveys, making user characterizations or scenarios	Audit Schotland, 2017; Boehm & Ross, 1989

Measure 32	Smaller, and if possible measurable, project milestones	Standish; Addison & Vallabh; Audit Schotland, 2017; Project team C2000, 2006
Measure 33	Choose one appropriate project management method	Audit Schotland, 2017; Project team C2000, 2006
Measure 34	Proactive strategies; that involve planning and executing program activities based on anticipating future events	Gemmer, 1997; Higuera et al., 1994
Measure 35	Make timely, well-informed decisions and commitments	Gemmer, 1997
Measure 36	Add a list of all known potential and relevant risks to the software project plan	Lubelczyk & Parra, 1992
Measure 37	Proper planning by understanding and appreciating the likely complexity	Brooks, 1995; Project team C2000, 2006; Standish Group, 2014
Measure 38	Assess cost and schedule impact of each change to requirements and specifications	Addison & Vallabh, 2002; Keil et al., 1998
Measure 39	Be aware of optimism bias	Audit Schotland, 2017
Measure 40	Cost-benefit analysis	Boehm & Ross, 1989
Measure 41	Detailed multisource cost and schedule estimation	Boehm & Ross, 1989; Brooks, 1995
Measure 42	Design to cost	Boehm & Ross, 1989; Project team C2000, 2006
Communication		
Measure 43	Involve management during the entire project lifecycle	Addison & Vallabh, 2002; Audit Schotland, 2017; Project team C2000, 2006
Measure 44	Seek diversity in perspectives and information sources	Gemmer, 1997; Higuera et al., 1994
Measure 45	Create a pull for risk information	Gemmer, 1997
Measure 46	Reward those who identify and manage risks early, even if the risks become problems	Gemmer, 1997
Measure 47	Recognize and minimize bias in perceiving risk	Gemmer, 1997

Measure 48	Avoid blame	Lunney & Lueder, 2017
Measure 49	Make clear agreements in advance on the financial distribution	Project team C2000, 2006
Development		
Measure 50	Requirements scrubbing: carefully examining for unnecessary or overly complex requirements	Boehm & Ross, 1989
Measure 51	Software reuse	Boehm & Ross, 1989
Measure 52	Benchmarking: evaluating by comparison with an internal or external standard	Boehm & Ross, 1989
Measure 53	Instrumentation to measure the product's performance	Boehm & Ross, 1989
Measure 54	Software tuning	Boehm & Ross, 1989
Measure 55	Good configuration management	Boehm & DeMarco, 1997
Measure 56	Build in appropriate quality assurance processes	Audit Schotland, 2017
Measure 57	Develop appropriate independent assurance arrangements	Audit Schotland, 2017; Cule et al., 2000
Measure 58	Work within a central assurance framework	Audit Schotland, 2017
Measure 59	Use egoless programming	Boehm & DeMarco, 1997; Brooks, 1995
Measure 60	Realistic expectations of the project team for the final product	Standish Group, 2014

4 Research Method

The outcome of the literature study, the tables of risks and measures, is tested through quantitative research. An online survey is carried out to draw conclusions from a larger sample size by combining the insights from experts to a consensus on the likelihood of risks and measures. Different set-ups for this questionnaire are explored. Amongst them, to let participants rank the risks. However, due to the number of risks, there would not have been a clear overview for the participants. Another option was in the format of a matrix, to let the par-

ticipants map the control measures to the risks they influence. However, due to the number of measures and risks, this was not feasible. Even with a predefined subset of risks each measure could affect it would have taken too much time. So it was chosen to reject this format and to continue with a survey consisting of Likert scale-based questions and open text entries. The question on the impact each risk has is left out to reduce the time to complete the survey. The impact of a risk is more dependent on the project itself than the probability it would occur in a project.

The final survey consists of three parts (Appendix 1). First, the probability that each risk would occur. The risks are shown in the six predefined categories for a clearer overview. The answers are defined on the generally accepted method of a 5-point Likert scale with the labels *Almost in all projects, In most projects, Neutral, Sometimes,* and *Almost never* (Burns & Burns, 2008). Each risk has an optional text entry next to the multiple-choice options. This text entry could be used for the argumentation of a particular choice, whether the risk is not clear or any other comment. Below the listing of risks, there is also the possibility to indicate a risk the respondent is missing from the survey. The second part consists of questions on the control measures. Again, the measures are listed in the four categories for more overview on the page. The question comprises of three parts: if the respondent uses the measure, whether it is specific enough to apply it, and the optional text entry. The respondent has the opportunity to add on to the list of control measures as well. In the last part, two demographic questions are asked. First, the job description of the respondent and second the size and type of the organization. The size and type are defined as *Small enterprise: 1 to 49 employees, Medium enterprise: 50 to 249 employees, Large enterprise: 250 employees or more,* and *Government organization* (OECD, 2020).

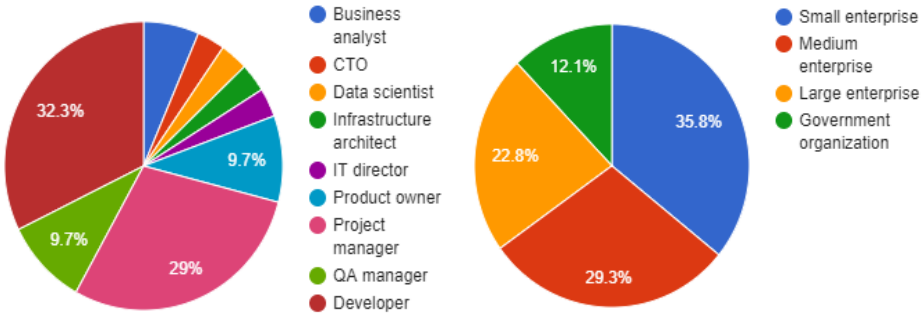
The target group of this research consists, broadly speaking, of people working on software projects. Thus, the survey is spread under IT project managers, product owners, developers, etcetera. The participants were reached online, mostly via LinkedIn.

5 Results

In order to draw conclusions from the survey, the results from the 31 responses are analyzed by using SPSS. More specifically, one sample t-tests are used to compare the means of the risks and measures and to examine the significance. The risks and measures are ranked by importance using the mean of each item. To be able to do this, the answers are converted to number-based answers. For the first part about the risks, the answers are as follows, 1: *Almost never, 2: Sometimes, 3: Neutral, 4: In most projects,* and *5: Almost in all projects.* For the second part about the measures, *Yes: 1* and *No: 2.* All outcomes of the means are significant with $p < .001$.

5.1 Participants

At last, 31 participants completed the survey. Of these participants 25.7 percent are project managers, 28.6 percent are developers, and 34.3 percent have another job description, including a CTO, QA managers, and product owners (Figure 3a). Of the participants, 31.4 percent works for a small enterprise, 25.7 percent for a medium enterprise, 20 percent for a large enterprise, and 11.4 percent for a Dutch government organization (Figure 3b).



(a) Distribution of Jobs

(b) Distribution of Organizations

Fig. 3: Background of the Participants

5.2 Project Risks

To give an overview, the overall mean of the risks is 2.795. the mean of the risks per category is presented in table 5. The differences are minimal, especially when taking into consideration the least scoring category of external risks consists of only two items. Next, the individual risks are ranked on their mean, 5 being the most likely to occur and 1 the least. Then the risks are categorized into three groups: the most occurring, average occurring, and least occurring risks. The difference between these groups is determined by the difference between the two risks on the boundary between two groups. So whether it would be possible for a risk to be part of another group with one more response deviating from the average. In this way, the ranking below has been established (Table 6). The complete overview including the amount of each answer given and the standard deviation can be found in appendix 2.

Risk category	**Mean**	**Std. deviation**	**Number of risks**
Project management	2.848	1.204	10
Project team	2.831	1.292	4
Users	2.826	1.140	5
Top-management	2.820	1.167	7
Development	2.753	1.249	9
External risks	2.694	1.338	2

Table 5: Mean of Risk Categories

Most occurring risks

Risk 22	Dependency on a few key people	4.000
Risk 03	The amount of work was not correctly estimated causing the planned functionality not to be completed on time	3.935
Risk 08	The communication between stakeholders is suboptimal, causing misunderstandings	3.580
Risk 04	Due to scope extensions, the product is not delivered on time and within budget	3.419
Risk 18	Insufficient resources	3.322
Risk 29	Availability of testing infrastructure	3.258
Risk 25	Unrealistic expectations of the users	3.225
Risk 35	Number of organizational units involved or affected	3.129
Risk 36	Subcontracting and external partners	3.064
Risk 17	Bad estimated budget or underfunding	3.032
Risk 34	Dependencies on other projects	3.000
Risk 11	Lack of post-go-live approach	3,000

Risk 24	Incorrect or optimistic status reporting	2.935

Average occurring risks

Risk 31	Technical complexity which may be due to the use of a new technology or straining computer-science capabilities	2.838
Risk 21	Insufficient staffing	2.838
Risk 01	The software changed, resulting in deteriorated quality of the software	2.806
Risk 20	Low project priority	2.742
Risk 07	Functional requirements are given too much priority so that the product lacks the correct non-functional properties	2.710
Risk 30	Relative large project size, compared to other internal projects	2.710
Risk 06	Inadequate management of the work, causing the product to not offer the correct functionality	2.645
Risk 13	Gold plating: additional or more polished features than what was asked for or expected	2.645
Risk 14	Project not based on a sound business case	2.645
Risk 26	Lack of user acceptance and therefore cooperation	2.645
Risk 12	Poor relationship management	2.613
Risk 15	Unstable corporate environment due to for instance organizational issues or changes in organizational management/environment	2.613
Risk 16	Mismatch between company culture and required business process needed for a new system	2.631
Risk 19	Lack of top management commitment to the project	2.581

Least occurring risks

Risk 02	The software environment changed, resulting in deteriorated quality of the software	2.484
Risk 28	Poor perceived usability	2.452
Risk 32	Lack of effective development process/methodology	2.452

Risk 09	Insufficient traceability of the development, use and management of customized software leads to no(non-demonstrable) compliance with obligations	2.387
Risk 27	Conflict between users or user departments	2.387
Risk 10	Because a lot of time is required to meet the preconditions for software development, the product is not delivered on time	2.323
Risk 37	Legal changes	2.323
Risk 05	The team does not have the right expertise, causing the software not to meet the requirements	2.194
Risk 23	Low morale in the team, which manifests in for instance the lack of project member's commitment to the project plan or not reporting possible risks	2.194
Risk 33	The project wants to release new versions of the software more often than the management organization/users want or can handle	2.097

Table 6: Ranked Results of the Most Occurring Risks
1: Almost never, 2: Sometimes, 3: Neutral, 4: In most projects, 5: Almost in all projects

To reference back to the Dutch guideline for software risk management from the NEN, the ranking of the ten risks is shown in table 7. Three out of the ten risks from the Dutch guideline are among the risks with the highest likelihood to occur (risks 3, 4 & 8). In the average occurring risks, three risks from the guidelines can be found as well (risks 1, 6 & 7). Four of the ten risks are part of the risks with the lowest likelihood of occurrence (risks 2, 5, 9 & 10).

Comments on the Risks

Participants had the opportunity to comment on the risks and their answers as well (Appendix 3). These comments offer an explanation or point of discussion for some of the outcomes. First, several risks are, among some of the respondents, seen as a fact of life in software development rather than a real risk (risks 3 & 8). The impact of these risks depend mostly on the project and the company. One of the companies characteristic affecting this is, for instance, the distinction between asset-based environments and pure IT firms. For asset-based companies the minimum viable product (MVP) of a product might need to be of a higher standard since the product needs to meet all the requirements not to bring disruption to the main processes. Whereas pure IT firms, especially start-ups, could offer more flexibility. Second, some risks are more of a hindrance in the beginning phase of the project, and are more likely to fade away when the deadline is near (risks 13 & 18). Towards the end of the project, the team is more likely to receive the additional resources needed since spending more on it

ID	Risk	Ranking	Group
Risk 01	The software changed, resulting in deteriorated quality of the software	16	Average occurring
Risk 02	The software environment changed, resulting in deteriorated quality of the software	28	Least occurring
Risk 03	The amount of work was not correctly estimated causing the planned functionality not to be completed on time	2	Most occurring
Risk 04	Due to scope extensions, the product is not delivered on time and within budget	4	Most occurring
Risk 05	The team does not have the right expertise, causing the software not to meet the requirements	35 ex aequo	Least occurring
Risk 06	Inadequate management of the work, causing the product to not offer the correct functionality	20 ex aequo	Average occurring
Risk 07	Functional requirements are given too much priority so that the product lacks the correct non-functional properties	18 ex aequo	Average occurring
Risk 08	The communication between stakeholders is suboptimal, causing misunderstandings	3	Most occurring
Risk 09	Insufficient traceability of the development, use and management of customized software leads to no(non-demonstrable) compliance with obligations	31 ex aequo	Least occurring
Risk 10	Because a lot of time is required to meet the preconditions for software development, the product is not delivered on time	33	Least occurring

Table 7: Ranking of the NEN Risks

would be better than a canceled project. However, by definition projects that go over budget are considered unsuccessful as well (Kwak & Stoddard, 2004).

A discussion point revolves around the risk of technical complexity (risk 31). A participant mentions a solution can always be found to make it work. There is just no guarantee of finding the best one. However, finding another solution could take more time than expected, causing the project to be delivered late or over budget. Another discussion point is the project not being the main priority (risk 20). On the one hand, it could be a risk due to, for instance, team members needed on other projects with a higher priority. While a participant argued, it can be a benefit as well since it takes away management pressure and over-meddling by stakeholders. Another comment is made on the risk of legal changes (risk 37), which is part of the least likely risks. However, it is mentioned this risk could have a very disruptive influence when it occurs. To illustrate, Brexit or GDPR are mentioned.

Finally, multiple participants mention two risks to be unclear. First, the risk of a relatively large project size (risk 30). The risk being unclear could explain the number of neutral answers, which is 35.34 percent, resulting in it to be part of the average occurring risk category. Another participant highlights this risk often has the same cause as the incorrect estimation of the amount of work (risk 03). Thus in further research, risk 30 could be merged into risk 3. Second, the lack of a post-go-live approach (risk 11) is now categorized as the most occurring risks. This could be explained by the number of neutral answers, 25.81 percent. The risk should be tested with another label, to validate that the risk is part of the first group. A suggested label including a clearer description would be: *Lack of post-go-live approach: the maintenance plan for after releasing the product.*

Missing Risks

The participants highlighted a few missing risks (Appendix 3). First, delays due to the time taken by customers to make a decision. As well as delays due to the time taken internally to come to a decision. These risks cause the goal to be unclear and a delay in the project. Second, insufficient time allocation for testing. It could be considered part of the incorrect estimation of the amount of work (risk 3), but it is worth to be mentioned separately. Brooks (1995) states that testing is often the most wrongly scheduled part of the planning. He proposes planning half of the time for testing to debug and complete the code. Especially in traditional software development methodologies when the project already faces delays, testing of the product might not be the main priority when the deadline approaches. In Agile working, quality assurance (QA) is integrated in every step of the development process, so this might be less of a problem. Third, not putting enough research into the right technology, which may lead to choosing the wrong technology. Last, a mismatch between sales and development visions. The alignment of the sales team and the development team is of paramount importance. It is a problem if the sales team sells a dream that cannot be built. Another side of this problem could be that opportunities are left unused. The underlying problem of this could be that the sales team does not fully comprehend the development vision's possibilities.

5.3 Control Measures

For the second part, the control measures are ranked on their mean, 1 being the best measure and 2 being the least effective measure. The three groups that cover the measures are: the best practices, the average practices, and the least effective practices. Resulting in the ranking below (Table 8). A complete overview of the distributions of answers can be found in appendix 4.

Best practices

Measure 03	Identify important functional requirements	1.000
Measure 32	Smaller, and if possible measurable, project milestones	1.000
Measure 60	Realistic expectations of the project team for the final product	1.000
Measure 02	Identify important non-functional requirements	1.032
Measure 07	Incremental delivery of the product	1.032
Measure 01	Identify and involve stakeholders	1.032
Measure 05	Identify technical debt, provide insights and solve it according to plan	1.032
Measure 08	Iterative development	1.064
Measure 51	Software reuse	1.097
Measure 20	Teambuilding	1.100
Measure 22	Spread knowledge of product components among various people	1.129
Measure 04	Product decomposition in incrementally deliverable parts with business value	1.129
Measure 12	An official product owner with a mandate	1.129
Measure 06	Explore possible solutions, which includes prototyping	1.129
Measure 56	Build in appropriate quality assurance processes	1.129
Measure 35	Make timely, well-informed decisions and commitments	1.133
Measure 29	Clear and consistent distribution of responsibilities	1.161

Measure 37	Proper planning by understanding and appreciating the likely complexity	1.167
Measure 16	Support the teams by providing specialist knowledge and resources	1.194
Measure 30	Shared product vision	1.194
Measure 43	Involve management during the entire project lifecycle	1.194
Measure 48	Avoid blame	1.194
Measure 21	Recognizing outstanding efforts	1.200
Measure 55	Good configuration management	1.200

Average practices

Measure 31	Clearly define the needs and benefits by for example carrying out user surveys, making user characterizations or scenarios	1.226
Measure 33	Choose one appropriate project management method	1.226
Measure 14	Archiving the documents and code after the development or the project is finished	1.226
Measure 50	Requirements scrubbing: carefully examining for unnecessary or overly complex requirements	1.226
Measure 10	Constantly meet the requirements using regression tests	1.226
Measure 09	Set up an automated development pipeline	1.258
Measure 15	Sound transfer to the customers or another team	1.267
Measure 11	Monitor progress using burndown charts	1.290
Measure 44	Seek diversity in perspectives and information sources	1.300
Measure 53	Instrumentation to measure the product's performance	1.322
Measure 19	Maintain stability in leadership and develop succession planning	1.333
Measure 54	Software tuning	1.345
Measure 26	Cross-training: teach (some) skills or knowledge required to perform other job function	1.355

Measure 36	Add a list of all known potential and relevant risks to the software project plan	1.355
Measure 13	Apply a quality-driven development method	1.357
Measure 23	Develop contingency plans to cope with staffing problems	1.357
Measure 17	Continuous risk management	1.400
Measure 40	Cost-benefit analysis	1.400
Measure 38	Assess cost and schedule impact of each change to requirements and specifications	1.419
Measure 27	Organization analysis	1.433
Measure 34	Proactive strategies; that involve planning and executing program activities based on anticipating future events	1.433

Least effective practices

Measure 49	Make clear agreements in advance on the financial distribution	1.448
Measure 52	Benchmarking: evaluating by comparison with an internal or external standard	1.484
Measure 18	Recognise the role of culture and tone at the top	1.500
Measure 59	Use egoless programming	1.520
Measure 39	Be aware of optimism bias	1.548
Measure 47	Recognize and minimize bias in perceiving risk	1.571
Measure 57	Develop appropriate independent assurance arrangements	1.586
Measure 58	Work within a central assurance framework	1.633
Measure 46	Reward those who identify and manage risks early, even if the risks become problems	1.645
Measure 25	Structuring career paths around an organizations product lines	1.667
Measure 45	Create a pull for risk information	1.690
Measure 42	Design to cost	1.710

Measure 24	Key-personnel agreements	1.714
Measure 28	Inspections at external companies	1.733
Measure 41	Detailed multisource cost and schedule estimation	1.742

Table 8: Ranked Results of the Most Effective Measures
1: Yes, 2: No

To reference back to the control measures suggested by the Dutch guidelines, the overview of their ranking is shown in table 9. Ten out of the seventeen control measures are part of the best practices (measures 1 up to 8, 12 & 16). While seven are part of the average practices (measures 9, 10, 11, 13, 14, 15, 17), and there are none in the least effective practices.

Comments on the Measures

From the comments given by the respondents, some advises arise as well (Appendix 5). First, several measures are not generally advised but only for large and complex projects (measures 6, 17 & 31). Second, some measures are advised when hard important decisions need to be made (measures 40 & 43). Implementing it continuously could stifle the process however. Related to the development risks, multiple measures exist but they may not always be the best choice. Their usefulness may depend on how they are implemented (measures 10 & 51). Using Regression tests is for instance considered more useful when they can automated or when it is only risk-based. Software reuse is commented on to be better when using libraries. However the choice to implement this depends on the goals of the project and it is important to acknowledge the risks of doing so. When the organization subcontracts a part of the project to another supplier it might be useful to check the quality. Measure 52, benchmarking, may be taken into consideration as a good practice to do this. For internal use, it could lead to trust issues which may be why it is considered one of the least effective measures. Last, the availability of time and budget is important in order to incorporate some of the measures (measures 26 & 44).

For the control measures, some discussion points arise as well. First of all, whether or not to avoid blame (measure 48). It is important to communicate mistakes so they will not be repeated. On the other hand, blaming could create a hostile environment. For instance, when it is used as a pressure tool on project members. It is important to communicate without crossing the line to blaming incorrectly to avoid these kinds of situations. However, the capability of employees to accept blame is also important in how it is received. Second, choosing one project management method (measure 33) might not be the best practice for all projects. Some argue using a mix of project management methods specified for a project could be better. Third, maintaining stability in leadership (measure 19)

ID	Measure	Ranking	Group
Measure 01	Identify and involve stakeholders	4 ex aequo	Best practice
Measure 02	Identify important non-functional requirements	4 ex aequo	Best practice
Measure 03	Identify important functional requirements	1 ex aequo	Best practice
Measure 04	Product decomposition in incrementally deliverable parts with business value	11 ex aequo	Best practice
Measure 05	Identify technical debt, provide insights and solve it according to plan	4 ex aequo	Best practice
Measure 06	Explore possible solutions, which includes prototyping	11 ex aequo	Best practice
Measure 07	Incremental delivery of the product	4 ex aequo	Best practice
Measure 08	Iterative development	8	Best practice
Measure 09	Set up an automated development pipeline	30	Average practice
Measure 10	Constantly meet the requirements using regression tests	25 ex aequo	Average practice
Measure 11	Monitor progress using burndown charts	32	Average practice
Measure 12	An official product owner with a mandate	11 ex aequo	Best practice
Measure 13	Apply a quality-driven development method	39	Average practice
Measure 14	Archiving the documents and code after the development or the project is finished	25 ex aequo	Average practice
Measure 15	Sound transfer to the customers or another team	31	Average practice
Measure 16	Support the teams by providing specialist knowledge and resources	19	Best practice
Measure 17	Continuous risk management	41 ex aequo	Average practice

Table 9: Ranking of the NEN Control Measures

is argued to be a good strategy. However in this research, it is combined with developing succession planning which is highlighted not to be a best practice. Fourth, monitoring progress (meaure 11) is in the results part of the average effective practices. However, it is mentioned by some respondents monitoring progress is a best practice, but burndown charts are not the preferred manner. Fifth, recognizing outstanding efforts (measure 21) is considered to be a best practice. However when implementing it, the management should be aware of the competition that could arise from it.

Implementation of the Measures

Apart from the discussion points, not all measures are as easy to implement successfully. First in the group of best practices, the question how to achieve it applies to the measures 30, 35, 37, 55, and 60. For the average effective measures some ambiguities arise as well (measures 15, 19, 27 & 34). Last, in the category of least effective practices, six unclear measures are recognized (measures 18, 39, 45, 47, 59 & 49). Actual guidelines containing action steps on how to achieve these measures should be specified. For some measures, for instance measure 18, it is important to specify how to proceed from a measure as well.

The guideline for risk management does offer guidance on how to facilitate a sound transfer (measure 15). It includes providing documentation on the used development and testing environment, processes, recovery procedures, possible shortcomings, and version history (NEN, 2019). Last, making clear agreements on the financial distribution (measure 49) needs a new description as it was not clear to all respondents which financial distribution is meant. This measure originates from a project evaluation (Project team C2000, 2006) involving a project anticipated by the Dutch government, which would be used by several separate institutions. Thus the main problem addressed by this measure is how the payment will be distributed when multiple parties are involved, all with their own budgets.

Other Measures

One participant mentions another control measure that could be used in order to facilitate better and open communication (Appendix 5). The measure is to organize weekly meetings with at least one representative from each involved department during the development phase of the product. In this way, all the latest progress could be shared, and possible complications are discussed. By stating at least one employee from each team should be present, the problem of miscommunication is addressed.

6 Conclusion

The main starting point of this research is the guideline for risk management during the development and maintenance of custom software issued by the NEN (2019). To draw a conclusion on its usability, it is essential to answer the sub-questions. The first subquestion entails how the standard relates to the existing

literature on risk management in software development projects. All ten risks from the guideline are covered in the existing literature, and are thus acknowledged in the field (Table 1). However, the literature study identified 27 more risks (Table 2). This fact, could lead to the conclusion that the standard is not complete. Most of the control measures from the standard are covered by the existing literature on risk management (Table 3). For five out of seventeen, no verification is found. In the same literature 43 other measures are identified (Table 4). This could lead to the same conclusion, thus that some important control measures might be missing from the guideline. To answer the first subquestion, most risks and measures from the standard are acknowledged in the literature. However, many more are as well.

This leads to the second subquestion, whether any important risks are missing from the standard. The quantitative research results in a ranking of the risks with the highest probability to occur during a software project (Table 6). The most occurring risks consists of three risks from the standard (risks 3, 4 & 8), and ten risks from other risk management literature (risks 11, 17, 18, 22, 24, 25, 29, 34, 35 & 36). Thus to answer the subquestion, there are risks that are likely to occur missing from the guideline. The risks with the highest probability to occur in software projects are:

1. Dependency on a few key people
2. The amount of work was not correctly estimated causing the planned functionality not to be completed on time
3. The communication between stakeholders is suboptimal, causing misunderstandings
4. Due to scope extensions, the product is not delivered on time and within budget
5. Insufficient resources
6. Availability of testing infrastructure
7. Unrealistic expectations of the users
8. Number of organizational units involved or affected
9. Subcontracting and external partners
10. Bad estimated budget or underfunding
11. Dependencies on other projects
12. Lack of post-go-live approach: the maintenance plan for after releasing the product
13. Incorrect or optimistic status reporting

The last subquestion relates to the control measures, namely which measures are most used in practice. The complete ranking of the most effective measures is shown in Table 8. From the best practices ten originate from the NEN guideline (measures 1 up to 8, 12 & 16), and the remaining fourteen from other existing literature (measures 20, 21, 22, 29, 30, 32, 35, 37, 43, 48, 51, 55, 56 & 60). The best control measures to prevent or mitigate software project risks are:

1. Identify important functional requirements
2. Smaller, and if possible measurable, project milestones

3. Realistic expectations of the project team for the final product
4. Identify important non-functional requirements
5. Incremental delivery of the product
6. Identify and involve stakeholders
7. Identify technical debt, provide insights and solve it according to plan
8. Iterative development
9. Software reuse
10. Teambuilding
11. Spread knowledge of product components among various people
12. Product decomposition in incrementally deliverable parts with business value
13. An official product owner with a mandate
14. Explore possible solutions, which includes prototyping
15. Build in appropriate quality processes
16. Make timely, well-informed decisions and commitments
17. Clear and consistent distribution of responsibilities
18. Proper planning by understanding and appreciating the likely complexity
19. Support the teams by providing specialist knowledge and resources
20. Shared product vision
21. Involve management
22. Avoid blame
23. Recognizing outstanding efforts
24. Good configuration management

To answer the main research question, the extent to which the guideline for risk management established by the NEN is a useful risk assessment framework for organizations to identify and mitigate risks. The standard is partly a useful assessment framework, since three out on ten risks are part of the most occurring risks and ten out of seventeen of the control measures are part of the best practices. Thus especially the control measures are considered useful. However, it is not complete since ten risks likely to occur and fourteen measures considered best practices are missing from the framework.

Overall, when working on software development projects it is good to be attentive towards project risks. This research provides guidance on which risks are likely to occur and which control measures are useful to prevent and mitigate these risks. First of all, the project management and other team-members should be aware of the thirteen risks mentioned above. Second, it is good to take into consideration the environment of the company and the type of project. Due to these aspects average and least occurring risks could be important as well. These aspects could also raise other risks, specific to the organization or project. After identifying the possible risks in a project, analyzing and evaluating them, the risks could be treated by using control measures. So third, the best practices mentioned above offer a guidance on how to treat the risks. A suggestion of which best practices could be used for the most occurring risks can be found in appendix 6. Besides, specific aspects of the environment and the project type could influence if a control measure is a good fit and how it works out. For example for large and/or complex projects it is advised to explore the

possible solutions, define the needs and benefits, and perform continuous risk management (measures 6, 17 & 31). Last, it is stressed open communication is very important among the stakeholders involved. Thus, the recommendation is to use the lists above, containing what is most common in practice, while being aware of other company and project-related risks and control measures as well.

7 Discussion

The basis of this research is the NEN guideline published in 2019. This research also covers a variety of literature in the field of risk management related to software projects. Most of this literature is relatively old: out of the 26 articles eleven are published before 2000, eight in 2000 until 2009, and seven are from 2010 until 2018. Both, the NEN guideline and literature in the field, are the foundation for the quantitative research where all aspects are tested within the same study to draw the conclusion. Hereby, updating the literature in the field into one contemporary list of the most important risks and control measures.

A limitation, however, is the small sample size. In some cases, there are not enough participants to compare different groups. Causing it not to be possible to draw a conclusion whether a risk is for instance more important for government organizations. For further implications of this, additional research is needed. In this way, the distinction of the circumstances under which risks are more likely to occur can be further specified. The same applies to the control measures, thus for instance when it is best to apply which measure. These distinctions could also be used to offer a few different matrices for specific situations instead of just one like the suggested matrix in appendix 6. These different matrices would contain the most important risks for a specific condition and corresponding control measures to mitigate these risks. For instance, one for large and complex projects, and another one for start-ups, etcetera. These different matrices could offer an even better and structured guideline. Another aspect to offer a better guideline, and that needs further research, are the action steps on how to implement the hard to accomplish measures (measures 18, 19, 27, 30, 34, 35 37, 39, 45, 47, 55, 59 & 60). Now some of these are not real practices that could be implemented, but rather something that should be achieved. Last, in further research the list of risks could be updated. This includes incorporating the newly identified risks to the list. As well as changing some of the names, like suggested in the comments on risks and measures sections. So amongst this is to monitor progress, not specifically using burndown charts (measure 11). Finally, some of the risks and measures could be left out. This makes it possible to test the impact of risks as well.

8 References

Abdel-Hamid, T. K., Sengupta, K., & Swett, C. (1999). The impact of goals on software project management: An experimental investigation. MIS quarterly, 531-555.

Addison, T., & Vallabh, S. (2002). Controlling software project risks: an empirical study of methods used by experienced project managers. In Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology , 128-140. South African Institute for Computer Scientists and Information Technologists.

Audit Schotland, 2017. Principles for a digital future: Lessons learned from public sector ICT projects.

Barki, H., Rivard, S., & Talbot, J. (1993). Toward an assessment of software development risk. Journal of management information systems, 10(2), 203-225.

Beekman, T. (2019). NPR 5326: kwaliteitsborging maatwerksoftwareontwikkeling en -onderhoud.

Boehm, B. (1991). Software risk management: principles and practices. IEEE software, 8(1), 32-41.

Boehm, B. & DeMarco, T. (1997). Software risk management. In European Software Engineering Conference, 17-19. Springer, Berlin, Heidelberg.

Boehm, B. & In, H. (1996). Identifying quality-requirement conflicts. IEEE software, 13(2), 25-35.

Boehm, B. & Ross, R. (1989). Theory-W software project management principles and examples. IEEE Transactions on Software Engineering, 15(7), 902-916.

Brooks Jr, F. P. (1995). The mythical man-month (anniversary ed.).

Bryman, A. (2016). Interviewing in qualitative research. Social research methods (5th ed.). New York: Oxford University Press, 465 - 499.

Burns, R. B., & Burns, R. A. (2008). Business Research Methods and Statistics Using SPSS. Sage Publications, 544.

Gioia, J. (1996). Twelve Reasons Why Programs Fail. PM Network, November, 16-20.

Han, W.M. & Huang, S.J. (2007). An Empirical Analysis of Risk Components and Performance on Software Projects. The Journal of Systems and Software, 80(1), 42-50.

Hoermann, S., Aust, M., Schermann, M., & Krcmar, H. (2012). Comparing risks in individual software development and standard software implementation projects: A Delphi study. In 2012 45th Hawaii International Conference on System Sciences, 4884-4893. IEEE.

Jones, C. (1998). Minimizing the risks of software development. Cutter IT Journal 11 (6), 13–21.

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. (1998). A framework for identifying software project risks. Communications of the ACM, 41(11), 76-83.

Kwak, Y. H., & Stoddard, J. (2004). Project risk management: lessons learned from software development environment. Technovation, 24(11), 915-920.

Lehder, W. E., Smith, D. P., & Weider, D. Y. (1988). Software estimation technology. AT&T technical journal, 67(4), 10-18.

Lubelczyk, J. & Parra, A. (1992). Recommended Approach to Software Development, Revision 3. Software Engineering Laboratory, NASA, June 1992.

NEN (2018). Ontwerp NPR 5326 - Kwaliteitsborging van maatwerksoftwareontwikkeling en -onderhoud.

NEN (2019). NPR 5326 - Risicobeheersing bij ontwikkeling en onderhoud van maatwerksoftware

NEN-ISO (2018). NEN-ISO 31000: Risk management - Guidelines (ISO 31000: 2018,IDT)

OECD (2020). Enterprises by business size. https://data.oecd.org/entrepreneur/enterprises-by-business-size.htm, accessed on May 5th 2020.

Paré, G., Sicotte, C., Jaana, M., & Girouard, D. (2008). Prioritizing clinical information system project risk factors: a delphi study. In Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), 242-242. IEEE.

Project team C2000, 2006. Rapport Eindevaluatie C2000.

Purdy, G. (2010). ISO 31000: 2009—setting a new standard for risk management. Risk Analysis: An International Journal, 30(6), 881-886.

Roberts, B. B. (2001). Lessons-learned: round 2. In 2001 INCOSE Proceedings of a Symposium on Risk Management, 3.

Ropponen, J. & Lyytinen, K. (2000). Components of Software Development Risk: How to Address Them? IEEE Transactions on Software Engineering, 26(2), 98-111.

Sangaiah, A. K., Samuel, O. W., Li, X., Abdel-Basset, M., & Wang, H. (2018). Towards an efficient risk assessment in software projects–Fuzzy reinforcement paradigm. Computers & Electrical Engineering, 71, 833-846.

Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying software project risks: An international Delphi study. Journal of management information systems, 17(4), 5-36.

Schwaber & Sutherland (2017). The Scrum Guide.

Standish Group (2014). Chaos Report

Wallace, L., Keil, M., & Rai, A. (2004a). How software project risk affects project performance: An investigation of the dimensions of risk and an exploratory model. Decision sciences, 35(2), 289-321.

Wallace, L., Keil, M., & Rai, A. (2004b). Understanding software project risk: a cluster analysis. Information & management, 42(1), 115-125.

9 Appendix

9.1 Appendix 1

Indicate for each risk how likely it is to occur in software projects, based on the projects you worked on.

Project management

	Chance					Comments
	Almost in all projects	In most projects	Neutral	Sometimes	Almost never	Optional
The amount of work was not correctly estimated causing the planned functionality not to be completed on time	○	○	○	○	○	
Inadequate management of the work, causing the product to not offer the correct functionalities	○	○	○	○	○	
Functional requirements are given too much priority so that the product lacks the correct non-functional properties	○	○	○	○	○	
The communication between stakeholders is suboptimal, causing misunderstandings	○	○	○	○	○	
Insufficient traceability of the development, use and management of customized software leads to no(non-demonstrable) compliance with obligations	○	○	○	○	○	
Because a lot of time is required to meet the preconditions for software development, the product is not delivered on time	○	○	○	○	○	
Lack of post-go-live approach	○	○	○	○	○	
Poor relationship management	○	○	○	○	○	
Gold plating: additional or more polished features than what was asked for or expected	○	○	○	○	○	
Project not based on a sound business case	○	○	○	○	○	

Top-management

	Chance					Comments
	Almost in all projects	In most projects	Neutral	Sometimes	Almost never	Optional
Unstable corporate environment due to for instance organizational issues or changes in organizational management/environment	○	○	○	○	○	
Mismatch between company culture and required business process needed for a new system	○	○	○	○	○	
Bad estimated budget or underfunding	○	○	○	○	○	
Insufficient resources	○	○	○	○	○	
Lack of top management commitment to the project	○	○	○	○	○	
Low project priority	○	○	○	○	○	
Insufficient staffing	○	○	○	○	○	

Project team

	Chance					Comments
	Almost in all projects	In most projects	Neutral	Sometimes	Almost never	Optional
The team does not have the right expertise, causing the software not to meet the requirements	○	○	○	○	○	
Dependency on a few key people	○	○	○	○	○	
Low morale in the team, which manifests in for instance the lack of project member's commitment to the project plan or not reporting possible risks	○	○	○	○	○	
Incorrect or optimistic status reporting	○	○	○	○	○	

Users

	Chance					Comments
	Almost in all projects	In most projects	Neutral	Sometimes	Almost never	Optional
Due to scope extensions, the product is not delivered on time and within budget	○	○	○	○	○	
Unrealistic expectations of the users	○	○	○	○	○	
Lack of user acceptance and therefore cooperation	○	○	○	○	○	
Conflict between users or user departments	○	○	○	○	○	
Poor perceived usability	○	○	○	○	○	

Development

	Chance					Comments
	Almost in all projects	In most projects	Neutral	Sometimes	Almost never	Optional
The software changed, resulting in deteriorated quality of the software	○	○	○	○	○	
The software environment changed, resulting in deteriorated quality of the software	○	○	○	○	○	
Availability of testing infrastructure	○	○	○	○	○	
Relative large project size, compared to other internal projects	○	○	○	○	○	
Technical complexity which may be due to the use of a new technology or straining computer-science capabilities	○	○	○	○	○	
Lack of effective development process/methodology	○	○	○	○	○	
The project wants to release new versions of the software more often than the management organization/users want or can handle	○	○	○	○	○	
Dependencies on other projects	○	○	○	○	○	
Number of organizational units involved or affected	○	○	○	○	○	

External risks

	Chance					Comments
	Almost in all projects	In most projects	Neutral	Sometimes	Almost never	Optional
Subcontracting and external partners	○	○	○	○	○	
Legal changes	○	○	○	○	○	

Do you miss a risk? Please explain why you find it important.

The listed measures below can be used to prevent or mitigate project risks. Indicate for each of one of them whether you would use it. Please indicate in the second column if the measure is specific enough to actually implement.

Organizational

	Do you use this measure?		Not specific enough to apply	Comment or further specification under which circumstances you would apply it
	Yes	No	Yes	Optional
Continuous risk management	○	○	○	
Recognise the role of culture and tone at the top	○	○	○	
Maintain stability in leadership and develop succession planning	○	○	○	
Support the teams by providing specialist knowledge and resources	○	○	○	
Teambuilding	○	○	○	
Recognizing outstanding efforts	○	○	○	
Spread knowledge of product components among various people	○	○	○	
Develop contingency plans to cope with staffing problems	○	○	○	
Key-personnel agreements	○	○	○	
Structuring career paths around an organizations product lines	○	○	○	
Cross-training: teach (some) skills or knowledge required to perform other job function	○	○	○	
Organization analysis	○	○	○	
Inspections at external companies	○	○	○	

Planning and vision

	Do you use this measure?		Not specific enough to apply	Comment or further specification under which circumstances you would apply it
	Yes	No	Yes	Optional
Clear and consistent distribution of responsibilities	○	○	○	
Identify important non-functional requirements	○	○	○	

Identify important functional requirements	○	○	○	
Shared product vision	○	○	○	
Clearly define the needs and benefits, by for example carrying out user surveys, making user characterizations or scenarios	○	○	○	
Product decomposition in incrementally deliverable parts with business value	○	○	○	
Smaller, and if possible measurable, project milestones	○	○	○	
Incremental delivery of the product	○	○	○	
Monitor progress using burndown charts	○	○	○	
Choose one appropriate project management method	○	○	○	
Proactive strategies; that involve planning and executing program activities based on anticipating future events	○	○	○	
Make timely, well-informed decisions and commitments	○	○	○	
Add a list of all known potential and relevant risks to the software project plan	○	○	○	
Proper planning by understanding and appreciating the likely complexity	○	○	○	
Assess cost and schedule impact of each change to requirements and specifications	○	○	○	
Be aware of optimism bias	○	○	○	
Cost-benefit analysis	○	○	○	
Detailed multisource cost and schedule estimation	○	○	○	
Design to cost	○	○	○	

Communication

	Do you use this measure?		Not specific enough to apply	Comment or further specification under which circumstances you would apply it
	Yes	No	Yes	Optional
Identify and involve stakeholders	○	○	○	
An official product owner with a mandate	○	○	○	
Involve management during the entire project lifecycle	○	○	○	
Archiving the documents and code when the development or the project is finished	○	○	○	
Sound transfer to the customers or another team	○	○	○	
Seek diversity in perspectives and information sources	○	○	○	
Create a pull for risk information	○	○	○	
Reward those who identify and manage risks early, even if the risks become problems	○	○	○	
Recognize and minimize bias in perceiving risk	○	○	○	
Avoid blame	○	○	○	
Make clear agreements in advance on the financial distribution	○	○	○	

Development

	Do you use this measure?		Not specific enough to apply	Comment or further specification under which circumstances you would apply it
	Yes	No	Yes	Optional
Requirements scrubbing: carefully examining for unnecessary or overly complex requirements	○	○	○	
Identify technical debt, provide insights and solve it according to plan	○	○	○	
Explore possible solutions, which includes prototyping	○	○	○	
Iterative development	○	○	○	
Software reuse	○	○	○	
Benchmarking: evaluating by comparison with an internal or external standard	○	○	○	
Set up an automated development pipeline	○	○	○	
Constantly meet the requirements using regression tests	○	○	○	
Instrumentation to measure the product's performance	○	○	○	
Software tuning	○	○	○	
Good configuration management	○	○	○	
Build in appropriate quality assurance processes	○	○	○	
Develop appropriate independent assurance arrangements	○	○	○	
Work within a central assurance framework	○	○	○	
Apply a quality-driven development method	○	○	○	
Using egoless programming	○	○	○	
Realistic expectations of the project team for the final product	○	○	○	

What other measures do you use to mitigate or prevent software project risks? Please explain.

Job description

- ○ Project manager
- ○ Software developer
- ○ Other, please specify

Size and type of the organization

- ○ Small enterprise: 1 to 49 employees
- ○ Medium enterprise: 50 to 249 employees
- ○ Large enterprise: 250 employees or more
- ○ Government organization

9.2 Appendix 2

ID	N	Mean	Std. Deviation	P (sig. 2-tailed)	1: Almost never	2: Sometimes	3: Neutral	4: In most projects	5: Almost in all projects
Risk 22	31	4,000	0,929	<,001	1	4	2	11	13
Risk 03	31	3,935	0,929	<,001	-	4	2	17	8
Risk 08	31	3,580	0,992	<,001	-	6	6	14	5
Risk 04	31	3,419	1,089	<,001	1	7	5	14	4
Risk 18	31	3,322	1,166	<,001	1	9	5	11	5
Risk 29	31	3,258	1,390	<,001	3	10	1	10	7
Risk 25	31	3,225	1,203	<,001	1	10	7	7	6
Risk 35	31	3,129	1,056	<,001	3	4	12	10	2
Risk 36	31	3,064	1,340	<,001	4	8	7	6	6
Risk 17	31	3,032	1,329	<,001	4	8	8	5	6
Risk 34	31	3,000	1,183	<,001	2	11	7	7	4
Risk 11	31	3,000	1,125	<,001	3	8	8	10	2
Risk 24	31	2,935	1,237	<,001	3	12	3	10	3
Risk 31	31	2,838	1,319	<,001	4	12	5	5	5
Risk 21	31	2,838	1,036	<,001	2	11	10	6	2
Risk 01	31	2,806	1,327	<,001	5	12	1	10	3
Risk 20	31	2,742	1,032	<,001	1	16	6	6	2
Risk 07	31	2,710	1,039	<,001	3	12	8	7	1
Risk 30	31	2,710	0,824	<,001	1	13	11	6	-
Risk 06	31	2,645	1,112	<,001	2	17	5	4	3
Risk 13	31	2,645	1,199	<,001	7	8	5	11	-
Risk 14	31	2,645	1,226	<,001	5	12	6	5	3
Risk 26	31	2,645	1,112	<,001	4	12	8	5	2

Risk 12	31	2,613	1,202	<,001	5	12	7	4	3
Risk 15	31	2,613	1,145	<,001	6	8	11	4	2
Risk 16	31	2,613	1,145	<,001	5	11	8	5	2
Risk 19	31	2,581	1,205	<,001	5	13	6	4	3
Risk 02	31	2,484	1,288	<,001	7	13	3	5	3
Risk 28	31	2,452	0,850	<,001	2	18	6	5	-
Risk 32	31	2,452	1,261	<,001	8	12	1	9	1
Risk 09	31	2,387	1,145	<,001	9	8	7	7	-
Risk 27	31	2,387	1,086	<,001	5	17	2	6	1
Risk 10	31	2,323	1,137	<,001	7	15	2	6	1
Risk 37	31	2,323	1,249	<,001	8	14	3	3	3
Risk 05	31	2,194	0,873	<,001	7	13	9	2	-
Risk 23	31	2,194	0,980	<,001	7	15	6	2	1
Risk 33	31	2,097	1,221	<,001	12	11	3	3	2

9.3 Appendix 3

Risk 01: The software changed, resulting in deteriorated quality of the software

- change is normal at Agile Development (almost never)
- I suppose this section means, the chance that the aspect forms a risk (in most projects)

Risk 02: The software environment changed, resulting in deteriorated quality of the software

- change is normal at Agile Development (almost never)

Risk 03: The amount of work was not correctly estimated causing the planned functionality not to be completed on time

- This is not a risk but a fact of life in IT. The impact depends on the effects of missing the deadline. (In most projects)
- it depends on how this question is interpreted. a project as a whole often does exceed it's estimated time. However, working in a Scrum team does make for better estimation on individual tickets of the project (in most projects)
- scrum-agile working helps a lot in feasible estimating (in most projects)

Risk 04: Due to scope extensions, the product is not delivered on time and within budget

- again the impact of this risk depends on the effects of missing the deadline (in most projects)
- scope extension also adds time+budget (almost never)

Risk 07: Functional requirements are given too much priority so that the product lacks the correct nonfunctional properties

- when working in agile, you will focus on an MVP and then add additional non-functional properties later (almost never)

Risk 08: The communication between stakeholders is suboptimal, causing misunderstandings

- this will always be the case, the question is if procedures are in place to facilitate mutual understanding (Neutral)

Risk 09: Insufficient traceability of the development, use and management of customized software leads to no(non-demonstrable) compliance with obligations

- not sure what you mean (neutral)
- not sure about this question (empty)

Risk 11: Lack of post-go-live approach

- not familiar with (empty)
- I don't understand the risk (empty)

– Idk (almost never)

Risk 13: Gold plating

– is often a big hindrance indeed (In most projects)
– as we do mostly product management, there is a multiplication factor that covers additional cost (in most projects)
– most often this happens at the start of a project when the deadline isn't near (sometimes)

Risk 16: Mismatch between company culture and required business process needed for a new system

– not sure what you mean (neutral)
– especially in larger companies (almost in all projects)

Risk 18: Insufficient resources

– the number of developers are a bottleneck if you are not working for an IT company, but IT has a business supporting function (in most projects)
– in my experience already started projects will receive additional resources if necessary, rather than getting cancelled (almost never)

Risk 20: Low project priority

– depends on the project (neutral)
– in my experience low priority can be a benefit to a project as well, by preventing management pressure and over-meddling by stakeholders (sometimes)

Risk 21: Insufficient staffing

– what is the difference between insufficient staffing and resources? (in most projects)

Risk 24: Incorrect or optimistic status reporting

– developers are always pessimistic (almost never)
– in my experience there are in fact two risks, There's optimistic status reporting because for example dependencies or testing or implementation efforts are underestimated, and there's downright fraud like delivering features for tests that have actually not been worked on at all or reporting tests as completed that have not been executed. (in most projects)

Risk 26: Lack of user acceptance and therefore cooperation

– in my experience this is usually coupled with "mismatch between company culture and required business process needed for a new system" (sometimes)

Risk 27: conflict between users or user departments

– not sure what you mean (neutral)

Risk 29: Availability of testing infrastructure

- unclear question (neutral)

Risk 30: Relative project size

- unclear question (neutral)
- what are you asking (neutral)
- in my experience this often has the same cause as the risk "The amount of work was not correctly estimated causing the planned functionality not to be completed on time" (in most projects)

Risk 31: Technical complexity

- you can always find a solution and make it work, you are just not sure if you found the best one (almost never)

Risk 35: Number of organizational units involved or affected

- unclear question (neutral)
- Again depends a bit on whether it is a pure IT company or an asset based company with IT. In the latter the business has to keep the place with IT and v.v. (almost in all projects)

Risk 37: legal changes

- But large impact: brexit, GDPR, etc. (Sometimes)

Other risks
I think it is a very good questionnaire covering all the key points,. As I mentioned on a couple of lines above, it is important to distinguish between a pure IT firm without tangible assets creating IT products and a company that has its core business in something else but has an IT development team to support it, as the risks can be quite different. When working in Agile for instance, the MVP of a product is to a much higher standard in an asset based environment because it needs to meet all the requirements to not bring and disruption to the production by the assets. While in a pure IT firms especially in the startup phase, there can be a lot more flexibility also because it might not require capital investments into assets.

Delayed decision making by customer or internally. This affects projects because it is unclear what the goal will be

Insufficient time allocated for testing. Not enough research into the right technology, resulting in choosing the wrong technology

Mismatch between sales and development visions. Focussing on projects that were delivered at suboptimal quality, in my experience the alignment of the sales team and the development team is very importance. If they sell a dream that can't be built that's a problem - if they don't comprehend the full possibilities of the development vision that's also a problem as there will be opportunities unused.

10 Appendix 4

Measure	N	Mean	Std. Deviation	P (sig. 2-tailed)	1: Yes	2: No	Not clear
Measure 03	31	1,000	0,000	<,001	31	-	1
Measure 32	31	1,000	0,000	<,001	31	-	2
Measure 60	30	1,000	0,000	<,001	30	-	5
Measure 02	31	1,032	0,180	<,001	30	1	2
Measure 07	31	1,032	0,180	<,001	31	1	1
Measure 01	31	1,032	0,180	<,001	30	1	2
Measure 05	31	1,032	0,180	<,001	30	1	3
Measure 08	31	1,064	0,250	<,001	29	2	1
Measure 51	31	1,097	0,301	<,001	28	3	1
Measure 20	30	1,100	0,305	<,001	27	3	3
Measure 22	31	1,129	0,341	<,001	27	4	1
Measure 04	31	1,129	0,341	<,001	27	4	2
Measure 12	31	1,129	0,341	<,001	27	4	2
Measure 06	31	1,129	0,341	<,001	27	4	2
Measure 56	31	1,129	0,341	<,001	27	4	2
Measure 35	30	1,133	0,346	<,001	26	4	6
Measure 29	31	1,161	0,374	<,001	26	5	1
Measure 37	30	1,167	0,379	<,001	25	5	4
Measure 16	31	1,194	0,402	<,001	25	6	2
Measure 30	31	1,194	0,402	<,001	25	6	5
Measure 43	31	1,194	0,402	<,001	25	6	1
Measure 48	31	1,194	0,402	<,001	25	6	2
Measure 21	30	1,200	0,407	<,001	24	6	-
Measure 55	30	1,200	0,407	<,001	24	6	5

Measure 31	31	1,226	0,425	<,001	24	7	1
Measure 33	31	1,226	0,425	<,001	24	7	2
Measure 14	31	1,226	0,425	<,001	24	7	3
Measure 50	31	1,226	0,425	<,001	24	7	2
Measure 10	31	1,226	0,425	<,001	24	7	2
Measure 09	31	1,258	0,445	<,001	23	8	2
Measure 15	30	1,267	0,450	<,001	22	8	9
Measure 11	31	1,290	0,461	<,001	22	9	2
Measure 44	30	1,300	0,466	<,001	21	9	3
Measure 53	31	1,322	0,475	<,001	21	10	1
Measure 19	30	1,333	0,480	<,001	20	10	8
Measure 54	29	1,345	0,484	<,001	19	10	2
Measure 26	31	1,355	0,486	<,001	20	11	2
Measure 36	31	1,355	0,486	<,001	20	11	2
Measure 13	28	1,357	0,488	<,001	18	10	3
Measure 23	31	1,357	0,495	<,001	19	12	2
Measure 17	30	1,400	0,498	<,001	18	12	2
Measure 40	30	1,400	0,498	<,001	18	12	2
Measure 38	31	1,419	0,502	<,001	18	13	3
Measure 27	30	1,433	0,504	<,001	17	13	7
Measure 34	30	1,433	0,504	<,001	17	13	11
Measure 49	29	1,448	0,506	<,001	16	13	5
Measure 52	31	1,484	0,508	<,001	16	15	3
Measure 18	30	1,500	0,509	<,001	15	15	7
Measure 59	25	1,520	0,510	<,001	12	13	14
Measure 39	31	1,548	0,506	<,001	14	17	6
Measure 47	28	1,571	0,504	<,001	12	16	6
Measure 57	29	1,586	0,501	<,001	12	17	3

Measure 58	28	1,633	0,488	<,001	10	18	6
Measure 46	31	1,645	0,486	<,001	11	20	2
Measure 25	30	1,667	0,480	<,001	10	20	2
Measure 45	29	1,690	0,471	<,001	9	20	11
Measure 42	31	1,710	0,461	<,001	9	22	3
Measure 24	28	1,714	0,460	<,001	8	20	3
Measure 28	30	1,733	0,450	<,001	3	22	2
Measure 41	31	1,742	0,445	<,001	8	23	2

10.1 Appendix 5

In brackets behind the comments the answers to the question are shown: first is the question whether they use it, and second whether it is clear enough to implement.

Measure 01: Identify and involve Stakeholders

- always a good idea, but a challenge with large stakeholder groups (yes, -)

Measure 05: Identify technical debt, provide insights and solve according to plan

- depends on if we have a green field or have legacy, if we have legacy definitely (yes, -)

Measure 06: Explore possible solutions

- sometimes (yes, yes)
- sometimes, not all projects (yes, -)
- I would use it for large projects (yes, -)

Measure 10: Constantly meet the requirements using regression tests

- if can be automated yes (yes, -)
- only risk based (no, -)
- not for all products in place yet (yes, -)

Measure 11: Monitor progress using burndown charts

- they suck (no, -)
- monitor progress: yes (yes, -)
- Monitoring progress is important, but we don't use burndown charts (yes,-)
- not with burndown charts (no, -)

Measure 12: An official product owner with a mandate Product owner

- not always, depends on the organization (yes, -)
- would like to, but was hard to realize in the organization (no, 0)

Measure 14: Archiving

- GIT does it for you (yes, yes)

Measure 15: Sound transfer to the customers or another team

- not sure what you mean (-, yes)

Measure 17: Continuous risk management

- good strategy. I would use it on big projects (yes, -)

Measure 18: Recognise the role of culture and tone at the top

- Always a good idea for the project manager (yes, -)

Measure 19: Maintain stability in leadership and develop succession planning

- always a good idea if possible, but unsure how to implement this (yes, yes)
- stability: yes, succession planning: no (yes, -)

Measure 20: Teambuilding

- always a good idea, especially when able to colocate teams (yes, -)
- we do kickoffs, not a lot of teambuilding (-, yes)

Measure 21: Recognizing outstanding efforts

- always a good idea, but be aware for the competition as a result (yes, -)
- it is more agile to reward teams (no, -)

Measure 24: key-personnel agreements

- not sure what you mean (-, yes)
- not sure what this means (-, yes)

Measure 25: Structuring career paths around the organization's product lines

- only applicable to hardware dev companies? (-, yes)

Measure 26: Cross-training

- always a good idea, if there is time and budget (yes, -)

Measure 28: Inspections at external companies

- not practical (-, yes)

Measure 30: Shared product vision

- always a good idea, but hard to implement (yes, yes)

Measure 31: Clearly define the needs and benefits

- depends on the size of the user group (yes, -)

Measure 32: Smaller, and if possible measurable, project milestones

- agile! (yes, -)

Measure 33: Choose one appropriate project management method

- use a mix depending on project (no, -)
- we are in transition from traditional to more agile (no, -)
- less management (yes, yes)
- we just have one we use for each project, it might not be appropriate every time though (yes, -)

Measure 34: proactive strategies

- what does this mean? (no, yes)

Measure 35: Make timely, well-informed decisions and commitments

- too broad (-, yes)

Measure 38: Assess cost and schedule impact of each change to requirements and specifications

- not always a good idea, might stifle the process (no, -)

Measure 39: Be aware of optimism bias

- developers are rarely optimistic (no, yes)

Measure 40: Cost-benefit analysis

- depends if clear hard important decisions are to be made. Do not burden every (yes, -)
- per feature? Or project as a whole? (-, yes)

Measure 41: Detailed multisource cost and schedule estimation

- no, too much work (no, -)

Measure 42: Design to cost

- it is not up to the development team to make such decisions. These are made by higher "authorities" within the company (no, -)

Measure 43: Involve management during the entire project lifecycle

- no, only involve management when decisions need to be made (no, -)
- you really need to (yes, -)

Measure 44: Seek diversity in perspectives and information sources

- if there is no rush (yes, -)

Measure 45: Create a pull for risk information

- what does this mean? (no, yes)

Measure 46: Reward those who identify and manage risks

- how is this different from recognizing outstanding efforts (no, yes)
- hard to implement (no, -)

Measure 47: Recognize and minimize bias in perceiving risk

- always a good idea, but how to achieve this? easier said than done (yes, yes)

Measure 48: Avoid blame

- always a good idea, but how to achieve this? easier said than done (yes, yes)
- although I have experience with other project leaders who used blame as a pressure tool on project members (yes, -)

- mostly (yes, yes)
- blame is important so mistakes are not repeated, what is more important is to teach how to blame correctly and how to accept blame (no, -)

Measure 50: Requirements scrubbing

- this is resolved via scrum (no, -)

Measure 51: Software reuse

- depends on the goals of the project (yes, -)
- only via libraries (yes, -)
- this brings its own risks (no, -)

Measure 52: Benchmarking

- only if we are using an external supplier, else leads to trust issues (yes, -)

Measure 53: Instrumentation to measure the product's performance

- often difficult to realize (no, -)
- not for all products in place yet (yes, -)

Measure 54: Software tuning

- depends on the goals of the project, small systems with hard deadlines, then definitely (yes, -)
- not clear what is meant here (empty, yes)

Measure 56: Build in appropriate quality assurance processes

- appropriate always a good idea, but often this means no (yes, -)

Measure 58: Work within a Central assurance framework

- not sure (no, -)

Measure 59: Use egoless programming

- always a good idea, but how to achieve this? Easier said than done (yes, yes)
- we hope (yes, -)
- ? (-, yes)
- no clue (-)
- what is that? (-, yes)

Measure 60: Realistic expectations of the project team for the final product

- make sure to have open communication to prevent misunderstandings (yes, -)

Other measures Weekly meetings with at least one person from the involved teams during the development. Having at least one colleague from each team helps enable open communication, which results in less miscommunication between departments.

10.2 Appendix 6

	Risk 03	Risk 04	Risk 08	Risk 11	Risk 17	Risk 18	Risk 22	Risk 24	Risk 25	Risk 29	Risk 34	Risk 35	Risk 36
Measure 01	X		X	X			X		X		X	X	
Measure 02	X												
Measure 03	X												
Measure 04	X	X			X								
Measure 05	X					X		X					X
Measure 06	X								X				
Measure 07	X	X	X					X	X				
Measure 08	X	X	X		X			X		X	X	X	
Measure 12		X	X	X				X	X		X	X	X
Measure 16						X	X						
Measure 20			X					X				X	
Measure 21								X					
Measure 22							X						
Measure 29	X		X				X				X	X	
Measure 30	X	X	X						X		X	X	X
Measure 32	X												
Measure 35	X		X		X						X	X	
Measure 37	X				X			X		X			
Measure 43					X	X					X	X	
Measure 48								X					
Measure 51	X	X			X		X						
Measure 55				X									
Measure 56				X							X	X	X
Measure 60													