

**Second Workshop on Requirements Prioritization for
customer-oriented Software-Development
RePriCo'11**

Held at the 17th International Working Conference on
Requirements Engineering: Foundation for Software Quality
(REFSQ2011)

31st of March 2011, Essen, Germany

Proceedings

Technical Program

1. RePriCo'11 - An Introduction	
<i>Georg Herzwurm, Wolfram Pietsch</i>	5
1.1 Welcome Message	5
1.2 Organization	6
1.2.1 Program Committee	6
1.2.2 Organizing Committee	6
2. Information Need of IT Risk Estimation – Qualitative Results from Experiments	
<i>Andrea Herrmann</i>	7
2.1 Introduction	7
2.2 Related Work	8
2.3 The IT Risk Estimation Experiments	9
2.4 Grounded Theory	11
2.5 Results	12
2.6 Discussion of Validity	15
2.7 Summary and Conclusion	17
3. Defining Product Lines and Product Variants based on Prioritization of Customer Segments and Customer Requirements	
<i>Andreas Helferich, Lars Oliver Mautsch</i>	21
3.1 Introduction	21
3.2 Quality Function Deployment	22
3.2.1 (Software) Quality Function Deployment	22
3.2.2 Software Product Lines and Product Line Engineering ...	24
3.3 Quality Function Deployment-Product Portfolio Planning (QFD-PPP)	25
3.4 Case Study: Job Scheduling Software	28
3.5 Conclusion	31
4. Knowledge-based Concretization of Requirements in Preparation for AHP	
<i>Constanze Kolbe, Robert Refflinghaus</i>	33
4.1 Introduction	33
4.2 Existing Approaches for the Establishment of a Requirements Hierarchy	34
4.3 Creation of a Knowledge Base	35
4.4 Knowledge-based Method for Concretization	37
4.5 Summary and forecast	42

5.	Construction and Evaluation of an Algorithmic and Distributed Prioritization Method	
	<i>Annabella Loconsole, Hannes Gruber, Adrian Nae, Björn Regnell</i>	45
	5.1 Introduction	45
	5.2 Requirements Prioritization Methods	46
	5.3 Context	48
	5.4 The Prioritization Method MAAD	48
	5.5 Evaluation	53
	5.6 Conclusions and Future Works	54
6.	Prioritizing business process chains for IT optimization	
	<i>Norman Riegel, Oezguer Uenalan, Thomas Jeswein</i>	57
	6.1 Introduction	57
	6.2 Methodological Background	58
	6.3 The Prioritization Approach	59
	6.4 Conclusion and Future Work	62

RePriCo'11 - An Introduction

Georg Herzworm¹, Wolfram Pietsch²

¹ Department for Business Administration and Information Systems, esp. Business Software,
University of Stuttgart, Keplerstr. 17, 70174 Stuttgart, Germany
herzwurm@wi.uni-stuttgart.de

² Business Management, International Sales and Service Management
Aachen University of Applied Sciences, Eupener Str. 70, 52066 Aachen, Germany
pietsch@fh-aachen.de

1 Welcome Message

Welcome to the Workshop on Requirements Prioritization for customer-oriented Software-Development (RePriCo'11) at the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ2011)!

We are glad about holding RePriCo'11 for the second time at REFSQ in Essen. Last year ambitious participants from research as well as industrial practice discussed two full research papers and four position papers in an open-minded and pleasant atmosphere. We are confident to hold a rewarding workshop alike 2010.

As far as prioritization is an essential task within requirements engineering in order to cope with complexity and to establish focus properly two perspectives can be identified:

From a formal standpoint of view prioritization is merely a matter of choice of the right specification method and granularity of analysis. From a practical perspective it is a matter of customer-orientation also: consensus must be achieved about the appropriateness of requirements from the view of the customers and fed back into the process.

The workshop therefore serves as a platform for the presentation and discussion of new and innovative approaches to prioritization issues for requirements engineering with a focus on customer-orientation.

RePriCo'11 attracted 9 submissions. Each submission was reviewed by two members of the program committee. Based on the reviews, four submissions were accepted as full research papers and one submission as short paper.

The submissions comprise current research findings from various fields: IT risk estimation based on qualitative empirical results (Andrea Herrmann); prioritization of requirements and customer segments for software product lines and variants (Andreas Helferich, Lars Oliver Mautsch); concretization of requirements by a knowledge-based and computer-aided method (Constanze Kolbe, Robert Refflinghaus);

formal prioritization methods and presentation as well as evaluation of one method implemented in industrial practice (Annabella Loconsole, Hannes Gruber, Adrian Nae, Björn Regnell); furthermore one approach for IT supported optimization of business process chains between public administration and business (Norman Riegel, Oezguer Uenalan, Thomas Jeswein).

We'd like to motivate all attendees to a high rate of active participation and intense exchange of ideas. We are convinced that the findings that will be shown encourage researchers as well as software-developers, requirements engineers or consultants to absorb new ideas and to carry them out into their daily work and research projects.

Enjoy RePriCo'11 and we hope you spent some eventful days at REFSQ2011 in Essen and the Ruhr Metropolis!

2 Organization

2.1 Program Committee

Chair

Prof. Dr. Georg Herzwurm, University of Stuttgart, Germany

Prof. Dr. Wolfram Pietsch, University of Applied Sciences Aachen, Germany

Member

Dipl.-Math. Peter Brandenburg, Vodafone D2 GmbH, Germany

Dr. sci. Math. Thomas Fehlmann, Euro Project Office AG, Switzerland

Priv. Doz. Dr. Andrea Herrmann, Axivion GmbH, Germany

Prof. Dr. Thomas Lager, Grenoble Ecole de Management, France

Dipl.-Betriebswirt (FH) Olaf Mackert, SAP AG, Germany

Dipl. Wirt.-Ing. Waldemar Meinzer, Volkswagen AG, Germany

Priv. Doz. Dr.-Ing. Robert Refflinghaus, TU Dortmund University, Germany

Dipl.-Wirt.-Inf. Sixten Schockert, University of Stuttgart, Germany

Prof. Dr. Hisakazu Shindo, University of Yamanashi, Japan

Dipl.-Ing. Gerd Streckfuß, iqm Institut für Qualitätsmanagement, Germany

Prof. Dr. Yoshimichi Watanabe, University of Yamanashi, Japan

2.2 Organizing Committee

Dipl.-Kfm. (FH) Benedikt Krams, University of Stuttgart, Germany

Dipl.-Wirt.-Inf. Sixten Schockert, University of Stuttgart, Germany

Information Need of IT Risk Estimation – Qualitative Results from Experiments

Andrea Herrmann

Axivion GmbH, Nobelstr. 15, 70569 Stuttgart, Germany
herrmann@axivion.com

Abstract. IT risk estimation is a basic activity supporting risk-based requirements prioritization, but also risk-based testing and risk management. Experiments participants experience IT risk estimation as difficult, estimated risks vary strongly among participants and estimations are badly repeatable. One main explanation for the difficulty and uncertainty in IT risk estimation is that much information is needed. This publication discusses what information is needed for IT risk estimation. Such information need was identified and classified by a Grounded Theory analysis of qualitative experiment results. Knowing what information is needed helps to prepare estimation workshops.

Keywords: requirements prioritization, IT risk, risk estimation, software risk

1 Introduction

Risk-based requirements prioritization ranks requirements regarding their potential to reduce risks. For instance, one can estimate the same risk twice – once assuming that the requirement is implemented and once without [AHP04], [HP09]. If, for example, a certain usability requirement reduces the risk of a user error from an estimated 10 times to 4 times per month and the cost of this risk is 10€, then the requirement reduces risk by $6 \times 10\text{€} = 60\text{€}$ month. This amount is to be compared to the requirement's cost as well as to the risk reduction achieved by the other requirements. Independently of which method or formula is used for transforming IT risk estimations into requirements priorities, the IT risk estimation task is the basis for all conclusions and decisions built on it. Similarly, risk estimations are used for other kinds of decision-making during software engineering, e.g. by a tester for risk-based testing (prioritizing test cases and errors) or by the project manager for risk management. Together with cost and benefit estimation, risk estimation can be said to be one of the fundamental tasks which provide for quantitative data which influence many decisions. Because the output of the IT risk estimation task is potentially used for success-critical decisions, it must be as correct as possible.

It has been shown by many researchers that risk estimation is difficult. In six experiments, we investigated more in depth with respect to which criteria IT risk estimation is difficult and how IT risk estimation can be improved. This publication analyzes qualitative findings (i.e. free text answers) from three of these experiments in order to investigate one main explanation for difficulty and uncertainty in IT risk estimation: We found that a lot of information is needed to estimate risk.

Which information this is we conclude from the experiment participants' comments. We analyzed these comments using the Grounded Theory method.

The remainder of this publication has the following structure: Section 2 gives an overview on related work. In Section 3, the three experiments are presented which produced the data which is analyzed in this publication. The Grounded Theory is introduced in Section 4. Section 5 presents the results we received when applying Grounded Theory to the qualitative data (i.e. free text comments) from the experiments. These results answer the question which information is needed for IT risk estimation. Section 6 discusses validity and Section 7 summarizes the results and draws conclusions.

2 Related Work

This related work section refers to two types of work: (1) work about how risk estimations can be used for requirements prioritization, and (2) what is difficult about IT risk estimation and what are Best Practices which improve it.

A *risk event* is an event which could take place in the future with a certain probability and which might have unwanted consequences. One risk event usually is not sufficient to describe what might happen. Instead, the security engineering often uses scenarios for describing risk, for instance in the form of Misuse Cases. Traditionally, Misuse Cases are used to elicit security requirements [SO00], [SO01]. However, MOQARE [HP08] successfully applies Misuse Cases to all types of non-functional requirements. Then, these Misuse Cases describe unwanted scenarios, where the misuser is not always a malicious attacker, but might be a user who by mistake impairs data integrity, or a developer who by negligence threatens the maintainability of a software. Misuse Cases during requirements elicitation can be used to identify countermeasures, i.e. requirements which, if satisfied, prevent, mitigate or detect Misuse Cases and thus support the satisfaction of security and of other non-functional requirements. Countermeasures can be requirements on the IT system, on its architecture, its development process, operation environment or personnel.

Risk r quantifies the importance of a risk event or Misuse Case by multiplying its probability p with the severity of the consequences (or, damage d): $r = p \cdot d$ [ISO02].

When requirements are prioritized based on risk estimations, then usually it is estimated by how much the countermeasure's implementation reduces a certain risk. This means to estimate risk twice for a system where this requirement is implemented and another system where it is not. This principle is used especially in the context of security (as in [AHP04]), but makes sense also with other non-functional requirements [FCK06], [HP09]. The Failure Mode and Effects Analysis (FMEA) [Sta03] and WinWin [PPB99] also apply it to functional requirements.

From literature and from our own experience, we have gathered the following difficulties of risk estimation and the corresponding recommendations:

Need of a reference system: Risk estimations must refer to the same reference system in order to be comparable [HP09], which is an idea of a system (in terms of which requirements are supposed to be satisfied).

Risk estimation demands *quantitative data about the past*. People have enormous difficulties in judging afterwards how often something happened [TK74], [BF00], [RRM02]. Good estimations need training and short feedback loops [BF00], [BF04]. There are three ways how to obtain quantitative data: measurement of historical data, public risk statistics (like [CER06], [Ric03]), or expert opinions [FC03].

Things change and risk estimation demands predictions about the future. You want to quantify risk for risk events which take place in the future, in a changed world and you do this before the new software system is implemented.

Do you ask the right person? Expert knowledge is needed from different perspectives, e.g. management knowledge, user knowledge, and technical knowledge.

Quantifying risk by *the product of probability and damage* does not allow to distinguish between high probability low damage risks and low probability high damage risks [BSI05], which you might want to treat differently. Therefore, always document the estimations of probability or damage.

Complexity: Feather et al. [FCG01], [FCL00] emphasize the importance of tool support, e.g. for visualization. For visualization, they use cost-benefit-diagrams or diagrams which for each risk present likelihood and impact on the two axes.

Time need and cost-benefit-ratio: The time need for risk-based requirements prioritization is higher than for other prioritization methods, because more estimations are to be made for performing the same task (i.e. estimate both probability and damage twice) and also each single estimation was found to take more time [HP09]. Effort-saving strategies can be: Estimate only those risks which are difficult to judge [Dav03] or which support the decisions of strategic importance [How68], reduce complexity by grouping similar risks into categories [REP03], [AHP04], and estimate risk at the point of time when it is needed.

3 The IT Risk Estimation Experiments

This publication analyzes qualitative findings (i.e., free text comments) from three experiments about risk-based requirements prioritization. Our objective is to explain why risk estimation was experienced as being difficult, why the risk estimations varied among participants and also varied when the same person repeated the same estimations several times. Especially, our research question is which information is needed for doing IT risk estimations. We search the answer to this question in the questionnaires analyzing the answers to the question what assumptions the estimators made when estimating risk and in comment fields concerning reasons for uncertainty and difficulty of IT risk estimation. While quantitative results have already been published [HP09] or will be published separately, here we summarize the qualitative results from three of our experiments, i.e. from 46 questionnaires.

These experiments are described in the remainder of this section. What is common to all three experiments is that risks were described as risk scenarios (Misuse Cases), specifying misuser, vulnerabilities, executed threat, and damage caused (like in MOQARE [HP08]). The participants knew MOQARE and its concepts.

The *first experiment* has been described and published before as “Experiment 1” in [HP09].

Objective: The experiment investigated practical needs of risk estimation, e.g.: time need, knowledge and material needed, the influence of group discussions. We also compared risk estimation to a simpler requirements prioritization method.

Sample population: Ten master students of the University of Heidelberg in the winter term 2006/07. They had been taught prioritization methods and risk-based requirements prioritization in a lecture before.

System: The experiment used twelve risks and nine requirements from a case study discussed in the lecture and in preceding homework: an Internet flea market.

Material: The material was an introduction (describing the objective of the experiment and the case study, including information about the company, competitors, project execution and staff, and the flea market's functionalities), the two questionnaires supporting the two prioritization methods, a questionnaire which asks the participants to rate the methods, and questionnaire 5.

Execution: The experiment was performed in a three hour session. Before the first questionnaire was distributed, there was an introductory presentation which explained the objective of the experiment and the methods, as well as the case example. During the experiment, the students individually estimated risks on paper questionnaires.

Main results: Moderated group discussions have positive effects (compared to individual estimations), although they are time-consuming. Risk estimation is difficult and requires a lot of information. Providing statistics to the estimators influenced results and lowered the standard deviation. However, the participants were not sure whether the statistics improved their results. The participants' and the moderator's experience with the method enhance confidence in the results.

Data analyzed using Grounded Theory: Here, we use the answers from questionnaire 5: One week after the experiment, during the post-test session, each participant received his/ her priorities resulting from each method. They were asked to comment whether the results reflect their opinion, on the methods and why results of different participants deviate so much from each other.

The *second experiment* was a pilot experiment to prepare the third experiment. **Objective:** Experiment 2 and 3 both tested the reliability, i.e. repeatability of risk estimations. Experiment 2 focused on the qualitative results, to learn which factors might influence repeatability and how to design Experiment 3.

Sample population: The participants were the author of this study and three students of the University of Braunschweig in the winter term 2009/10. The students had not learned about risk-based requirements prioritization before.

System: The experiment reused two case studies from former experiments: Experiment 1 and a second case study, a ticket vending machine. The latter experiment considered three risks and six requirements.

Material: The participants got the introduction from Experiment 1 describing the case study, while the ticket vending machine was illustrated by screenshots. The risk estimation questionnaires from previous runs of the same experiment were reused.

Execution: After a 20 minutes introductory presentation about the methods and case studies, both experiments were executed once. Then, each of the participants executed a second and third run at home, after at least two days. The author of this publication executed each estimation five times. Assumptions made were noted.

Results: The probability and damage estimations varied strongly. Two participants better repeated their risk estimations, while the other two produced risk estimations which seemed to stem from different persons.

Data analyzed using Grounded Theory: The list of assumptions made during the estimations and the results of the joint discussion of why repeatability might be low.

The *third experiment* repeated Experiment 2 with more persons and improved material.

Objective: The experiment investigated the repeatability of risk estimations and which factors influence the repeatability of risk estimation.

Sample population: Eleven bachelor and master students of the University of Heidelberg in the winter term 2010/11. They had been taught prioritization methods and risk-based requirements prioritization in a lecture before, and applied four different prioritization methods in homework. Methods and case study used in the experiment, however, were new.

System: The system used as a case study was the e-learning system which the students know and use for two to eight semesters already.

Material: The material included the case study description with screenshots and explanations, which partly quantify the risk damage, e.g. in terms of hours of downtime, and two questionnaires supporting one prioritization method each – the risk-based requirements prioritization and ranking.

Execution: The first run of the estimations took place in a lecture, the second as homework, the third and fourth in the lecture a month later. Each run took 30 minutes.

Results: The risk estimations have been found to be badly repeatable again. The quantitative results will be published elsewhere.

Data analyzed using Grounded Theory: Each of the prioritization questionnaires asked for assumptions made and for information which might have been missing. Further questions: Are there requirements or Misuse Cases where you have been especially uncertain? Which and why? Which information was missing? Do you think that risk estimation leads to realistic and practically useful results? Why or why not?

4 Grounded Theory

The Grounded Theory was developed by Strauss and Corbin [StCo90] for building theories from qualitative data (like protocols from interviews, research publications and other text of any form) in social research. It structures the data in a traceable way. We have successfully applied this method for analyzing results from empirical software engineering research and literature research before (e.g. in [RDH10]). The steps of Grounded Theory are [StCo90]:

- *Asking the research question*: “The research question [...] is a statement that identifies the phenomenon to be studied. It tells you what you specifically want to focus on and what you want to know about this subject.” ([StCo90], p.38)

- *Reading technical and non-technical literature to stimulate theoretical sensitivity*: “Theoretical sensitivity refers to the attribute of having insight, the ability to give meaning to data, the capacity to understand, and capability to separate the pertinent from that which isn’t.” ([StCo90], p.41f) This sensitivity can be reached by reading, by practical work or during the research.
- *Open coding*: Data are analyzed in order to identify concepts, grouping them to categories, identifying the categories’ properties and dimensions.
- *Questioning for enhancing theoretical sensitivity*: The categories are discussed, especially with respect to the data.
- *Axial coding*: Data are structured by making connections between categories; this is done by utilizing a coding paradigm involving causal conditions, phenomena, context, intervening conditions, action/ interaction, consequences.
- *Selective coding*: This means the identification of the core category, to write the story and story line.
- *Identification of process* means linking of action/ interaction sequences and contingency (unplanned happening).
- *Transactional analysis by a conditional matrix*: “The conditional matrix may be represented as a set of circles, one inside the other, each (level) corresponding to different aspects of the world around us. In the outer ring stand those conditional features most distant to action/ interaction; while the inner rings pertain to those conditional features bearing most closely upon an action/ interaction sequence.” ([StCo90], p.161)
- *Theoretical sampling*: The empirical data are sampled, using the concepts and categories found before. The objective is the discovery of as many relevant categories as possible, along with their properties and dimensions.

These steps can be and usually are traversed iteratively several times.

5 Results

This section summarizes the results of the Grounded Theory analysis of the qualitative results (text answers from participants on the questionnaires) of the three experiments described in Section 3. Due to lack of space, we cannot trace each step of the analysis and not present the results in their chronological order of emergence. This chronological order would describe a consecutive structuring of the data in a bottom-up fashion: From the raw data, abstracting properties were identified and grouped into categories, then linked to each other in “condition -> action -> consequence” stories. Finally, the core category is identified and all other results grouped around it. In what follows, the analysis results are summarized top-down instead.

The research question is: “What information is needed for IT risk estimation?”

Consistent with the research question and topic of the experiments, the *core concept* was chosen to be *risk*, which describe a deviation from the specified system use. The *phenomenon* to study are Misuse Cases.

It was found that the axial coding concepts directly map to the Misuse Case concepts: The *causal condition* of a Misuse Case is a user who tries to execute a functionality and/or a misuser who is motivated to execute a threat, what is an action

which deviates from the specified use. Intervening conditions are vulnerabilities (which enhance the probability of the Misuse Case or the damage caused), user and misuser properties. Context is characterized by the point of time when misuse happens and when a countermeasure is applied, and by the properties of the reference system against which the risk estimation is performed (its functionalities, technical implementation and countermeasures taken). Action/ interaction are described by the Misuse Case scenario. Consequences of the Misuse Case can either be success (i.e. specified use) or damage, because the Misuse Case by definition happens with a certain probability and therefore damage might occur or not. In what follows, the Misuse Case concepts are used as the categories to structure the data.

The story line emerging from the data is presented graphically in Fig. 1 in the form of an event tree. It is important to mention that this is not the usual model used to describe Misuse Cases, but it is more detailed. Misuse Cases can follow each other, i.e. the “damage” of one Misuse Case can be the threat which initiates another Misuse Case. The experiment participants asked a lot of what-if questions about probabilities of different event flows, like “How many users use the countermeasure?” or “What happens if the countermeasure is not effective?” or “When the Misuse Case happens, does this mean that the student is excluded from the exam?” (Remark: Success and damage are defined from the user perspective, not the misuse perspective. A successful hack of a database by hacker therefore is counted as damage.) Note that the risk probability estimated in the experiment should correspond to the sum of the probabilities of all scenarios which end with “damage”. However, as these numerous different possible flows of events were not specified in the experiment material, it is possible that the estimators were not aware of this complexity, and this can be one reason for the deviations between risk estimations of different participants and for the participants’ feeling that risk estimation is rather difficult to execute and the results might partly not be too realistic.

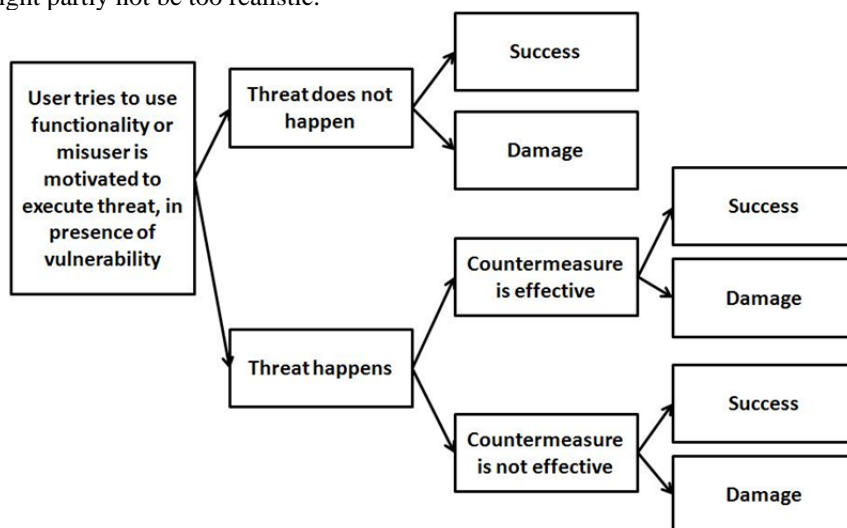


Fig. 1. Story line.

Open coding was a bottom-up analysis of the data. We present separately the information about reality (i.e. system and system (mis)use), and the information which is specifically needed for risk estimation. Both information types together are needed for IT risk estimation. **Table 1** summarize the results of open coding which are categories (i.e. concepts), properties of the categories and dimensions of the properties. The tables also give each concept's category of axial coding and its category from the conditional matrix. When doing a simple transactional analysis, only few categories were found: There is the system, its users, the organization where the system is operated (company or university) and the world outside this organization. For classifying the present data, no more categories were needed.

Table 1. Analysis results by open coding, axial coding and conditional matrix: What information describes the system and system (mis)use?

category/ concept	property	dimensions	axial category	conditional matrix category
reference system	countermeasures	further countermeasures implemented	context	system
	functionalities	tasks, steps, importance, number, frequency, support by system, number of screens	context	system
	technical properties	implementation of database, hardware availability, further vulnerabilities	context	system
countermeasures	degree of usage by users	sometimes, often not	intervening conditions	users
	effectiveness against a certain risk	how often, to what degree? sometimes, considerably	intervening conditions	system and users
	alternatives to countermeasure	available or not; which: workaround, other media/ system functionality for achieving the same objective (e.g. e-mail submission of homework, instead of using the e-learning system, the browser's back button)	intervening conditions	system and users
	risks of the countermeasure which make it less effective	damage happens nevertheless, technology failure, user error	actions	system and users
	countermeasure implementation	easily found? Understandable?	intervening conditions	system
points of time	point of time when countermeasure is applied	before submission of homework, after submission of homework	context	users
	point of time when Misuse Case happens	a day before homework deadline, directly before deadline	context	users
consequences	consequences of consequence	consequences for higher goals like admission to exam, pattern (how many data are concerned, does it happen regularly?), alternative ways for reaching a goal exist	consequences	system and users
users	user role	student, study domain, developer	intervening conditions	users
	technology usage skills	Moodle, internet browsers, web interfaces	intervening conditions	users
	general knowledge	able to read, intelligent	intervening conditions	users
	do they use countermeasure?	sometimes, often not, are angry when must use countermeasure	intervening conditions	users
	behaviour in case of misuse case	accept deficit, migrate to competitor, achieve goal by trial and error, organizational countermeasures taken, do they hear of security problems?	intervening conditions	users
	experience in using this system	for the first time	intervening conditions	users
misusers	motivation	how many want to execute the misuse case?, effort put into executing the misuse case, misuser's abilities	intervening conditions	users and rest of the world
system environment	organization within which system is operated	University Heidelberg	context	organization
	technological context	interfaces to other data systems	context	organization
	market	properties of competing systems	context	rest of the world

Table 2. Analysis results by open coding, axial coding and the conditional matrix: What information is needed specifically for risk estimation?

category	property	dimension
probability	metric	number of times, percent
	transformation of scales	How many times correspond to 1%?
	statistics	historical data
damage perspective of estimator	metric	image, cost
	role	user, maintainer, developer
	consistent with perspective of misuse case?	yes, no
	experience	completeness: some misuse cases are not experienced, although they happen
	priorities	according to user perspective, relative priorities of quality attributes (e.g. security versus usability)
	experience and knowledge	technical, market
context of estimation	time pressure	yes
system goals	task	submission of homework
	constraint on tasks	e-learning: each student does his homework without knowing the solutions of others
	relevance of data	relevant for lecture or not

6 Discussion of Validity

Validity of empirical research means that the results reliably answer the research question posed and that it is known for which conditions these results are valid and where they are not. The research question investigated here is: “What information is needed for IT risk estimation?”

The data used for answering this question stem from three experiments where IT risk was estimated. These experiments investigated other research questions than the one of this publication, the data were gathered asking questions which were different in each of the experiments. These questions mainly asked for missing information and for difficulties in using the method. This means we use material from heterogeneous experiments and questions. There might be a threat for validity due to this, however, only a small one as in all these questions the experiment participants mainly discussed missing information, even when not being explicitly asked for that. We expect that the information that was given in the experiment material is not mentioned in these answers. However, this given information can also be described by the model in **Table 1** and **Table 2**, i.e. it would not add further categories.

The categories found seem to be saturated. For most of the categories, several instances were found in the data, often five to ten. We analyzed more than 100 sentences or keywords, and after half of them, only very few new categories emerged. However, the dimensions evidently are not saturated. For instance, for the property “degree of usage of the countermeasure”, only two dimensions were found (“sometimes” and “often not”), although there exist more possibilities like “always” or “never”.

One reason for the low number of dimensions is that often the data had the form of questions, like: “Do users apply this countermeasure?” instead of “I assume that users often do not apply this countermeasure”. Such questions contributed to the model categories and properties, but no dimensions. Either more data than that of 46 questionnaires resp. three experiments would be needed for completing the dimensions, or questions should be asked which rather ask for assumptions than missing information. Whether the properties are saturated, we cannot say. The low number of three systems analyzed also poses a threat to validity. It is possible that with respect to other systems, further categories, properties and dimensions would have emerged. These partly are system-specific.

In Grounded Theory, usually it is recommended that more than one scientist does the coding of the data. In the present research, this was not possible, as it was executed by one person only. Therefore, subjective classification errors would not have been discovered.

Open coding demands much discipline to keep to the data (and not add categories to the model that is not based on data), especially when categories start to emerge which one knows from a long-used conceptual model like the Misuse Cases in this study. However, in our Grounded Theory protocol, the original data are traceably mapped to the categories, to make sure that no concepts enter the resulting model without being supported by real data and vice versa, that all data fit into these categories and no data stay unclassified.

The experiment participants as well as the coder know Misuse Cases and their concepts. This easily explains why the Misuse Case concepts were found in the data and these are overrepresented against concepts which are not part of a Misuse Case description (like “point of time”). It is possible that if the involved persons did not know Misuse Cases, different or further concepts would have been found.

We can assume that IT risk estimation in practice and IT risk estimation in an experiment is a different situation. For instance, we have observed in an unpublished (confidential) case study where we applied IT risk estimation in a real software project that practitioners who are experts of the system feel much more confident about their IT risk estimations and in the using risk-based requirements prioritization method, compared to students in an experiment. The practitioners know the system to treat very well, including the developer perspective. And they may modify the wording and granularity of Misuse Cases and requirements and even modify the method according to their need. Experiment participants may not modify method, Misuse Case descriptions or requirements, and have no possibility to research for statistics. This turns risks estimation more difficult in an experiment using hypothetical scenarios or even a fictitious system. In the experiments, therefore, we have always chosen a system which the estimators know at least from the user perspective by regular use. However, they were aware that their own experience with the system is not representative. While these factors reduce the ease of IT risk estimations and its quality, they are rather helpful in investigating which information is needed for IT risk estimations. We expect that a student experiment can provide for a more complete model of information than asking practitioners, who use implicit expert knowledge without being aware of it.

7 Summary and Conclusion

This publication presents the results of a Grounded Theory analysis of qualitative data from three experiments about risk estimation, i.e. from 46 questionnaires. This analysis follows the research question “What information is needed for IT risk estimation?” Knowing what information is needed helps to prepare a risk estimation workshop, in real-life workshops as well as in teaching and scientific experiments.

Our findings imply that risk estimation needs diverse information about the system and its context, and good descriptions of the risk scenarios – the success scenarios as well as the diverse flows of the damage scenarios. These scenarios could be described as event trees, better than with the conventional Misuse Case descriptions. Based on our findings from our experiments, we recommend to provide risk estimators with the following material:

- A case study description including the success scenarios of system use, the technology and users, and the context. More details about what is needed can be found in **Table 1** and **Table 2**. As this is a lot of information, instead of defining all this a priori, one can discuss and define the assumptions jointly during the estimation workshop.
- Descriptions of the success and risk scenarios in the form of event trees, where the risk of each step and branch should be estimated separately. As before, we would illustrate these scenarios by adding screenshots of the system for different scenarios (although our experiment results say nothing about the usefulness of the screenshots). Quantification seemed helpful, like defining the number of hours of a system breakdown.

Due to validity threats to the results (which are stronger for the dimensions than for the categories), the model developed in this publication probably is not yet complete. Furthermore, as we have discussed above, that both experiment participants and researcher know Misuse Cases well may have influenced the data in the experiments and the model based on it. Therefore, in future it would be interesting to gather similar data by executing further experiments with participants who do not know Misuse Cases. Nevertheless, they should have a computer science background or at least substantial software user experience in order to discuss about IT risks competently.

References

- [AHP04] Arora, A., Hall, D., Pinto, C.A., Ramsey, D. and Telang, R.: An ounce of prevention vs. a pound of cure: How can we measure the value of IT security solutions? Carnegie Mellon CyLab. (2004)
- [BF00] Boehm, B.W., Fairley, R.E.: Software Estimation Perspectives. *IEEE Software* 17(6), 22-26 (2000)
- [BSI05] BSI: BSI-Standard 100-1: Managementsysteme für Informationssicherheit (ISMS) Version 1.0, https://www.bsi.bund.de/cae/servlet/contentblob/471450/publicationFile/30749/standard_1001_pdf.pdf (2005)
- [BF04] Bundschuh, M., Fabry, A.: Aufwandschätzung von IT-Projekten. 2. Edition, mitp Verlag, Bonn (2004)
- [CER06] CERT/CC Statistics 1988-2006, http://www.cert.org/stats/cert_stats.html
- [Dav03] Davis, A.M.: The Art of Requirements Triage. *IEEE Computer* 36(3), 42—49 (2003)
- [FC03] Feather, M.S., Cornford, S.L.: Quantitative Risk-based Requirements Reasoning. *Requirements Engineering Journal* 8(4), 248—265 (2003)
- [FCG01] Feather, M.S., Cornford, S.L., Gibbel, M.: Scalable Mechanisms for Requirements Interaction Management. In: *IEEE International Conference on Requirements Engineering*, Schaumburg, USA, 119—129 (2001)
- [FCK06] Feather, M.S., Cornford, S.L., Kiper, J.D., and Menzies, T.: Experiences using Visualization Techniques to Present Requirements, Risks to Them, and Options for Risk Mitigation. In: *Int. Workshop on Requirements Eng. Visualization*, Minneapolis/St. Paul, Minnesota (2006)
- [FCL00] Feather, M.S., Cornford, S.L., Larson, T.: Combining the Best Attributes of Qualitative and Quantitative Risk Management Tool Support. In: *15th IEEE International Conference on Automated Software Engineering*, Grenoble, pp. 309–312 (2000)
- [Her10] Herrmann, A.: *Praktische Tipps für die Risikoschätzung und risikobasierte Anforderungspriorisierung*. ignite, Düsseldorf, Germany (2010)
- [HP08] Herrmann, A., Paech, B.: MOQARE: Misuse-oriented Quality Requirements Engineering. *Requirements Engineering Journal* 13(1), 73—86 (2008)
- [HP09] Herrmann, A., Paech, B.: Practical Challenges of Requirements Prioritization Based on Risk Estimation. *Journal of Empirical Software Engineering* 14(6), 644--674 (2009)
- [How68] Howard, R.A.: The foundations of decision analysis. *IEEE Trans. Systems, Sci. Cybernetics* 4(3), 211—219 (1968)
- [ISO02] ISO (International Standards Organization): ISO, Risk management – Vocabulary – Guidelines for use in standards, ISO Guide 73. Geneva: International Standards Organization (2002)
- [PPB99] Park, J., Port, D., Boehm, B., In, H.: Supporting Distributed Collaborative Prioritization for WinWin Requirements Capture and Negotiations. In: *Int. 3rd World Multiconference on Systemics, Cybernetics and Informatics SCI'99*, Vol.2, 578--584 (1999)
- [RDH10] Racheva, Z., Daneva, M., Herrmann, A.: A Conceptual Model of Client-driven Agile Requirements Prioritization: Results of a Case Study. *ESEM Conference 2010*, Bolzano (Italy)
- [RRM02] Raiffa, H., Richardson, J., Metcalfe, D.: *Negotiation analysis - the science and art of collaborative decision making*. Cambridge: Belknap (2002)
- [Ric03] Richardson, R.: *CSI/FBI Computer Crime and Security Survey*, Computer Security Institute, http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2003.pdf (2003)
- [REP03] Ruhe, G., Eberlein, A., Pfahl, D.: Trade-Off Analysis For Requirements Selection. *Int. J. Software Eng. And Knowledge Eng.* 13(4), 345--366 (2003)

- [SO00] Sindre, G., Opdahl, A.L.: Eliciting Security Requirements by Misuse Cases. In: TOOLS Pacific 2000, 120--131 (2000)
- [SO01] Sindre, G., Opdahl, A.L.: Templates for Misuse Case Description. In: 7th Int. Workshop on Requirements Eng.: Foundation of Software Quality – REFSQ, Essener Informatik Beiträge Band 6. Essen, Germany, pp. 125--136 (2001)
- [Sta03] Stamatis, D.H.: Failure Mode and Effect Analysis - FMEA from Theory to Execution. Milwaukee, USA: American Society for Quality Press (2003)
- [StCo90] Strauss, A.L., Corbin, J.M.: Basics of qualitative research - grounded theory procedures and techniques. 6th print, Sage, Newbury Park, USA (1990)
- [TK74] Tversky, A., Kahneman, D.: Judgment under uncertainty: Heuristics and biases. *Science* 185, 1124—1131 (1974)

Defining Product Lines and Product Variants based on Prioritization of Customer Segments and Customer Requirements

Andreas Helferich¹, Lars Oliver Mautsch²

¹ Andreas Helferich – Software Management Consultant,
Braystr. 18, 81677 Munich, Germany
mail@andreashelferich.de

² Universität Stuttgart, Chair of Information Systems II (Business Software),
Keplerstr. 17, 70174 Stuttgart, Germany
mautsch@wi.uni-stuttgart.de

Abstract. Quality Function Deployment (QFD) is a well-known Quality Management and Product Engineering method used in various industries all over the world. It can also be used in Software Development, where it is usually used to prioritize requirements. We extended QFD in a way that allows for the prioritization of requirements and also customer segments, which in turn allows for the definition of (Software) Product Lines and the Product Variants included in a Product Line. Our approach – called Quality Function Deployment – Product Portfolio Planning (QFD-PPP) enables software companies to base decisions on core and variable assets on actual customer needs and therefore assists in controlling product (and especially architectural) complexity.

Besides explaining the method and related approaches, this paper also describes the successful use of QFD-PPP in an industrial case study in which a standard software package used in data center operations was analyzed with regard to possible variants.

Keywords: Case Studies, Product Portfolio Planning, Product Variants, QFD, Software Product Lines

1 Introduction

Serving the needs of a large number of customers with different needs is one of the distinguishing features of commercial off-the-shelf (COTS) software packages. This is achieved through including a large number of features and a significant degree of built-in flexibility, allowing the customers to adapt the software to their specific needs. This allows software companies to “mass-produce” COTS software packages (or rather to develop the software once, make copies of the software with hardly any effort and sell it to a potentially large number of customers), resulting in significantly lower prices compared to custom-developed software. Where different customers’ requirements are relatively similar but diverse when analyzed in further detail, offering a number of distinct variants of the software package can be beneficial.

A large number of software development techniques have been developed to cope with this complexity. Still, every variant developed results in additional effort.¹ Research on complexity management shows that purposefully controlling complexity by limiting the number of variants offered can be very beneficial. The proponents of Software Product Lines take this into account by arguing that an activity called Scoping should precede the main development effort. Scoping is the process during which decisions about what to develop, i.e. which products will be part of the product line and what the commonalities and variabilities will be, are made. At the same time – and maybe even more important – it is also determined what not to develop, i.e. the product line is bound on several levels building upon each other [1]: Product Portfolio Scoping, and Asset Scoping.

Currently, no single scoping approach addresses all three levels. One of the most comprehensive scoping approaches is the PuLSE-Eco approach [1]. Unfortunately, current approaches to Product Line Scoping usually treat the number of variants to be offered and the customer segments targeted with each variant as input from Marketing [1]. We designed QFD-PPP to support decisions on the product portfolio and to limit the number of variants offered according to actual customer requirements.

As pointed out previously [2], QFD-PPP allows software companies to develop a platform-based product portfolio (possibly implemented using Product Line Engineering) that optimally satisfies customer demands and at the same time restricts the number of products offered. Focus of this paper is to demonstrate how QFD-PPP can support software companies' portfolio planning. To this end, we begin by briefly describing QFD, Software Product Line Engineering as well as existing variations of QFD used to define product variants. Following this, we describe QFD-PPP and present the results of a case study performed with a German software company offering standard software used in data center operations.

2 Quality Function Deployment

2.1 (Software) Quality Function Deployment

QFD has been developed in the Japanese manufacturing industry [3], [4] and has been successfully applied to develop customer-oriented products in services in a variety of industries [5]. Compared to traditional ways of formalizing and specifying product requirements, “QFD provides a systematic but more informal way of communication between customers and developers”. [6] Instead of trying to compile an exhaustive list of requirements that specifies the requirements in great detail, QFD aims to identify the most important requirements and to define a vision for the product to be developed. Therefore, a project team consisting of customer representatives, developers/engineers and QFD-experts serving as facilitators collaborates during the whole QFD process. The mission of this team is to assure that the final product's features are not determined by the technically possible but by the fitness for use,

¹ Actually even variants not yet developed but intended to be offered in the future and already considered in the software architecture add a certain level of complexity.

i.e. the features the customers demand. The software developers and/or engineers assure that the features can be implemented and that technological breakthrough innovations are not ignored.

In order for QFD to be used for software development, two differences are considered: first, the software production process is basically a duplication process and implementation is largely determined by the system design, especially the system architecture. Therefore, the effort has to be directed mainly into the earlier stages. Secondly, “Software [...] is valued not for what it is, but for what it does” [7]. Thus, the distinction between product functions and quality elements has to be made: a product function is a “functional characteristic feature of the product, usually not measurable (creates perceptible output)” [8], while a Quality Element is a “Non-functional characteristic feature of the product, possibly measurable during development and before delivery (does not create perceptible output)” [8].

The most important purpose of QFD in software engineering and the main focus of product planning is on setting prioritized development goals based on the most important customer requirements [8].

The best-known instrument of QFD is the so-called House of Quality (HoQ) [9]. Generally speaking, the HoQ is the matrix² which analyzes customer requirements in detail and translates them into the developers’ language. Since QFD is a very flexible method, a QFD project can use a number of matrices or none at all. Still, the HoQ and related matrices are at the core of most QFD projects. Frequently, these matrices are linked with each other with the output of one matrix serving as the input of another matrix. Various models describing possible deployments and matrix sequences exist, applicable for software development are (among others) the deployment framework developed by the German QFD-Institute [10], Zultner’s Comprehensive Software Quality Deployment [11] or Ohmori’s Software Quality Deployment [12]. For an in-depth description of QFD and its application in various contexts see [3], [4].

But applying QFD is more than filling out a HoQ matrix. A number of techniques (e. g. Conjoint Analysis [13], Target Costing [14] or New Lancheater Theory [15]) can be used so as to provide the most benefit.

In order to make sure all relevant points of view and all available knowledge about customer requirements are included in the process, the entire QFD process is carried out by a project team consisting of representatives of all departments (development, quality management, marketing, sales, service etc.) and is to be extended in several team meetings by carefully selected customer representatives. Substituting a customer survey, one of the first meetings tries to ascertain customer needs and to classify them in the so-called Voice of the Customer (VoC) Table. These requirements are structured using affinity- and tree diagrams and weighted (e. g. by pair-wise comparison or the Analytic Hierarchy Process [16]) by as many members of the customer groups as possible under control of the customer representatives. The weights of the different groups are then used to calculate the average weight by calculating the average of the weights assigned by the customer groups weighted with the importance of the groups.

² In QFD, a matrix is basically a table contrasting two factors and their correlations, for the HoQ these are Customer Requirements and Technical Requirements.

If a new release of an existing product is developed, customer requirements can be used to evaluate the level of satisfaction with the current fulfilment of the requirements (measured on a scale ranging from 1 indicating total dissatisfaction to 5 indicating perfect satisfaction). If sufficient and reliable information about competing products is available, these can also be evaluated using the customer requirements collected. Analysis of the satisfaction with the competing products supports decisions regarding the focus of the new release. The second major input is the Voice of the Engineer Table, compiled by the QFD team, among them particularly developers, that includes potential product functions and measurable quality elements. Both are derived from the requirements by the developers. The relationships between product functions and customer requirements are identified together with the customer representatives. Figure 1 displays an excerpt of a Software-HoQ for an email client including the tables of customer requirements and product functions.

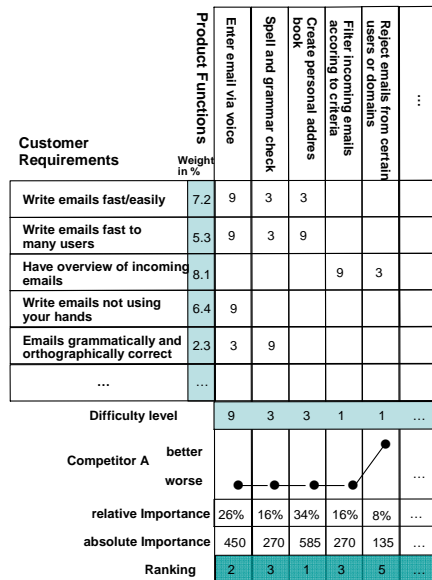


Fig. 1. Excerpt of the Software-HoQ for an email client [8].

2.2 Software Product Lines and Product Line Engineering

While the Marketing definition of a Product Line does not make any implications whether the members of the product line are technically related [17], the term “Software Product Line” implies that different products of one domain (also referred to as problem space or application range, e. g. operating systems for mobile telephones or software support of the sales department) are viewed as a product family and not as single products. According to the Software Engineering Institute, Software Product Lines are defined as “set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way“. [18] The components of a Software Product Line are the product

line architecture and the individual products which are part of the product line. The product line architecture describes the individual products, their common components and - at least in outlines - the differences between the products of the family (e.g. [18]). Different process models exist for the development process of product lines, e.g. those described in [19] or [20]. They have in common, that the product line development process is modeled along the structure of a product line. Just as the product line consists of product line architecture and product line members, the development process also consists of the process of the development of the product line architecture and the development process of product line members. The development of the product line architecture is called domain engineering and the development of product line members is called application engineering.

As briefly mentioned in the introduction, preceding both is an activity called scoping during which the use of the product line or its products is planned (e.g. [1]). More explicitly, scoping aims at distinguishing between requirements that are outside of the scope of the product line and requirements that are within the scope of the product line. The latter are further distinguished in requirements common to all products and variable requirements, i.e. requirements that only one or some of the members of the product line need to fulfill. For a further description of the activities and results of Software Product Line Engineering see for example [18].

3 Quality Function Deployment-Product Portfolio Planning (QFD-PPP)

There are a few examples in literature where QFD was used to define product variants. In [21], these are analyzed and an explanation is given why these examples fall short of realizing the full potential of applying QFD for Software Product Lines. To fix these shortcomings, we developed QFD-PPP.

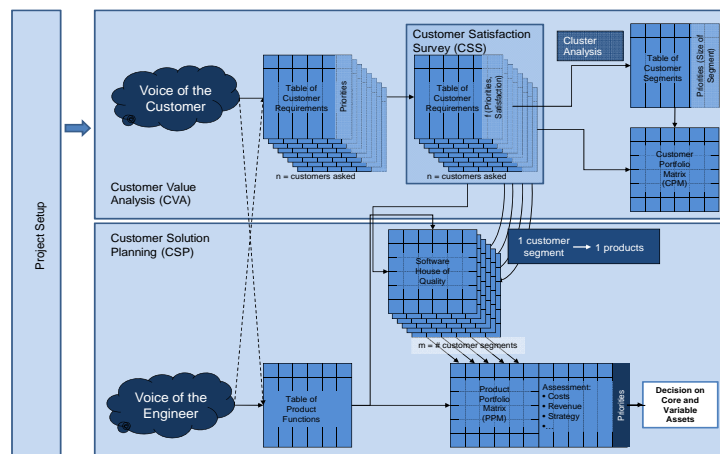


Fig. 2. Overview of QFD-PPP (simplified).

Our approach to Product Portfolio Planning makes extensive use of QFD while at the same time introducing two new matrices (see Figure 2). First of all, the Voice of the Customer (VoC) is collected by asking existing and potential customers about the requirements they have for the product line. Once these answers are collected, they are analyzed and sorted before asking the customers to assign priorities to all requirements. Once these priorities are assigned, customer segments are formed based on these priorities using cluster analysis. Thus, unlike traditional QFD approaches, there is no weighting of customer groups as this is only necessary to come up with common (or rather average) priorities. Another difference is the identification of customer groups not by attributes of the customer (e.g. job title or role description) but by statistical analysis. In this regard, our approach also differs from approaches which identify customer segments and their sizes based on the products purchased and their market shares (e.g. [15]).

The next step is to get together developers, software architects and selected customers (based on the clusters identified) to build the Software House of Quality. Explicitly including the Voice of the Engineer in the form of product functions is important to identify innovative software characteristics that customers themselves would not have come up with.

Since a product function's level of fulfilling a customer requirement is independent from the weight assigned to the requirement, there is basically only one Software-HoQ for all the members of one product line. But since the weights of the customer requirements depend on the customer segments, the weight of the product functions does so either. Therefore, there is a different Software-HoQ for each product variant, but the different Software-HoQs differ only in the weights of customer requirements and product functions. The Software-HoQ in Figure 1 equals the Software-HoQ for one of the customer groups (including the weights) for an email-client, e.g. attorneys used to dictate letters who would therefore being able to dictate emails, too. The first new matrix is called **Customer Portfolio Matrix**. It provides an overview of all customer requirements and customer segments, including the importance assigned to the requirements, and is displayed in Figure 3.

Customer Requirements	Customer Segments /Products (in %)			...
	Product Line Member/ Customer Segment #1	Product Line Member/ Customer Segment #2	Product Line Member/ Customer Segment #3	
Write emails fast/easily	7.2	5.3	6.8	
Write emails fast to many users	5.3	6.3	7.5	
Have overview of incoming emails	8.1	11.2	8.1	
Write emails not using your hands	6.4			
Emails grammatically and orthographically correct	2.3	7.4	5.3	
...				

Legend

: 10% most important customer requirements

: 25% most important customer requirements

Fig. 3. Customer Portfolio Matrix.

As a working hypothesis, for each customer segment exactly one product variant is designed (as indicated in Figure 2). Core and variable features are identified by comparing the weight of the product functions for the different customer segments. This is visualized in the second new matrix called Product Portfolio Matrix that relates product functions to members of the product line. This matrix is not a prioritization matrix such as the HoQ, but rather a way of displaying relevant information next to each other in order to facilitate decision-making. For example, the software company could decide to offer a very advanced spell and grammar check as part of all product line members (as displayed in Figure 4) order to a) provide an advantage with regard to competitors A and B and b) avoid the complexity that would result from offering different spell checks.

Product Functions	Products					
	Product Line Member #1	Product Line Member #2	Product Line Member #3	...	Competitor A	Competitor B
Enter email via voice	●	○	○		○	○
Spell and grammar check	●	●	●		◐	◐
Create personal address book	●	●	●		●	●
Filter incoming emails according to criteria	◐	◐	◐		◐	◐
Reject emails from certain users or domains	◐	◐	◐		●	◐
...						

Legend

- : fulfilment level 100%
- ◐ : fulfilment level 75%
- ◑ : fulfilment level 50%
- ◒ : fulfilment level 25%
- : fulfilment level 0%

Fig. 4. Product Portfolio Matrix (excerpt).

Thus, the Product Portfolio Matrix relates product functions and members of the product line and helps prioritizing the variants. Additional inputs are the expected costs for the product functions and the expected revenue a product will achieve. The second depends on the size of the potential market, the products currently available on the market and the customer satisfaction with these products and the advantage the member of the product line have over these products. Ulwick's so-called *opportunity algorithm* [22] or the algorithm used in [8] can be used as indicators here. Both algorithms use the importance of a feature and the customers' satisfaction with the current solutions provided by own and competitors' products to identify features where improvements provide a competitive advantage. A more detailed economic assessment is presented in [1].

In addition to the described prioritization and decision-support for functional requirements and characteristics, prioritization and decision-support for non-functional

characteristics and software quality attributes are enabled. To this end, the software developers and software architects evaluate different software architectures and technologies taking into account necessary quality attributes. This is also done by using matrices, where the roof is intensively used to analyze the impact that different architectural or technological elements have on each other. The results of this analysis are used to decide on the software architecture and the technologies to be used for prototypes. If necessary, an in-depth evaluation of architecture alternatives using methods such as SACAM or ATAM can be executed [23].

These prototypes can also be used to demonstrate exciting features the software developers and software architects came up with and the proposed solutions to the requirements voiced by the customers. If all prototypes are shown to all customers, some of the customers will decide to include some features they previously hadn't assigned value to, maybe drop some features they requested. This discussion is based on the product functions, not the original customer requirements and their weights. Only when large changes are asked for, the customer requirements will be re-evaluated.

Additionally, derivation of new products for a Software Product Line and the evolution of the Software Product Lines and its members are facilitated, since the already existing matrices can be used as templates (a similar course of action for agile software development was proposed in [24]). Using the matrices as a starting point leads to reductions in both time-to-market and costs and helps achieving important goals associated with Software Product Lines.

4 Case Study: Job Scheduling Software

This case study was conducted in cooperation with a German software company that currently offers a standard software package in the field of datacenter operations. Due to technical reasons, the company intended to completely redesign the software. At the same time, they were considering to serve another market segment using a variant of their product with reduced functionality compared to the main product. As a long-term option, they were also interested in offering every customer a version of the product that was tailored to his needs. The current product is offering extensive functionality for a comparatively modest price, whereas management had decided the product variant was to be positioned as easy to use and quite competitively priced. The company approached us in order to a) assure that the resulting small product line met customer requirements and b) analyze the differences in customer requirements between the projected customer segments for the two members of the product line.

During project setup, this objective was defined in more detail: the present release plan of the standard software should be evaluated (and if necessary adapted) with regard to the coverage of the requirements of the currently served market segment. Additionally, competing products should be examined referring to these requirements and their fulfillment. Furthermore, we were asked to investigate how far the requirements of the targeted new market segment differ from those of the currently served market segment and which consequences these differences have on the product characteristics.

After the objectives were defined, a first workshop was conducted to collect and structure the customer requirements. Therefore representatives from marketing and sales as well as developers from the German software company, representatives of several customers and representatives of a potential customer as well as an industry expert were part of the project team. An external marketing expert, who was responsible for developing a new marketing concept for the software package, was also part of the team. The end users of the software, the so-called operators, and their supervisors were identified as relevant customer groups.

The responsibility of the operators is to assure that different software programs are executed in their data centers according to plan. The supervisors need to know about any exceptions and problems, since they are responsible for quality of service and will receive complaints from the data centers' customers. Representatives of both groups were invited to and participated in the process. The Voice of the Customer Analysis generated 71 customer requirements, which were divided into 12 requirement categories. Then the requirements were weighted by the customer representatives. Figure 5 displays an excerpt of the customer requirements including the results of a customer satisfaction survey and a comparison with two competing products that were also conducted with the participants of the workshop.

Members of the additional customer segment that was intended to be targeted with the "scaled-down" variant of the software were included after this workshop. In order to get an even more detailed impression of their requirements, in-depth interviews with three companies were conducted. During these interviews, the customer requirements identified during the workshop were discussed. According to the characteristics given by marketing, all three companies belonged to the market segment targeted with the scaled-down version. The objective of this discussion was to evaluate a) in what way the potential second market segment differs from the currently served customer segment and b) if this second market segment is actually homogeneous enough to be targeted with one single product variant. As expected there were some common patterns of difference in the importance of the customer requirements: rather basic requirements were relatively rated higher, whereas a few of the advanced requirements were rated lower.

Looking at the results for the new market segment in further detail, there were unexpected differences between the requirements of the three companies. E.g. one of the interviewed companies would need this software primary for the support of its in-house SAP system, so that several requirements such as multi-client capability or business process monitoring wouldn't be relevant. The second company instead wouldn't need the support of different time zones, as it currently operates just in Germany, unlike the other two companies. The third company focuses especially on the multi-client capability as it operates as a service provider offering the administration of IT systems for other companies. These differences lead to the conclusion that the targeted market segment was not as homogeneous as expected. While it was clearly defined (companies with a midsized datacenter, which usually did not use this kind a software so far, but had the operators fulfill this task manually or using simple scripts), pretty large and not served well by existing products, the customer needs within the segment are too inhomogeneous to be covered by one single clearly defined product (respectively such a product would only be purchased by parts of the segment). This reflects a well-known problem in marketing:

any segmentation approach should lead to segments that are easy to define and target with your marketing and sales campaigns and at the same time highly relevant in predicting purchasing behavior [25]. Obviously, the segment planned here was easily targeted but not very relevant in predicting purchasing behavior since customer needs vary too much. As a result the German software company decided to implement a modular design which consists of a basic system and several additional modules which are independent from each other. This way, the planned additional market can be served with the pretty aggressively priced basic system, making it attractive to first-time users. At the same time, the company can demonstrate to the currently served market segment how powerful the “all in”-system is and how long reference customers have been using such a system. The decisions which product characteristics the basic system has to provide and which modules to be offered are supported by the results obtained using QFD-PPP.

Using the information gained in the workshop and through the satisfaction survey as basis, the plans for the next releases of the existing products were re-evaluated next. 71 potential product characteristics grouped into 12 groups were presented to the project team in another workshop and evaluated with regard to their fulfillment of the customer requirements.

requirements (category)	functional requirement	level of satisfaction	satisfaction customer A	satisfaction customer B	comparison - competitor A	comparison - competitor B
A administration		6,43	(10=very good, 1= very poor)		6,43	5,86
1 A.1	easy to use authorization scheme	6,5	6	7	8	5
2 A.2	easy identification of users	7,5	6	9	9	5
3 A.3	effort-saving software-deployment	5	5	5	5	7
4 A.4	easy (un)installation of software-agents	6,5	5	8	4	7
5 A.5	configurable system-parameters	7,5	6	9	7	7
6 A.6	high usability software-administration	6,5	5	8	8	5
7 A.7	effort-saving release-updates	5,5	5	6	4	5

Fig.5. Customer Requirements including satisfaction and competitive analysis (excerpt).

As mentioned before, the requirements of the new customer segment were too inhomogeneous to be satisfied with a single additional software solution. At the same time the development team was unable to divide the segment into smaller customer groups which requirements could be met with a variant of the product portfolio. This was caused by the small pool of customers who could be asked about their requirements. So it seemed reasonable to start with a different approach: The software company decided to avoid developing a fixed number of variants of their software. Instead of this, a full version like before and a basic version of the product for the new customer segment should be offered. For this basic version a flexible upgrade path which allows customers to add several optional modules when needed was developed. Because customers in the software sectors are used to that kind of cafeteria system when licensing software, this approach seems promising. Figure 6 shows an excerpt of the identified modules for the basic and full version. So customers can upgrade their basic version step-by-step to a full version according to their actual needs.

Product functions		basic version	optional modules	full version
I	architecture			
a	...			
b	...			
c	...			
d	...			
e	...			
II	GUI			
a	...			
b	...			
c	...			
d	...			
e	...			
f	...			
g	...			
h	...			
III	security			
a	...			
b	...			

Fig. 6. Comparison of the product versions (details blurred out for confidentiality).

5 Conclusion

It has been demonstrated how QFD-PPP can be used to identify different customer segments and their needs, to derive a product portfolio (i.e. members of a product line) systematically and derive common and variable product functions including exciting requirements that the customers would not have come up with. By identifying customer and the importance that these segments ascribe to different requirements, Value-Based Product Development is made possible. The results of the case studies and feedback gathered from participants show that QFD-PPP is well-suited to aid software companies in planning software product lines.

While we were not able to actually use cluster analysis as planned due to problems acquiring the number of potential customers necessary for the statistical analysis, we were able to demonstrate in the case study that a segment formed using traditional marketing methods was too broad and that a value-based analysis helped defining optional modules that can be offered to the customers. Collecting sales figures for these modules and having all new customers prioritize the requirements collected so far, clusters of customers can be formed over the next few years using real customers. Already with the small sample of three potential customers, we were able to detect heterogeneous requirements of customer groups before actually developing the software. This way, companies using QFD-PPP can avoid spending too much effort in developing software not suitable for the requirements of a customer segment.

References

1. Schmid, K.: Planning Software Reuse – A Disciplined Scoping Approach for Software Product Lines, Stuttgart: Fraunhofer IRB (2003)
2. Helferich, A., Herzwurm, G., Milcz, M. and Schockert, S.: Product Portfolio Planning using QFD, Proc. of the 14th International Symposium on QFD, Beijing (2008), pp. 334-343

3. Cohen, L.: *Quality Function Deployment*, Addison-Wesley, Reading, MA (1995)
4. Akao, J.: *Quality Function Deployment: Integrating Customer Requirements into Product Design*, translated by Glenn H. Mazur, Cambridge, MA (1990)
5. Chan, L.-W. & Wu, M.-L.: Quality function deployment - A literature review. *European Journal of Operational Research*, 143 (2002), pp. 463–497.
6. Herzwurm, G.; Schockert, S., Pietsch, W.: QFD for Customer-Focused Requirements Engineering, in: *Proceedings of the 11th IEEE International Requirements Engineering Conference*, Monterey Bay, USA (2003) 330-338.
7. Zultner, R. E.: Software quality function deployment – the North American experience, in: *Software Quality Concern for people. Proceedings of the Fourth European Conference on Software Quality*, Zürich (1994), pp. 143-158.
8. Herzwurm, G.; Schockert, S., Mellis, W.: *Joint requirements engineering: QFD for rapid customer-focused software and Internet-development*, Vieweg, Wiesbaden (2000)
9. Hauser, J.R., Clausing, D.: The House of Quality. *Harvard Business Review*, 66 (3), (1988), pp. 63-73.
10. Herzwurm, G., Schockert, S.: What are the Best Practices of QFD, in: *Proceedings of the 12th International Symposium on QFD*, Tokyo (2006)
11. Zultner, R.E.: *Quality Function Deployment for Software: Satisfying Customers*, in: *American Programmer* (1992), pp. 28-41.
12. Ohmori, A.: Software quality deployment approach: framework design, methodology, and example, in: *Software Quality Journal* No. 3 (1993), pp. 209-240.
13. Green, P.E., Krieger, A.M., Wind, Y.: Thirty Years of Conjoint Analysis: Reflections and Prospects, in: *Interfaces*, 31 (3), (2001), pp. 57-573.
14. Gustafsson, A.: *Customer focused product development by conjoint analysis and QFD*. Dissertation Univ. Linköping, Dep. of Mechanical Eng., Div. of Quality Techn. (1996)
15. Fehlmann, T.: *New Lanchester Theory for Requirements Prioritization*, in: *Proc. of the 2nd International Workshop on Software Product Management* (2008), pp. 35-40.
16. Saaty, T. L.: *Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World*, 3rd edition, Pittsburgh, USA (1995)
17. Helferich, A., Schmid, K., Herzwurm, G.: *Reconciling Marketed and Engineered Software Product Lines*. In: *Proceedings of the 10th International Software Product Line Conference*, 23-27, Baltimore (2006)
18. Clements, P., Northrop, L.: *Software product lines: practices and patterns*. Addison-Wesley, Boston, MA (2002)
19. Bayer, J. et al.: PuLSE: A Methodology to Develop Software Product Lines, *Proceedings of the 5th Symposium on Software Reusability* (1999), pp. 122-131.
20. Weiss, D.M., and Lai, C.T.R.: *Software product-line engineering: a family-based software development process*, Boston (1999)
21. Herzwurm, G., Schockert, S., and Helferich, A.: QFD-PPP: Product Line Portfolio Planning using Quality Function Deployment, *Proceedings of the 9th International Software Product Line Conference*, Rennes (2005), pp. 162-173.
22. Ulwick, A.W.: Turn Customer Input into Innovation, *Harvard Business Review*, 80(1), (2002), pp. 91-97.
23. Stoermer, C., Bachmann, F., Verhoef, C.: SACAM: The Software Architecture Comparison Analysis Method. Technical Report CMU/SEI-2003-TR-006, Software Engineering Institute, Carnegie Mellon University (2003)
24. Herzwurm, G., Schockert, S., Breidung, M., Dowie, U.: *Requirements Engineering for Mobile-Commerce Applications*, in: *Proceedings M-Business*, Athens (2002)
25. Homburg, C., Krohmer, H.: *Grundlagen des Marketingmanagements*. 2nd Ed., Gabler, Wiesbaden (2009)

Knowledge-based Concretization of Requirements in Preparation for AHP

Constanze Kolbe¹, Robert Refflinghaus²

¹ Technische Universität Dortmund, Lehrstuhl für Qualitätswesen,
Joseph-von-Fraunhofer Str. 20, 44227 Dortmund, Germany
Constanze.Kolbe@lqw.mb.tu-dortmund.de

²RIF e.V.,
Joseph-von-Fraunhofer Str. 20, 44227 Dortmund, Germany

Abstract. While planning a complex product, a huge number of requirements and weightings from several stakeholders are emitted. These requirements must be implemented into solutions in consideration of their weightings. The method Analytic Hierarchy Process (AHP) can be applied in order to prioritize the requirements structured. Therefore a hierarchy of requirements must be established. However, Stakeholders are only to a limited extent able to refine their requirements systematically during the planning process. Due to that the definition of a requirements hierarchy is difficult. A knowledge-based and computer-aided method for the concretization of requirements will be presented. Therewith stakeholders can refine their emitted requirements systematically which leads to the establishment of a requirement hierarchy. This serves as a preparation for the requirements prioritization in the framework of AHP.

Keywords: requirements concretization, AHP, hierarchy, ontology

1 Introduction

While planning a complex product (e.g. software) a high amount and variety of requirements and their weightings coming from different stakeholders are emitted. Thus, for example, the development of a computer-aided quality (CAQ)-software is a complex matter, because different industries and companies have specific requirements on a CAQ-software and there exists a multitude of application areas. Due to that, requirements and their weightings on software must be investigated carefully, in order to transform them into the optimal solution and thus to offer the optimal software-solution consisting of different application modules. Stakeholders of software are for example the buyer, the software engineer, system maintenance engineer and the user. Due to the complexity of planning process, a structured and traceable weighting of requirements is difficult to realize. There is a high risk that the emitted weighting of the whole amount of requirements are not consistent. The Analytic Hierarchy Process (AHP) is a method which expedites an effective decision-making process and which enables to check and to improve the consistence of weightings [1]. This method comprises several steps (fig. 1).



Fig. 1. Steps of AHP.

In the first step a goal has to be formulated that forms the top of the hierarchy [2]. In the case of software development the goal would be the establishment of a software according to the requirements of the stakeholders. After that the lower levels of the hierarchy must be generated which have to consist of sub-criteria and alternatives [3]. In the framework of product development sub-criteria have to be formulated in form of requirements and the alternatives in form of possible solutions. By means of paired comparisons weightings of the sub-criteria can be obtained [1]. Afterwards the consistence of the comparison can be checked [4]. The prioritization of alternatives forms the final step.

Accordingly, the requirements hierarchy forms the basis for the prioritization by using the method AHP [5], [6], [7], [8]. Thereby, the concretization of the goal by dint of more concrete criteria is the most creative and crucial step during the appliance of AHP [3]. One problem in this case is that the gathered requirements are different regarding their level of concretization [9]. An establishment of a hierarchy of these requirements is therefore a complex matter. Outstanding expertise is necessary in order to build up a requirements hierarchy. After a stakeholder has formulated his requirements, it is not traceable for a third person (etc. the developer of the software) which of these requirements defines another requirement more precisely according to the stakeholder. The structuring of the whole amount of requirements into a hierarchy is therefore a complex and uncertain process. In order to solve the problem each stakeholder should define by himself which requirements do define his formulated requirement in more detail. This would create a requirement hierarchy for each stakeholder.

2 Existing Approaches for the Establishment of a Requirements Hierarchy

Approaches for the establishment of a requirements hierarchy already exist. One approach is the structuring of requirements into primary, secondary and tertiary requirements [10], whereby the degree of detail rises from a primary requirement to its tertiary requirements. One further approach for the concretization makes a distinction between the concretization of reference object, attribute and value of a requirement [11].

With these approaches a requirements hierarchy can be manually defined. However, an average stakeholder is unpracticed in formulating and refining of requirements. He has no appropriate methods, in order to set requirements systematically [6]. Due to that a new computer-aided method for concretizing their requirements has been developed. This method identifies potential refining requirements of an emitted requirement and provides these to the stakeholder.

Thus the stakeholder can choose from these provided refining requirements those which are in sum sufficient to fulfill his emitted super-ordinated requirement. Therefore a high amount of knowledge about potential requirements as well as their potential detailing requirements must be available. On order to solve this problem a knowledge base has been created which can be accessed by the developed method during the automatic concretization of requirements. The knowledge base has been established in the form of an ontology which represents a formal description of knowledge by a set of concepts within a domain [12]. Within the ontology necessary knowledge can be modeled and becomes therefore machine-interpretable. In each individual application of AHP, a new requirements hierarchy has to be created [13]. Due to that the knowledge base can be utilized steadily with each planning process. The ontology is extendable and therefore its knowledge base increases with each additional planning process.

3 Creation of a Knowledge Base

An ontology consists of a class hierarchy in which the relevant terms are housed and consists of properties which are placed between the classes. A property specifies the unidirectional dependency between two ontology classes. The classes themselves contain a set of individuals, which can be named objects as well as instances [14]. Under each main-class a set of sub-classes have to be defined. Each of these sub-classes contains a term of a considered knowledge-domain.

In order to create a knowledge base for the concretization of requirements at first information components for the depiction of possible requirements were established. A typical requirement has a reference object, a property and a value [15]. The following figure 2 shows an example of a requirement which has been divided up into its three components.

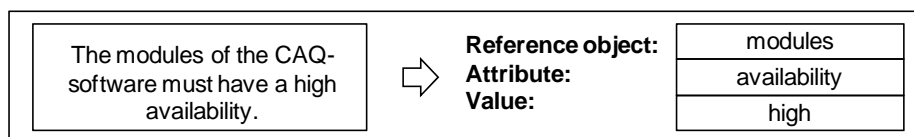


Fig. 2. Requirement components.

In this way an amount of potential requirements were defined and modeled within the ontology by using the classes "reference object", "attribute" and "value" as well as the properties "has attribute" and "has value" (fig. 3). "Has_attribute"-properties connect the sub-classes of the „reference object“-hierarchy with the sub-classes of the „attribute“-hierarchy. They are needed to specify which attributes each reference object can potentially have. The "has_value"-properties are defined between "attribute"-sub-classes and "value"-sub-classes. This kind of properties specifies which values an attribute can assume. The mentioned properties define only sensible connections between the classes and subclasses. These combinations comprise all potential requirements that can occur during the collection of requirements.

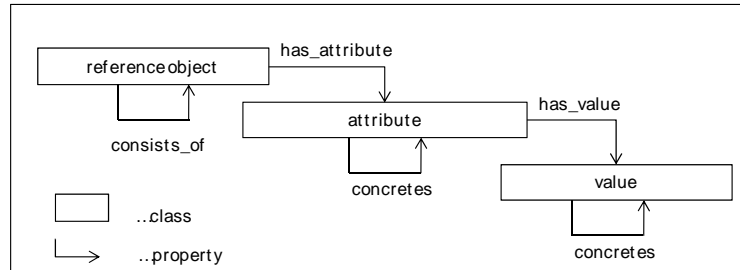


Fig. 3. Schematic overview of the ontology.

Also knowledge about which requirement component concretizes potentially another component is modeled within the ontology. This knowledge is on the one hand stored by means of the structure of class hierarchy. In this regard, each term of a class must have a „is a“-relation to the term of its super-ordinated class-, which means that each sub-term must be a sub-form of the super-ordinated class term. And on the other hand, the knowledge is defined by different properties. Concerning the “reference objects”, these are the “consists of” properties which are set between two sub-classes of the class hierarchy “reference object”. They define which reference object could potentially consists of another reference object. In order to store knowledge about the concretization of attributes „concretizes“-properties were defined between the classes of the „attribute“-hierarchy. With reference to the concretization of values, “concretizes“-properties were set between classes of the “value”-hierarchy. In the following, cut-outs of an ontology belonging to the domain CAQ-software are presented.

Class Hierarchies

The reference object-hierarchy for the product “CAQ-Software” (fig. 4) comprises the “CAQ-Software” as a whole, its modules such as “First sample examination module” and its components like the “Presentation layer”. The second upper class is named „Attribute“. Its subclasses are for example the “Availability” and the “Inspectability”. The third upper class is the demanded “Value” of the attribute. The required value of the reference object’s attribute can be differentiated into qualitative (e.g. “High”, “Low”) and quantitative value.

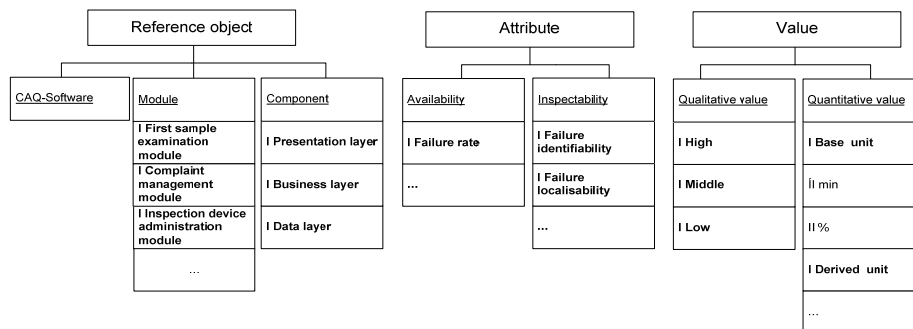


Fig. 4. Class-hierarchy.

Properties

The ontology-cut-out shown in figure 5 comprises properties which are defined between the three relevant classes (“Module”, “Availability” and “High”) of our given example in figure 1 and other classes of the ontology.

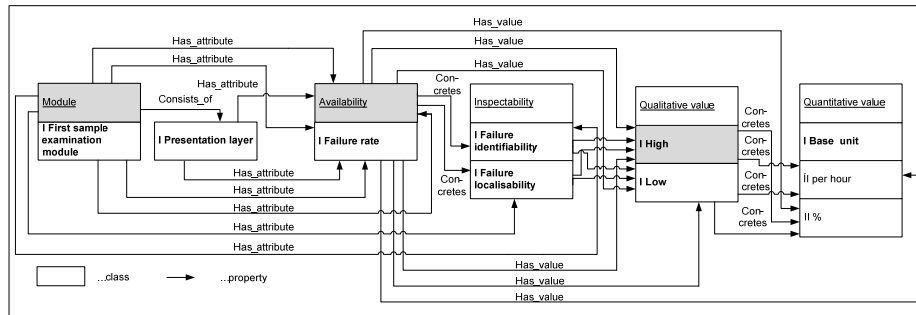


Fig. 5. Example for defined properties.

Over the “Has_attribute”-properties it is defined which attributes a module of a software can have. By means of the “Has_value”-properties it is defined which values the attribute “Availability” can assume. The relations between the reference objects were defined by using a “consists of” property. A “module” consists e.g. of a “Presentation layer”. Also “concretizes”-properties are defined between the components attribute and value of our given example and the other ontology-classes.

4 Knowledge-based Method for Concretization

During the planning process stakeholder utter their requirements in natural language. In order to put the requirements in relation to the knowledge of the ontology the requirements have to become highly formalized. During their formalization, a requirement is divided into its information components (fig. 1).

During the appliance of the developed method the stakeholder can choose between the concretization of the reference object, the attribute and the value of his considered requirement. In case of the concretization of reference object the user can also chose if there should be a “is a” or a “consists of” relation between the reference object of the given requirement and the reference objects of the provided detailing requirements. The following procedure describes how the method identifies automatically detailing requirements of a given requirement on the basis of the ontology (fig. 6).

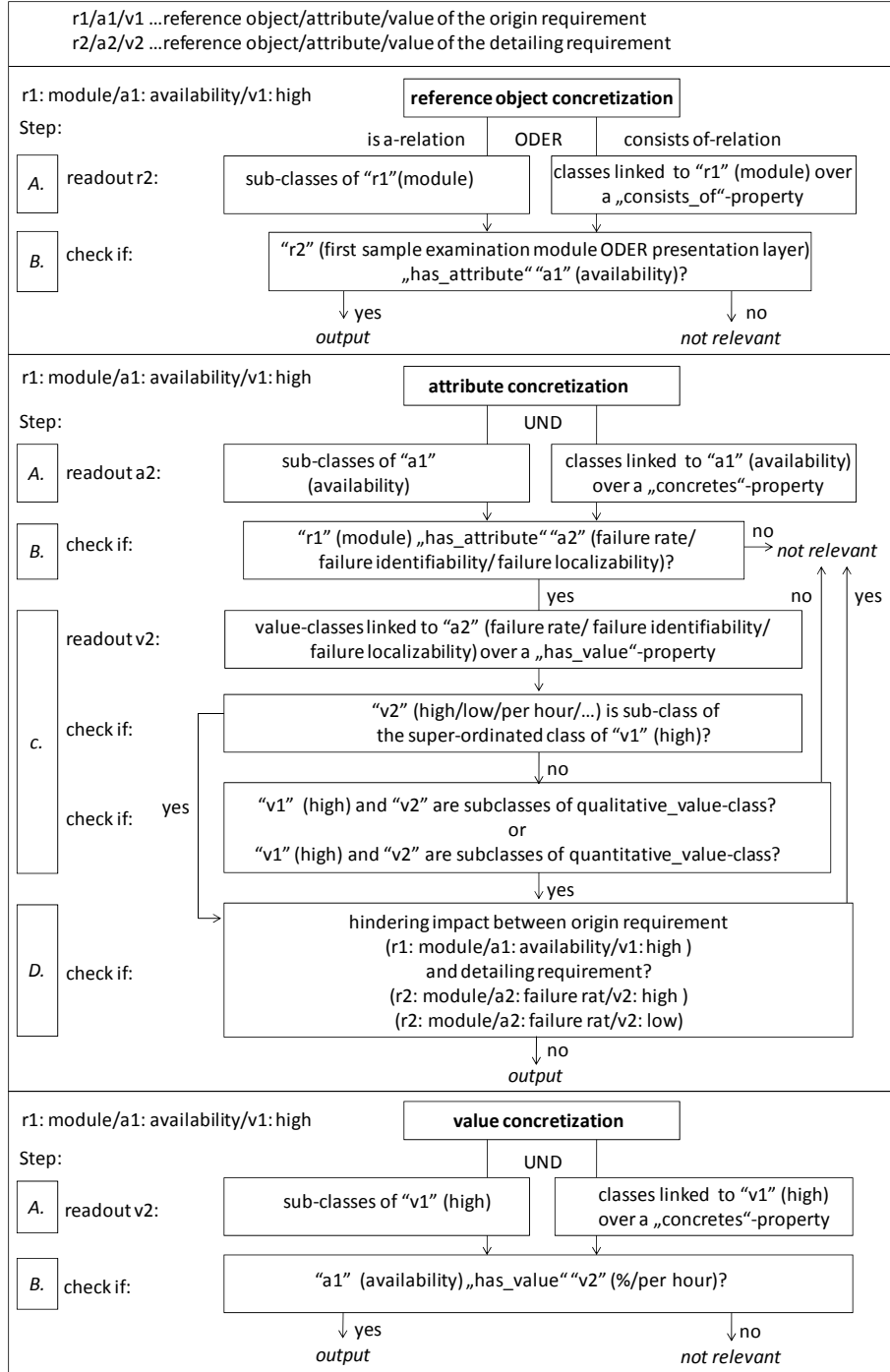


Fig 6. Procedures of the method for concretization.

Reference object concretization

During the reference object concretization the reference object of an origin requirement becomes more concrete and the attribute as well as the value remain the same. At first (**step A**) the method checks if the stakeholder demands a „is a“- or a „consists of“-relation between the origin reference object and the detailing reference objects. A „is a“-relation between two reference objects is defined by the structure of the reference object-class-hierarchy. If the stakeholder chooses this kind of relation, all reference objects which are a sub-class of the reference object-class of the origin requirement will be identified. In the case of the given example (fig. 1) this would be the reference object “first sample examination module” (fig.5). If the stakeholder decides for a „consists of“-relation, the method detects all reference object-classes to which a „consists_of“-property, coming from the origin “reference object“-class, points. Considering the given example, a “consists_of“-property points to the class “presentation layer” (fig. 5).

In order to provide only sensible requirements to the stakeholder, the method checks for each refining reference object from step A, whether it could potentially assume the attribute of the origin requirement (**step B**). This is the case with reference objects, whose ontology-class is connected with the attribute-class of the origin requirement over a „has_attribute“-property. The detailing reference objects of the given example do comply with this condition and can be therefore provided to the stakeholder (fig. 7).

Origin requirement:	<u>Modules</u>	<u>Availability</u>	I High
Detailing requirement („is a“-relation):	I First sample examination module	<u>Availability</u>	I High
Detailing requirement („Consists of“-relation):	I Presentation layer	<u>Availability</u>	I High

Fig. 7. Results of the reference object concretization.

Attribute concretization

During the attribute concretization the attribute is described more detailed (Humpert95) and the reference object remains the same. Although the value stays on the same abstraction level as the origin value, in some cases the value has to change, in order to support the goal of the super-ordinated requirement. In the first step (**step A**), the method identifies all attribute-classes within the ontology which are a sub-class of the super-ordinated attribute-class of the considered requirement or which are connected with the super-ordinated attribute-class over a “concretes“-property. For our given example these are the attribute classes “failure rate” (sub-class) and “failure identifiability” as well as the “failure localizability” (concerning the connection over “concretes“-properties).

In order to obtain a meaningful output the method checks (**step B**) whether the reference object of the origin requirement can have the more concrete attributes which were identified within step A. Therefore, it will be checked whether these attribute-classes are connected over a “has_attribute”-property to the “reference object”-class of the abstract requirement. This precondition is fulfilled concerning the attributes identified in step A of the given example (fig. 5). In order to ensure the achievement of the objective of the super-ordinated requirement, the value of the detailed requirement must perhaps change. In this case, the value of the detailing requirement remains on the same abstraction level of the origin value.

In order to identify an appropriate value the method (**step C**) determines the value-classes to which each detailing attribute is linked over a „has_value“-property. For example, the attribute “failure rate” is linked with the value-classes “high”, “low” and “per hour”. Afterwards it has to be checked if these identified value-classes are sub-classes of the super-ordinated class of the origin value-class (“high”). Values which comply with this are on the same abstraction level as the origin value. For example the classes “high” and “low” are sub-classes of the super-ordinated value class of the origin value.

If this is the case, it has to be checked (**step D**) if there exists a hindering impact between the origin requirement and the detailing requirement. This can be recognized by applying a method for the identification of hindering impacts between requirements. For an in-depth description of this method see [15]. If e.g. the requirement on a high failure rate is implemented this has a negative impact on the realization of the requirement on a high availability. In this case a high failure rate is inappropriate for detailing the origin requirement. If there is no hindering impact between two considered requirements, then the detailing requirement can be presented to the stakeholder. There is no negative impact between the origin requirement and the requirement on a low failure rate. Thus, the requirement on a low failure rate can be provided to the stakeholder (fig. 8). Anyway, there occur also detailing attributes whose potential values are not a sub-class of the super-ordinated class of the origin value-class. In this case, for each value has to be checked whether the detailing value and the value of the origin requirement are part of the “qualitative value”-class-hierarchy or if both classes are part of the “quantitative value”-hierarchy. Therewith it can be checked if both values are located on the same level of abstraction. Values which are on the same level as the origin value must be analyzed concerning their potential hindering effects on the super-ordinated requirement. Only if there is no hindering impact, the detailing requirements can be made available to a stakeholder. For our given example the method identifies a set of the more concrete requirements shown in figure 8.

Origin requirement:	<u>Modules</u>	<u>Availability</u>	I High
Detailing requirement:	<u>Modules</u>	I Failure rate	I Low
Detailing requirement:	<u>Modules</u>	I Failure identifiability	I High
Detailing requirement:	<u>Modules</u>	I Failure localisability	I High

Fig. 8. Results of the attribute concretization.

Value concretization

Only the value of a given requirement changes during the automatic value concretization. In **step A**, the method indicates all „value“-classes that are a sub-class of the origin value-class or that are connected to the origin value-class over a “concretes”-property. In the given example coming from the value-class “high” a “concretes”-properties point to the value-classes “%” and “per hour”. Afterwards (**step B**), it is checked whether the attribute of the origin requirement could potentially assume the refining values, which were determined in step A. This is fact when a “has_value”-property coming from attribute-class of the origin requirement points to the refining value-classes. Only these values, which comply with the mentioned condition, will be presented to the stakeholder. In the event that it is a quantitative value, only the unit of a value will be provided to the stakeholder. The stakeholder must add an appropriate numerical value to the unit, in order to complete the value of the refining requirement. Considering the given example, a “has_value”-property is defined between the classes “high” and “%”. Consequently, the detailing requirement “Modules”, “Availability”, “per hour” is no appropriate requirement in order to detail the sub-ordinated requirement. Just the detailing requirement shown in figure 9 can be provided to the stakeholder.

Origin requirement:	<u>Modules</u>	<u>Availability</u>	I High
Detailing requirement:	<u>Modules</u>	<u>Availability</u>	II %

Fig. 9. Results of the value concretization.

Selection of appropriate refining requirements

While the refining requirements were provided to a stakeholder he has to select those requirements which are in sum adequate, to fulfill their super-ordinated requirement. Afterwards, the refining requirements will be stored. This leads to the establishment of a requirements hierarchy. During the next step of application of AHP alternatives in form of possible solutions must be added to the requirements hierarchy.

Afterwards by weighting the requirements a prioritization of alternatives can be obtained. Thus solutions which can best implement the given requirements have been identified.

5 Summary and forecast

The work presented herein describes a computer-aided method for the knowledgebase construction of a requirements hierarchy. This method automatically provides requirements which can formulate a considered requirement more precisely to the stakeholder. Due to that the stakeholder has the possibility to substitute his abstract requirement by all or by several of these detailing requirements. This allows an optimization of expressing requirements and enables a secured deriving of concrete requirements. It becomes traceably which concrete requirements must be actually fulfilled in order to fulfill the super-ordinated requirement. An ontology serves as the knowledge base for the presented method. This ontology contains knowledge about potential requirements and knowledge of which requirements can potentially refine another requirement. Due to the method the whole amount of requirements can be captured and structured in form of a hierarchy. All in all the method supports the appliance of AHP because the quality of the hierarchy can be improved.

During the construction of a hierarchy for the appliance of AHP it has to be secured, that the criteria being on the same level should not have relationships between each other [16]. But in most cases an independency of criteria cannot completely ensured. Extended approaches for the handling of dependencies during the appliance of AHP already exist. However, due to the high amount of requirements there is a high risk that dependencies between requirements remain undiscovered. The above mentioned method for the detection of impacts between requirements allows the detection of hindering and supporting dependencies between requirements. It has to be investigated in the future if this method is sufficient or adaptable in order to identify all relevant dependencies between the requirements during the appliance of AHP. Moreover, it must be clarified how the different forms of requirements dependencies should be handled during the appliance of AHP.

Acknowledgements

The authors wish to thank the Deutsche Forschungsgemeinschaft (DFG) for supporting their work within the framework of the Collaborative Research Centre 696.

References

1. Saaty T.-L., Vargas L.-G.: Models, methods, concepts & applications of the analytic hierarchy process, Springer (2000)
2. Xie, M., Tan, K.C., Goh, T.N.: Advanced QFD Applications, United States: ASQ Quality Press (2003), pp. 216.
3. Navneet B.: Strategic decision making : Applying the analytic hierarchy process / Navneet Bhushan and Kanwal Rai. - London : Springer (2004). - IX, 172 S. : graph. Darst. : 24cm. - (Decision engineering) ISBN 1-85233-756-7
4. Daniel J. Power: Decision Support Systems: Frequently Asked Questions
5. Saaty, T.: How to make a decision: The Analytic Hierarchy Process. European Journal of Operational Research no. 48: p. 1990
6. Hepler, C.; Mazur, G.: „Finding customer delights using QFD“; Tagungsbeitrag: 18th Symposium on QFD (2006)
7. Refflinghaus R.: Einsatz des Analytischen Hierarchie Prozesses zur Vorbereitung der kundenspezifischen Eingangsgrößen eines Quality Function Deployments, Sonderforschungsbereich 696: Forderungsgerechte Auslegung von intralogistischen Systemen– Logistics on Demand Technical Report 0901, ISSN 1867-3473 (2009)
8. Childs D., Serino D., Bartlett M., Stover S.: DREAM/QFD to Re-design Staff Service Excellence at Rutland Regional Hospital Systems, In Proceedings of the 16th International Symposium on QFD, Portland/USA (2010)
9. Jörg, M.-A., 2005: Ein Beitrag zur ganzheitlichen Erfassung und Integration von Produktanforderungen mit Hilfe linguistischer Methoden. Aachen: Shaker Verlag (2005) ISBN: 978-38322-4032-5.
10. Kamrani, A.-K., Nasr E.A.: Engineering Design and Rapid Prototyping, New York: Springer (2009)
11. Jung, C.: Anforderungsklä rung in interdisziplinärer Entwicklungsumgebung, München: Dr. Hut (2006)
12. Maletz M.: Integrated requirements modeling: A contribution towards the integration of requirements into a holistic product lifecycle management strategy, Kaiserslautern: Technische Universität Kaiserslautern (2008) ISBN 978-3-939432-92-0
13. Naumann F.: Quality-driven query answering for integrated information systems, Number 2261 in Lecture Notes in Computer Science. Springer (2002)
14. Horridge M., Knublauch., Rector A., Stevens R., Wroe C.: Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools. Edition 1.0. The University Of Manchester (2007)
15. Crostack, H.-A.; Kolbe, C.; Refflinghaus, R.: Computer-aided method for automatic identification of effect relations between requirements on an intra-logistics facility. In Proceedings of the 16th International Symposium on QFD, Portland/USA (2010)
16. Meixner O., Haas R.: Wissensmanagement und Entscheidungstheorie, Facultas, Wien, (2010)

Construction and Evaluation of an Algorithmic and Distributed Prioritization Method

Annabella Loconsole^{1,3}, Hannes Gruber², Adrian Nae², Björn Regnell³,

¹ Department of Computer Science, Malmö University, Sweden

² Flygprestanda AB, Malmö Sweden

³ Department of Computer Science, Lund University, Sweden

Annabella.Loconsole@mah.se

{ic05hg7, c03an}@student.lth.se, {Björn.Regnell, Annabella.Loconsole}@cs.lth.se

Abstract. Prioritization of requirements and tasks is an important activity of the development process; it can however be a very complex and time-consuming activity. In this paper we present MAAD (Method for Agile, Automated and Distributed prioritization) an algorithmic prioritization method. The method was developed at Flygprestanda AB, which needed a systematic and automated way of prioritizing requirements and work tasks. The method is constructed based on existing prioritization methods, focusing on minimizing the end-user time-consumption. Factors considered in the priority calculation are Urgency, Benefit, Cost, and Type. Furthermore, in order for the method to suit different companies and projects the importance of the different factors is adjustable. An evaluation was performed, which showed that MAAD prioritization was easier-to-use, less time-consuming, more accurate, more scalable and the prioritization method most suitable for Flygprestanda compared to Wieggers method and the prioritization method used at the company.

Keywords: prioritization, factor, weight, algorithm, urgency, benefit, cost and type

1 Introduction

Which is the most important task at hand? This is the central question that a prioritization method aims at answering. Prioritization must be a part of every decision-making process. In the context of requirements engineering, prioritization usually means deciding the order in which requirements shall be implemented. Widening the perspective, the same principles hold for any form of work, every kind of work task must be evaluated and planned accordingly. Exactly how the prioritization is done varies between different persons and companies. Usually it is done through subjective judgments of the factors that affects, and are affected by, the decision at hand.

Prioritization can however be very time consuming, especially when dealing with large amounts of information. During the prioritization process both short-term and long-term effects of information items such as requirements, tasks and requests must be evaluated. This is particularly complicated if the items affect each other.

Another issue when performing the prioritization is that the knowledge required to prioritize usually is distributed among several persons, for instance employees and customers.

These challenges lead to the following research questions: 1. Is it possible to reduce the individual workload by introducing an automated and distributed prioritization method for requirements, tasks and change requests? 2. Can an automated and distributed prioritization method increase the quality of a company's software and services?

In order to answer these questions, we constructed a Method for Algorithmic, Automated and Distributed prioritization (MAAD) at Flygprestanda AB. The method is based on existing prioritization methods, with the focus on minimizing the end-user time-consumption. Factors considered in the priority calculation are Urgency, Benefit, Cost and Type and in order for the method to suit different companies and projects, the importance of the different factors is adjustable. An evaluation was performed with the purpose of comparing MAAD to Wiegers method [3] and the prioritization method used at Flygprestanda. The evaluation showed that MAAD prioritization, was easier-to-use, less time-consuming, more accurate, more scalable and the prioritization method most suitable for Flygprestanda.

The paper is structured as follows: in the next Section we briefly describe different requirements prioritization methods, in Section 3 it is described the context of this study. In Section 4 we present our prioritization method. An evaluation is described in Section 5. Finally discussion and conclusions are presented in Section 6.

2 Requirements Prioritization Methods

In this section we describe several techniques that are available in the literature used to prioritize requirements. An overview of prioritization factors can be found in [1].

The Analytic Hierarchy Process (AHP) was first introduced by Saaty [16]. The basic idea of this method is to determine a ratio scale³ list of items by comparing all possible pairs (pair-wise comparison). The comparison complexity of $n(n - 1)/2$ which results in 45 comparisons for 10 requirements, is the major disadvantage of the AHP method, but can be lowered by exploiting machine learning techniques [6], [7], [8].

Case-Based Ranking (CBRank) is a method similar to AHP that was designed specifically to use machine learning techniques to overcome the scalability problems in AHP. Another difference with CBRank is that the user has to specify a strict preference, i.e. decide which of the compared items is the most important [7], [8], [9]. Other methods that use pair-wise comparisons are Bubblesort that has the same comparison complexity as AHP and Binary Search Tree (BST) data structure [10].

In the Cumulative voting technique, each stakeholder gets an amount of imaginary money or points that can be spent on the different requirements, tasks or requests that are subject to prioritization. This technique is used in the Hundred-dollar-test where all stakeholders are given 100 imaginary dollars to distribute among requirements [11]. Variations have been tested with larger amounts of money to better suit prioritization of larger sets of requirements [12].

³ See [5] for a definition of nominal, ordinal, interval, and ratio scales.

When performing prioritization using cumulative voting, an issue could be that stakeholders can use different voting strategies and tactics.

Table 1. Comparison of prioritization methods.

Method	Technique(s)	Factors considered	Result type	Reference
AHP	Pair-wise comparisons	Priority	Ratio	[16]
Binary Search Tree	Pair-wise comparisons	Priority	Ordinal	[14]
Bubblesort	Pair-wise comparisons	Priority ⁴	Ordinal	[14]
CBRank	Pair-wise comparisons	Cost, benefit ⁵	Ratio	[7]
Hundred-dollar-test	Cumulative voting	Priority	Ratio	[11]
MoSCoW	Numerical assignment	Priority	Nominal	[15]
Planning game	Numerical assignment, ranking	Effort, priority	Ordinal	[4, 13]
Top-ten	Ranking	Priority	Nominal	[2]
Wiegiers method	Numerical assignment	Benefit, penalty, cost, risk	Ratio	[3]
MAAD	Numerical assignment	Urgency, Benefit, Cost, Type	Ratio	[1]

Numerical assignment is a common technique where items are put into different priority groups. Often three or four different groups are used, such as Low, Medium, High or as in the MoSCoW method: Must have, Should have, Could have, Won't have. Numerical assignment is used in eXtreme Programming as part of the Planning game when the contents of the next release are planned by the developers and customer [13]. The items, called stories, are placed in different piles according to priority and risk and can thereafter be more easily ranked within the groups [4]. The Top-ten method consists in ranking only the 10 most important items at the time are identified, the rest are left until the next time the prioritization is performed [2]. Wiegiers method also uses numerical assignment for estimation of benefit, penalty, cost and risk [3]. The different factors are estimated for every item and are assigned a value ranging from 1 to 9. The priority score is calculated according to the following formula:

$$Priority = \frac{Benefit\% * Benefit\ weight + Penalty\% * Penalty\ weight}{Cost\% * Cost\ weight + Risk\% * Risk\ weight}$$

Table 1 summarizes the described methods. It also includes our MAAD prioritization, which is presented in section 4. Among these, only the Wiegiers method can be applied in the company Flygprestanda since it considers several factors. Therefore it will be compared with MAAD during the evaluation.

⁴ The method supports any factor but only one at a time.

⁵ The method can combine up to two factors at a time.

3 Context

The method described in this paper has been developed for the company Flygprestanda AB, which is specialized in airport analyses and performance calculations within the aviation industry. The company has currently about 60 employees whereof half of these are software developers. The typical customer is a small to medium sized airline company, which relies on the services provided by Flygprestanda for their daily operations.

Currently about 7000 to 10000 requirements and change requests are handled each year, distributed over a few hundred software and service projects, but these numbers will increase as the company grows. The work process at Flygprestanda is agile in the sense of iterative work, relatively short release cycles and a clear focus on producing code rather than extensive documentation [1]. Differently from the waterfall development, iterative work demands regular re-prioritization.

The current prioritization method at Flygprestanda is ad hoc. The customer prioritize themselves their requests and error reporting. The problem is that almost every request and report is assigned the highest priority by the customer. This causes distrust in the set priorities and it makes it hard to recognize which request that should be handled first. This is a widespread and well known problem in requirements prioritization resulting in disbelief and in limited usefulness of the set priorities [2], [3].

Our prioritization method MAAD, described in the next section, will be integrated in Scope which is a new software system recently introduced in the company. Scope is developed as an in-house project whose main objective is to help manage and plan company projects. The system also handles the information flow within the company as well as between the company and its customers. The system is built to handle *nodes*, a data structure which serves as a container for different kind of information. The information could for instance specify requirements, tasks, and change requests. A *project* is defined as a set of nodes, which can change over time. Every project has its own set of nodes.

4 The Prioritization Method MAAD

According to the company's needs, the prioritization method shall be implemented as an algorithm in Scope and be able to handle several factors. It shall also allow for adjustments since the importance of different factors to different company departments or projects might vary.

The method that we have created has been integrated in Scope, which is a web based client-server application. The priority calculation is performed on the server-side by the algorithm before a node is sent to the client. The graph in Figure 1 visualizes the priority-related data flow in Scope. The data flow paths are explained in the following list.

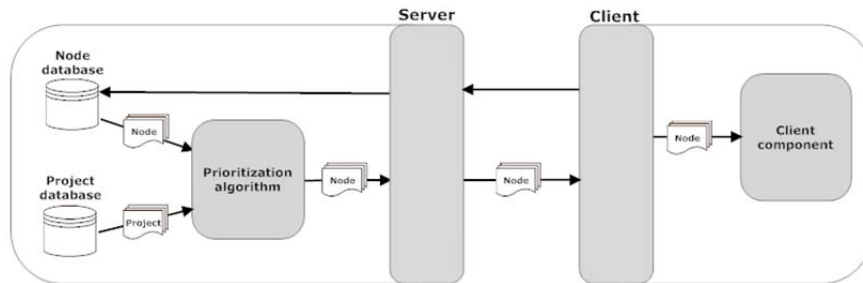


Fig. 1. Priority-related data flow in Scope when a client requests nodes from the server.

1. The client sends a request to get a set of nodes from the server.
2. The nodes are extracted from the node database and every node is, in turn, handed to the prioritization algorithm.
3. The algorithm retrieves parameters from the node (please refer to [1], chapter 3 for a definition of parameters).
4. The algorithm requests the data structure of the project which the node belongs to using the node parameter QM number (the quality management number that identifies the node project).
5. The node project is extracted from cache or from the project database.
6. The algorithm collects parameters from the project.
7. The algorithm calculates and sets the priorities parameters in the node data structures.
8. The set of nodes is thereafter sent to the client.

The stages 3-7 in the above list are repeated once for every extracted node. No significant negative performance impact due to integration of the prioritization algorithm has been experienced at the time of writing.

Table 2. Factors and weights.

Factors and weights	Based on parameter(s)
Urgency factor	Effort, Deadline date or Target date, Current date
Benefit factor	Severity or Value
Cost factor	Effort
Type factor	Type, Enhancement weight, Defect weight, NCR ⁶ weight, Task weight
Urgency weight	Urgency weight, CB weight
Benefit weight	Benefit weight, Cost weight
Cost weight	Benefit weight, Cost weight
CB weight	Urgency weight, CB weight

⁶ NCR means Non-Conformity Report. A report used to describe a serious fault and the immediate and preventive actions taken to correct it.

Our algorithm calculates node priority based on four main factors; Urgency, Cost, Benefit and Type. The algorithm is made highly adjustable by the use of different weights. The factors and weights used by the prioritization algorithm are presented in Table 2 as well as the node and project parameters that they are based on. To distinguish factors, weights, parameters, constants and variables the following notation will be used:

- Factors are named F_x , where x is the name of the factor.
- Weights are named W_x , where x is the name of the factor that the weight corresponds to.
- Parameters are named P_x , where x is the name of the parameter.
- Constants are named with unique lowercase letters.
- Variables are named with unique capital letters.

The following formulas are used to calculate the priority of a node. These formulas are based on the design decisions and on the literature study both described in [1], where the details of the algorithmic prioritization method can also be found; for limited space reasons a brief overview is given here:

Table 3. Formulas used to calculate the prioritization.

Formula Num	Formula
1	$Priority = \left(F_{Urgency} * W_{Urgency} + \frac{F_{Benefit} * W_{Benefit}}{F_{Cost} * W_{Cost}} * W_{CB} \right) * F_{Type} .$
2	$F_{Urgency} = \frac{c_1}{c_2 + b * H^2} .$
3	$F_{Benefit} = P_{Severity} \text{ OR } P_{Value} .$
4	$F_{Cost} = k * P_{Effort} + m .$
5	$W_{Urgency} = \frac{P_{Urgency weight}}{P_{Urgency weight} + P_{CB weight}} .$
6	$W_{CB} = \frac{P_{CB weight}}{P_{Urgency weight} + P_{CB weight}} .$
7	$W_{Benefit} = \frac{P_{Benefit weight}}{P_{Benefit weight} + P_{Cost weight}} .$
8	$W_{Cost} = \frac{P_{Cost weight}}{P_{Benefit weight} + P_{Cost weight}} .$
9	$F_{Type} = \frac{T}{P_{Enhancement weight} + P_{Defect weight} + P_{NCR weight} + P_{Task weight}} + c .$
10	$H = \text{Work hours until Deadline or Target date} - \text{Effort} .$
11	$c = 1 - \frac{1}{\text{number of types}} = 1 - \frac{1}{4} = 0.75 .$

Urgency factor. The Urgency factor in formula 2 is based on the multi-user specified node parameter Effort, the user specified Deadline or Target date and the Current date. Every Effort choice has a corresponding numerical value, which is the matching number of work hours. The choices are presented below, with their numerical values in parenthesis. The default value, which is used in the calculation if no user has estimated Effort, is presented in bold.

1 day (8), 2 days (16), 3 days (24), 1 week (40), 2 weeks (80), 3 weeks (120)
1 month (160), 2 months (320), 3 months (480)

The variable H , defined as the remaining time (in work hours) to when work has to begin, is calculated according to the principle shown in formula 10 in Table 3.

The Deadline date is used to calculate H if it is before the Target date. The Target date is used to calculate H if it is before the Deadline date. When one or more users have estimated Effort, the average of all estimations is used in the calculation in formula 10. The calculation of course has to be adjusted to consider the number of work hours per workday, the number of workdays per work week and so on. A lower limit of 0 is enforced on the H value to ensure that the Urgency factor in formula 2 stays at its maximum value when Deadline or Target date is passed. The priority difference between items that have passed their deadline or target dates will therefore entirely depend on their benefit, cost and type factor values.

The ratio c_1/c_2 specifies the maximum resulting value of formula 2 and this ratio, in combination with b , decides how fast the value will increase when H approaches 0. The actual values of c_1 , c_2 and b used in the prioritization algorithm are presented below.

$$c_1 = 10000, c_2 = 1000, b = 0.5 .$$

These values were chosen to limit the maximum value to 10 and to achieve a priority increase curve. This results in a period of approximately two weeks, where priority increases. Two weeks, which is a short term planning, correspond to the iteration length of the iterative development of the company. This should be well enough for an employee to change focus and switch work task.

Benefit factor. The Benefit factor in formula 3 in Table 3 is based on the user specified node parameter Severity or Value depending on the node parameter Type. The Severity and Value choices have corresponding numerical values, which are used in the priority calculation in formula 1. The choices are presented below, with their numerical values in parenthesis and the default choice presented in bold. Severity choices for flight-specific projects and node types Defect and NCR are the following:

- **None (1)**, • No Op impact (2), • Minor Op impact (3),
- Major Op impact (5), • Ac on ground (7), • Fleet on ground (10)

Severity choices for non-flight-specific projects and node types Defect and NCR are the following:

- **Minor frustration (1)**, • Major frustration (3), • Minor functionality loss (5),
- Major functionality loss (7), • Complete functionality loss (10)

Value choices for all projects and node types Enhancement and Task are the following:

- **Trivial improvement (1)** • Minor improvement (3), • Normal improvement (5), • Major improvement (7), • Essential improvement (10)

The values we have chosen for the different Severity and Value levels represent the exponential increase of criticality or value that is the “general opinion” of the Severity and Value levels at Flygprestanda.

Cost factor. The Cost factor in formula 4 in Table 3 is based on the multi-user specified node parameter Effort. The Effort choices and their corresponding numerical values are described above (see the description of Urgency factor) and once again it is the average of the estimations that is used in the calculation unless no estimations have been made. In that case the default value is used. The Cost factor is calculated using a linear equation where k decides how fast the score increases and m decides the starting value. The actual values of k and m used in the prioritization algorithm are presented below.

- $k = 0.003$
- $m = 1$

These values were chosen to limit the maximum value of the Cost factor to approximately 2.5. This is the case when the Effort average is at its maximum (480 hours). This limit was introduced in Scope to prevent high effort times to decrease the combined Cost-benefit priority to such a low level that items are neglected.

Urgency and CB weights. The Urgency weight in formula 5 and the CB weight (Cost Benefit weight) in formula 6 are based on the user specified project parameters Urgency weight and CB weight. The weights are used in the priority calculation in formula 1 to achieve a suitable balance between main factor Urgency and the combined main factors Benefit and Cost.

The parameter choices have corresponding numerical values that are used when calculating the weights. The choices are presented below, with their numerical values in parenthesis and the default choice (for both Urgency weight and CB weight) presented in bold.

- Very low (0.2), • Low (0.35), • **Medium (0.5)**, • High (0.65), • Very high (0.8)

The maximum values of formula 5 and formula 6 are thus 0.8 and the minimum 0.2. This means that it is not possible to totally disregard a main factor. The range of 0.2-0.8 is specific for the implementation in Scope.

Benefit and cost weights. The Benefit weight in formula 7 and the Cost weight in formula 8 are based on the user specified project parameters Benefit weight and Cost weight. The weights are used in the priority calculation in formula 1 to achieve a suitable balance between the main factors Cost and Benefit. The default and available parameter choices and their numerical values are the same as Urgency and CB weights for both Benefit weight and Cost weight. The maximum and minimum values of the weights are therefore also the same (0.8 and 0.2 respectively). The combined effect of the Benefit weight and Cost weight on the Cost-Benefit part of the priority calculation thus ranges from 0.25 to 4.

Type factor. The Type factor in formula 9 in Table 3 is based on the user specified node parameter Type and the user specified project parameters Enhancement weight, Defect weight, NCR weight and Task weight. The Type factor includes the weighting between different node types, and behaves similar to a weight for the whole priority calculation, as seen in formulas 1 and 9. Therefore no additional weight for the Type

factor was added. The node parameter *Type* determines which of the project parameters to use as the value *T* in formula 9.

$$T \in \{P_{\text{Enhancement weight}}, P_{\text{Defect weight}}, P_{\text{NCR weight}}, P_{\text{Task weight}}\}.$$

The available choices and their numerical values for the project parameters are the same as those presented for Urgency and CB weights. However the parameters do not have the same default choices. The default choice for each project parameter is presented below.

- Enhancement weight: Medium (0.5), • Defect weight: High (0.65),
- NCR weight: High (0.65), • Task weight: Medium (0.5)

This means that, by default, defects and NCR's will get slightly higher priorities than enhancements and tasks.

The constant *c* in the *Type* factor calculation in formula 9 ensures that the resulting value is close to 1. This is important because it is used to slightly adjust the Priority value. The value of *c* was determined using the formula 11 in Table 3. From the formula can be deducted that the lowest possible *Type* factor value is 0.827 while the highest possible value is 1.321.

5 Evaluation

We performed an evaluation of our method. Aspects that were evaluated are Time consumption, Ease-of-use, Expected accuracy, Perceived accuracy, Scalability, Suitability Overall best method. The evaluation consisted in asking six employees at Flygprestanda to prioritize a set of items chosen from a real software project. We choose items for which we expected high medium and low values of prioritization to be represented.

The prioritization was made with 3 different methods in order to compare MAAD prioritization with the method currently used at Flygprestanda (which is ad-hoc, as described in section 3) and with the Wiegiers method. We have chosen to compare MAAD with the Wiegiers method since we believe it would be Flygprestanda's only alternative to MAAD. The other methods shown in Table 1 are not suitable since they do not consider several factors.

The participants were chosen in order to have different project roles represented: two customers, one manager and three developers. Two of the authors performed a pilot test of the study.

The instruments used in the study were three paper forms used to prioritize the set of project items, one form for each prioritization method. The forms contained item name, description, deadline, and prioritization factors for each item in the set. Two questionnaires were used to evaluate the aspects mentioned above. Except for the time consumption that was estimated in minutes, the other factors were estimated on a 5-point Likert scale, where 1 corresponds to the text "I strongly disagree" and 5 corresponds to the text "I strongly agree". We also interviewed the participants in order to get subjective opinion and experience on using the different prioritization methods, focusing on the use of different factors, scales, and expected impact on Flygprestanda. The answers were annotated using pen and paper.

All the forms, questionnaires and questions used during the semi-structured interviews can be found in [1].

The participants were not told which method was developed by the authors and the methods were referred to as methods 1, 2 and 3. After the prioritization was performed, the participants were asked to fill in the questionnaires. The interviews were performed later once the results of the prioritization had been calculated.

Table 4. Aspects measured during the evaluation with questionnaires.

Aspects measured	MAAD	Wieggers method	Traditional
Time consumption	5 min 25 sec	7 min 35 sec	7 min 45 sec
Ease-of-use	4.33	2.50	3.00
Expected accuracy	4.00	3.50	3.00
Perceived accuracy	3.83	3.00	3.67
Scalability	4.33	3.83	1.33
Suitability	4.00	3.33	2.00
Overall best	5 particip.	0 particip.	1 particip.

Table 4 shows the values of time consumption in minutes for the three methods. The values of the other aspects evaluated are mean values of a normalized Likert scale. The evaluation showed that MAAD prioritization had the lowest time consumption, was the easiest to use, and produced the most accurate results. The evaluation also showed that MAAD prioritization was the most scalable of the methods evaluated. Most of the participants in the evaluation considered MAAD prioritization to be the overall best, and the prioritization method most suitable for Flygprestanda. The results of the questionnaires were also confirmed during the interviews.

6 Conclusions and Future Works

In this paper we have presented the MAAD prioritization method that was developed to support agile, automated and distributed prioritization of requirements, change requests and work tasks at Flygprestanda AB. The method has been implemented as an algorithm in the Scope system and the algorithm calculates priorities automatically. Several factors (Urgency, Benefit, Cost and Type) influence the priority calculation and their relative importance is adjustable. The algorithm can receive input by several stakeholders, both customers and employees at Flygprestanda. An example of the use of MAAD is presented in [1]. An evaluation was also performed showing that MAAD was the best method among those evaluated.

The positive results obtained suggest that the limited number of factors improved both ease-of-use and time consumption of the method as can be seen in table 4 (time consumption). The automated urgency calculation minimizes the risk of missing deadlines, something that several participants to the interview study recognized as highly important to increase product and service quality.

Another strength of the MAAD method is that prioritization does not have to be performed on a set of items, which is the case in methods such as AHP, CBRank, the Hundred-dollar-test and Wieggers method. MAAD makes it possible to calculate the priority of a single item at any time. It also makes possible to compare the calculated priorities of any items. Furthermore, new requirements can easily be prioritized individually, without the need to compare (as in AHP) each newly added requirements with all others which are already present.

However, there are several issues regarding the MAAD prioritization method that could be further investigated. For instance there could be dependencies among items that can affect priorities. Tasks Risk is also a factor that currently is not considered in MAAD prioritization. Including a risk factor would negatively affect both time consumption and ease-of-use but possibly increase the quality of the results. Further investigation could be done on whether a risk factor would increase quality.

Uncertainty is a factor that is not included in the priority calculation in MAAD. An indication of uncertainty regarding effort estimations is however available in Scope since all estimations are visible. Uncertainty should probably not directly affect the priority of an item, but it could be further visualized in Scope by automatically triggering a warning if the variance of effort estimations is high. This would not have any significant negative effects on time consumption or ease-of-use but could possibly lead to increased quality.

Items that do not have specified deadline or target dates are usually wanted as soon as possible, i.e. the sooner they are completed the better. This could call for automatically increasing the priority of “old” items in order to prevent these from being “forgotten”. This is achievable without any negative effects on time consumption and ease-of-use. It is however risky since it could possibly result in old, originally low-priority, items to be prioritized over new items that actually are more important. A thorough investigation should be performed before anything like this is implemented in order to assure that it does not cause quality to decrease instead of increase.

A way to specify and visualize the progress of an item is another open issue. This could possibly have a positive effect on urgency calculation and the quality of the results. It could also improve the possibility to track progress of projects. On the other hand it would increase time consumption and decrease ease-of-use as it would require users to frequently update node parameters.

Two factors, Urgency and Cost, are currently based on the mean value of the effort estimations. This could be extended by including a weight for estimators, in order to take into account their estimation accuracy. The estimation accuracy of employees is however hard to determine and it is questionable if this way of combining effort estimations would increase the quality of the priority result.

The impact of the MAAD prioritization method on product and service quality is something that can be evaluated further at Flygprestanda as the use of the Scope system within the company increases. At the time of writing, only few employees at the company use Scope (and therefore MAAD).

Our method has been constructed for the company Flygprestanda and therefore has local validity. Besides investigating further the factors mention above, we plan to test this method in other companies of different size and domain. This will further enhance the validity of our method.

Acknowledgments

We would like to thank the interviewees at Flygprestanda for their dedicated participation in this study, and reviewers of the paper for their valuable comments.

References

1. Gruber, H., and Nae, A., MAAD prioritization A method for agile, automated and distributed prioritization of requirements and work tasks, Master Thesis, Lund University, ISSN 1650-2884, <http://sam.cs.lth.se/ExjobGetFile?id=293> (June 2010)
2. Berander, P., Evolving Prioritization for Software Product Management. Doctoral Dissertation Series No. 2007:07, Blekinge Institute of Technology, (2007)
3. Wiegers, K., First Things First: Prioritizing Requirements, Software Development, vol. 7, no. 9 (1999)
4. Karlsson, L., Thelin, T., Regnell, B., Berander, P. and Wohlin, C., Pair-wise comparisons versus planning game partitioning - experiments on requirements prioritisation techniques, in Proc. 8th Int'l Conf. on Empirical Assessment in Software Eng., Edinburgh, pp. 145--154 (2007)
5. Fenton, N.E. and Pfleeger, S.L., Software Metrics: A Rigorous and Practical Approach, 2nd ed. Boston, USA: PWS Publishing Co. (1998)
6. Karlsson, J., and Ryan, K., A cost-value approach for prioritizing requirements, IEEE Software, vol. 14, no. 5, pp. 67-74 (1997)
7. Perini, A., Ricca, F., and Susi, A., Tool-supported requirements prioritization: Comparing the AHP and CBRank methods, Information and Software Technology, vol. 51, no. 6, pp. 1021--1032 (2009)
8. Avesani, P., Bazzanella, C., Perini, A., and Susi, A. Facing scalability issues in requirements prioritization with machine learning techniques, in 13th IEEE International Conference on Requirements Engineering, pp. 297-305 (2005)
9. Avesani, P., Bazzanella, C., Perini, A., and Susi, Supporting the requirements prioritization process - A machine learning approach, in Proceedings of 16th International Conference on Software Engineering and Knowledge Engineering, Banff, Alberta, Canada, pp. 306--311 (2004)
10. Karlsson, J., Wohlin, C., and Regnell, B., An evaluation of methods for prioritizing software requirements, Information & Software Technology, vol. 39, pp. 939-948 (1998)
11. Leffingwell, D., and Widrig, D., Managing Software requirements: A Use Case Approach, 2nd ed. Boston, Addison-Wesley (2003)
12. Regnell, B., Host, M., Natt och Dag, J., Beremark, P., and Hjelm, T., An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software, Requirements Engineering, vol. 6, no. 1, pp. 51-62 (2001)
13. Beck, K., Extreme Programming explained, 2nd ed. USA: Addison-Wesley (2004)
14. Aho, A.V., Hopcroft, J.E., Ullman, J.D., Data Structures and Algorithms. Addison-Wesley, Reading, MA (1983)
15. Clegg, D., and Barker, R., Case Method Fast-Track: A Rad Approach. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1994)
16. Saaty, T.L., The Analytical Hierarchy Process: Planning, Priority Setting, Resource Allocation (New York: McGraw-Hill), (1980)

Prioritizing business process chains for IT optimization

Norman Riegel¹, Oezguer Uenalan¹ and Thomas Jeswein¹

¹Information System Development Department
Fraunhofer IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{norman.riegel, oezguer.uenalan, thomas.jeswein}@iese.fraunhofer.de

Abstract. In the present economy, where financial resources have become scarce, the interest in optimizing the interactions between public administration and business is greater than ever - on both sides of the public-private fence. A conscious effort has to be made in order to select those processes and services that promise to yield the highest return on investment in terms of efficiency gains, tax revenues or customer satisfaction, for example. Therefore it is paramount for both sides to identify only those business process chains (BPCs) between public administration and business that are most beneficial for both parties in regard to IT-supported optimization. In this paper, we introduce a method which employs prioritization to streamline this process of identifying precisely those BPCs that are most promising for IT optimization and will truly exhibit measurable gains on both the side of public administration and business.

Keywords: Requirements prioritization, requirements prioritization method, business process, business process chain, information system, business process-driven requirements engineering, E-Government, B2G, G2B

1 Introduction

Business organizations constantly strive to improve their internal business processes to become more successful [4]. However, it is not enough to overhaul and align just the internal procedures and routines. Besides business-to-business (B2B) interactions, it is also crucial to look closely at business process chains (BPCs) between business and public administration (called government-to-business (G2B), or business-to-government (B2G)) and to find ways to improve these interactions. The term *business process chain* (between business and public administration) describes the relative length and higher complexity of integrated processes when several business and governmental stakeholders work together to achieve a common goal, disregarding the limitations of their respective organizations. The foundations of BPCs between business and public administration are legal principles that define the business' information obligation concerning the public, respectively the public administration. A BPC therefore describes the functions needed to fulfil legal obligations in order to inform the acting stakeholders from business and public administration [3].

These relations tend to be bound by strict regulations, standards and laws. But they are not set in stone. Especially since the advent of the Internet and the pervasiveness of web technologies, more and more public administrations feel the need to change procedures and thereby improve services for their customers, i.e., for citizens and businesses. However, since financial resources are scarce, a conscious effort must be made to target for improvement only those processes and services that promise to yield the highest return on investment (ROI), for example in terms of efficiency gains, tax revenues or customer satisfaction [6]. It follows that the potential value of changing any BPC must be determined relative to other BPCs. In other words, the task of the prioritization method is to assess the benefit of undertaking IT-supported optimization of B2G (respectively G2B) BPCs for both business and public administration and thus to create a ranking order for BPCs. A method is clearly needed that provides guidance for the stakeholders regarding the prioritization and the integration of the interests of both stakeholder groups. In this paper, we describe a generic prioritization method which addresses these issues. The approach described here addresses all levels of public administration as instigators of administrative rules, issues, and orders, but also companies which are affected by the aforementioned regulatory system. By using this approach, those BPCs are identified, whose IT optimization is anticipated to be most beneficial for both sides. The paper is structured as follows: Chapter 2 gives an overview of the methodological background, which resulted in the approach presented in this paper. Chapter 3 presents our approach on how to prioritize BPCs between business and public administration. Chapter 4 summarizes and provides an outlook on future work.

2 Methodological Background

In 2005 the state government of Rhineland-Palatinate in Germany commissioned a study to determine the BPCs between government agencies and enterprises in certain industries (automotive, chemical, agriculture) in order to demonstrate the high potential for optimization by means of IT, especially web technologies. With respect to our approach described in this paper, there was one relevant observation during this study: enterprises too often do not see or understand the full potential of “E-Government”, i.e. they miss the opportunity to optimize the interconnected links between themselves and public administration. The main reason is a lack of a formal and concerted approach regarding the identification and optimization of BPCs by means of IT [5].

Based on this finding, we therefore developed a preliminary version of the prioritization approach, which was then applied in a feasibility study commissioned by the German Federal Ministry of the Interior. About 200 information and reporting obligations for employers were analyzed and prioritized. The survey among enterprises yielded 17 reporting obligations considered to be most valuable for IT optimization. These 17 reporting obligations were further discussed with five receiving government agencies. Finally, a stakeholder workshop was organized including all involved parties (enterprises and government agencies) as well as additional stakeholders, e.g., expert groups. Its outcome was the identification of the three topmost reporting obligations as a starting point for IT optimization.

This approach proved very valuable since it helped to uncover hidden assumptions on both sides of the fence about the interconnected BPCs and to determine the types of services and processes that should be optimized based on a balanced and objective method [3].

Based on the experience gathered in these two previous studies, we reengineered our methodical approach for the prioritization of BPCs, which is described in the following chapter.

3 The Prioritization Approach

The BPC prioritization basically involves five steps as shown in Figure 1. As input for this approach, a set of preselected BPCs (determined by the concrete project context) to be in focus of the prioritization is taken.

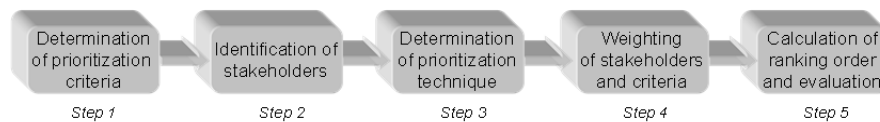


Fig. 1. Steps of the BPC prioritization approach.

Step 1: Determination of quantitative and qualitative prioritization criteria.

Objective criteria (e.g., number of cases) and subjective criteria (e.g., political relevance) are needed to support the prioritization of the BPCs, i.e. to assess the BPCs according to different criteria. Generally, it is hard to find meaningful criteria which serve as basis for prioritization. In our previous studies we already collected a (still incomplete) set of over 20 criteria and created a catalogue for this step of the method (see example in Table 1). The criteria catalogue consists of a description of each criterion, hints on how to elicit or measure the values for this criterion (for objective criteria), as well as some instructions on how further criteria can be collected, for example via empirical studies. As our approach is generic, one has to decide first which criteria are relevant for the concrete project. For example, if BPCs are of interest, where the number of cases are relevant, this criteria will be included in the list of criteria to be prioritized. The selection of criteria should be aligned with the identification of stakeholders in step 2. Only stakeholders that have knowledge about the chosen criteria should participate in the prioritization.

Table 1. Excerpt from the criteria catalogue.

Criterion	Description	Unit	Elicitation method
Frequency	The regularity by which the compulsory registration has to be reported (e.g. quarterly)	1/t	Determination by legal duties or past data

Step 2: Identification of stakeholders.

For the actual prioritization procedure, relevant stakeholders have to be identified who will perform the prioritization according to the selected criteria. Stakeholders can be entire organizations or individual persons. Basically, in the field of BPCs, one can differentiate between business stakeholders and public administration stakeholders. Generally, only companies and public administrations will conduct the prioritization, which are involved in the preselected BPCs. Important stakeholders on the business side are, for example, process owners of the BPC, functional employees who are involved in the BPC, technical staff, and roles responsible for security and legal conformance. Process owners generally are not directly involved in the creation process of a compulsory registration (at best by approval steps), but they are good contact persons to identify other important roles in the organization, which should be involved in the prioritization also.

Step 3: Determination of prioritization technique.

For prioritizing the BPCs, an appropriate prioritization technique has to be chosen, which describes a guideline on how to assign values to the chosen criteria and how to calculate the actual ranking order of the BPCs. In general, the prioritization techniques are independent from the chosen stakeholders, but as mentioned before, it should be ensured that the stakeholders are able to assess the criteria used in the prioritization technique. In the literature, there already exist numerous approaches for the prioritization of alternatives (e.g., [1], [2], [7]). They differ in the way data for individual criteria are determined (different scales, assignment of points, etc.), in the way of the procedure (flat vs. hierarchic), or in the degree of complexity of the procedure. As the techniques are generally not directly applicable for the prioritization of BPCs, we created a collection of state-of-the-art prioritization techniques for our approach, showing advantages and disadvantages as well as modifications that are necessary for computing a ranking order for BPCs. We decided to do this, because different project situations require different techniques that are appropriate. For example, if many BPCs have to be prioritized, some techniques are in favor of others (e.g. through differences in execution time or number of steps in the technique). Also, if tool support is available, some more sophisticated techniques can be used. In the context of BPCs it has also to be regarded for example, that objective and subjective criteria have to be combined in the calculation. Objective criteria might be measured only once for a BPC, while subjective criteria are rated by all stakeholders. Furthermore, certain criteria might only be relevant for one stakeholder group (i.e. assessed by this group). An example of the collection is shown in Table 2.

Table 2. Excerpt from the prioritization technique collection.

Technique	Description	Instructions for use	Necessary extensions/adaptions
Likert scale method (cp. e.g. [7])	The Likert scale method is a simple prioritization technique. It bases on a bipolar scale, where the alternatives are assessed along one aspect. The scale is often used in market research and is an established means for data collection.	The Likert scale method is a good and simple means for prioritization. However, differentiation is limited, which can be disadvantage in for precise assessments. ...	The Likert scale method does not prescribe how to calculate the result of the prioritization itself. After each criteria for each BPC is assessed by each stakeholder, a summation has to be done (including weights), which leads to an overall result for each alternative. ...
...

Step 4: Weighting of stakeholders and criteria.

Where required, stakeholders and criteria have to be weighted, i.e., assigned a weight. The emphasis thus determines the influence of a certain stakeholder or criterion on the overall result of the prioritization. Therefore, a certain value is assigned to every stakeholder, which is included in the actual prioritization. In this context, the stakeholder's influence on the project and his/her importance for the project should be considered. Subsequently, the prioritization criteria determined before are weighted. Most often, weighting is needed to point out the importance of a certain criterion. This step is also dependent on step 3 of the method. Depending on the selected prioritization technique, the value assignment takes place in another mode (relative, absolute, etc.).

Step 5: Calculation of ranking order and evaluation of result.

By means of the concrete approach of the prioritization technique, the values for the criteria that have to be evaluated are determined. On the one hand, this is done by measuring (quantitative evaluation); on the other hand, it is done directly through the stakeholders' judgments. Finally, the prioritization result is calculated, resulting in a ranking order of the BPCs. The ranking order of the BPCs shows the view of the companies and public administrations directly involved. The result is analyzed and interpreted by means of evaluation and comparison by different (groups of) stakeholders or criteria. This means that the results from the enterprises and the public administration are first regarded separately and are combined later. From this, conclusions regarding potential conflicts of interests can be drawn, which can be discussed afterwards. Primarily the actual ranking order is the main result of the prioritization. However, part results of the prioritization could also be interesting. For example, the analysis and separate description of single criteria could give better insights how the result was composed. Also, a sensitivity analysis could be conducted to check the stability of the result of the prioritization, showing how changes to the weights of stakeholders or criteria would influence it.

In order to get an impression that is as broad and valid as possible, the prioritization results have to be verified, negotiated and adapted, if necessary, by involving additional experts and collaterally involved stakeholders (e.g., intermediates and special interest groups). This may be done by means of individual interviews or in workshops.

4 Conclusion and Future Work

In this paper we have presented a generic approach to prioritize BPCs between business and public administration in order to select the best processes for IT optimization in terms of an E-Government roadmap. The approach is based on our previous work in [3] and [5]. It regards criteria relevant for prioritizing BPCs, as well as different stakeholders from business and public administration side. In our future work, we plan to extend our criteria catalogue, to describe how to extend the approach by empiricism and to extend it by experience gained from practice. Furthermore, we want to refine and formalize the negotiation activities in greater detail within the last step of our method. We also want to analyze, how we can align the approach presented in this paper with our prioritization approach described in [4].

Acknowledgements

This work was supported by the project “Pilotierung und Realisierung eines Prozess-Daten-Beschleunigers (PDB) für den Datenaustausch zwischen Wirtschaft und Verwaltung” (Order: B2.28-0004/09/001), funded by the German Federal Ministry of the Interior (BMI).

References

1. Berander, P., Jönsson, P.: Hierarchical Cumulative Voting (HCV) - Prioritization of Requirements in Hierarchies. *International Journal of Software Engineering and Knowledge Engineering* 16, vol. 6, pp. 819--849 (2006)
2. Saaty, T. L.: *The Analytic Hierarchy Process*. McGraw-Hill, New York (1980)
3. Bundesministerium des Innern (ed.): *Machbarkeitsstudie zum Forschungsauftrag „Entwicklung von Prozessketten zwischen Wirtschaft und Verwaltung“ Los 3 „Informations- und Meldepflichten für Arbeitgeber“*, Berlin (2009)
4. Riegel, N., Adam, S., Uenal, O.: Integrating Prioritization into Business Process-driven Requirements Engineering. In: *16th International Working Conference on Requirements Engineering: Foundation for Software Quality. Proceedings of the Workshops CreaRE, PLREQ, RePriCo and RESC*, pp.113-118. Essen (2010)
5. Steffens, P., Grützner, I., Horch, J., Hufen, A., Jeswein, T., Thomas, L.: *Branchenprozesse mit Schnittstelle zur Landesverwaltung Rheinland-Pfalz: Projektabschlussbericht*. IESE-Report; 004.09/D, Kaiserslautern (2009)
6. Rombach, D., Steffens, P.: E-Government. In: Nof, S. Y. (eds.) *Springer Handbook of Automation, Part I*. pp. 1629-1643. Springer, Berlin (2009)
7. Wiegers, K. E.: First Things First: Prioritizing Requirements. *Software Development* 7, vol. 9, pp. 48--53 (1999)

