PhD Theses in Experimental Software Engineering

Vol. 44

Sebastian Adam

Incorporating Software Product Line Knowledge into Requirements Processes

FRAUNHOFER VERLAG

PhD Theses in Experimental Software Engineering

Volume 44

Contact:
Fraunhofer-Institut für Experimentelles Software Engineering (IESE)
Fraunhofer-Platz 1
67663 Kaiserslautern
Telefon +49 631 6800 - 0
Fax +49 631 6800 - 1199
E-Mail info@iese.fraunhofer.de
www.iese.fraunhofer.de

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliografic data is available in the Internet at <http://dnb.d-nb.de>.
ISBN: 978-3-8396-0514-1

D 386

Zugl.: Kaiserslautern, Univ., Diss., 2012

Printing and Bindery:
Mediendienstleistungen des
Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart

Printed on acid-free and chlorine-free bleached paper.

Fraunhofer Information-Centre for Regional Planning and Building Construction IRB
P.O. Box 80 04 69, D-70504 Stuttgart
Nobelstrasse 12, D-70569 Stuttgart
Phone +49 (0) 711 970-2500
Fax +49 (0) 711 970-2508
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

Incorporating Software Product Line Knowledge into Requirements Processes

Beim Fachbereich Informatik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation
von

Dipl.-Inf. Sebastian Adam

Fraunhofer Institut für Experimentelles Software Engineering (IESE)
Technische Universität Kaiserslautern

Berichterstatter:	Prof. Dr. H. Dieter Rombach Prof. Dr. Klaus Schmid
Dekan:	Prof. Dr. Arnd Poetzsch-Heffter
Tag der Wissenschaftlichen Aussprache:	19.12.2012

D 386

Acknowledgment

In the first place, I would like to thank my parents, who offered me an excellent education from the early days of my life. Although they made many sacrifices during the entire time, they never complained but always encouraged me on my way.

Next, I would like to thank all the people who supported, accompanied, and empowered me in doing my PhD during the last four years.

I thank my supervisors Prof. Dr. Dr. h.c. Dieter Rombach and Prof. Dr. Klaus Schmid for their continuous advice and guidance, and Prof. Dr. Stefan Deßloch for accepting to take over the chair of the dissertation committee.

Many thanks also go to my department head Dr. Marcus Trapp and my division head Dr. Jörg Dörr for giving me fruitful feedback all the time. In this regard, I also thank Dr. Christian Webel for allowing me to do research in "his" large research projects ADiWa and EMERGENT, and Dr. Dirk Muthig, who helped me in shaping my topic in 2008.

I would also like to thank all the other people at Fraunhofer IESE who made it possible for me to work in such an inspiring, lively environment. In particular, I would like to thank the people from the (former) requirements engineering group, namely Anne Heß, Norman Riegel, Özgür Ünalan, and Michael Eisenbarth, who became friends rather than just colleagues in the last years. The same holds true for my long-term student research assistant Lena Karjalajnen.

Many thanks also go to all the other colleagues and students, especially those who supported me in my empirical study or in shaping my conceptual model. In this regard, I would like to highlight Dr. Ralf Carbon, Dr. Matthias Naab, Dr. Andreas Jedlitschka, Dr. Martin Becker and Jessica Jung for their advice. Furthermore, I would like to thank Michael Ehresmann from Insiders Technologies, who allowed me to conduct two case studies in his organization and who was a likeable contact right from the early days of my work at Fraunhofer IESE.

For their support during my defense preparation, I thank again Dr. Jörg Dörr, Dr. Marcus Trapp, Dr. Matthias Naab, and Dr. Jens Knodel. Further thanks go to Stephan Thiel and Sonnhild Namingha for supporting me in publishing this thesis.

Finally, I would like to thank all the people in my private life who reminded me that there is more to life than just research and work (even though not in the last months before submission). In particular, I would like to thank my brother Philipp and all my friends from home who accompanied me in the past, such as Stephan, Bianca, Jenni, Marc, Tina, Silke, Carsten, Stefan, Patricia, Uwe, Manuela, and Jasmin.

Abstract

Almost every information system (IS) is nowadays assembled from existing assets instead of being developed from scratch [SLS+09]. However, developing customer-specific systems with reuse in so-called application engineering (AE) projects is often less efficient than expected [DSB05]. One important reason is that state of the art reuse approaches are very inflexible when they have to cope with unforeseen requirements [PKG+08]. The elicitation and negotiation of such requirements is not systematically supported yet [ORR+09], and it heavily relies on experts to assure that these requirements are compatible with the constraints and capabilities of the given reuse assets. In many cases, a tight fit between requirements and reuse capabilities can therefore either not be achieved at all or takes too much time, which reduces the overall development efficiency. Unfortunately, approaches that solve this problem in a satisfactorily way do not exist yet.

To cope with this challenge, this thesis aims at improving the effectiveness of elicitation by providing requirements engineers with better guidance on how to reconcile requirements with the capabilities and constraints of a reuse asset base. More precisely, requirements engineers are systematically guided through the elicitation process, and made aware of the reuse characteristics they have to consider there. As a result, requirements that are hard to satisfy can be detected directly during elicitation sessions, and do therefore not need to be re-negotiated and reworked in tedious iterations.

However, making people aware of reuse characteristics is not easy, as communicating all capabilities in terms of variability models (as commonly used) is hard due to the complexity of industrial reuse asset bases [RGD09] [ORR+09]. The unsolved, scientific problem to be addressed by this thesis therefore deals with the question how knowledge about the reuse asset base of a software product line (SPL) can be systematically extracted, and incorporated into application engineering requirements (ARE) processes and supporting artifacts. The computer science contribution of this work therefore comprises an algorithmic, tool-supported method that guides the identification and translation of reuse characteristics into suitable elicitation instructions. Hence, the thesis approach prescribes different steps for the systematic analysis of a given SPL and the definition of corresponding ARE processes. To make this happen, the approach is based on a conceptual model that expresses the relationships between SPLs and ARE processes, as well as on an issue model and an ARE instructions template that both formalize RE best practices.

While the feasibility of the approach was successfully tested in a case study, the advantages of elicitation instructions defined according to the thesis approach were shown in a controlled experiment with students. Hence, an incorporation of reuse knowledge into ARE processes is basically possible, and the usage of these processes during ARE can increase the elicitation effectiveness significantly.

This dissertation describes the motivation, development, and evaluation of the depicted approach as well as its components and related work. Finally, it gives an outlook on future work that is worth to be done according to our experience made.

Table of Contents

List of Figures

List of Tables

1 Introduction

"One must have learned a lot to know how to ask for something that one does not know."
Jean-Jacques Rousseau

This chapter motivates the context and topic of this thesis, and explains its contributions and benefits. The chapter also presents the research approach as well as an outline of the remaining chapters.

1.1 Context

As a key concept for streamlining software development, reuse has received much attention in recent years. In general, *"reuse is a development approach by which a system can be built from existing components already described, carried out, tested and accepted in past experience"* [GRT00]. In contrast to custom development, in which a system is developed from scratch, reuse-based development therefore promises higher productivity and shorter time to market. Especially in the area of information systems (IS), which highly relies on these quality aspects, almost every system is therefore built in a reuse-based manner today [SLS+09]. This holds mainly true for organizations that are focused on a specific domain, as the reusability of existing artifacts is potentially high in such a context [LJB98].

Among other approaches, software product lines (SPLs) are one promising concept for software reuse [DSB05], and are often considered the most strategic and advanced form with the highest return on software development investment [Mut02].

Definition – Software Product Line (SPL)

"A software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment and that are developed from a common set of core assets in a prescribed way." [SEI08] [CN01]

To make SPLs work, an important aspect is the separation of the phase of domain engineering or family engineering (DE/FE) from the phase of application engineering (AE) as shown in Figure 1.

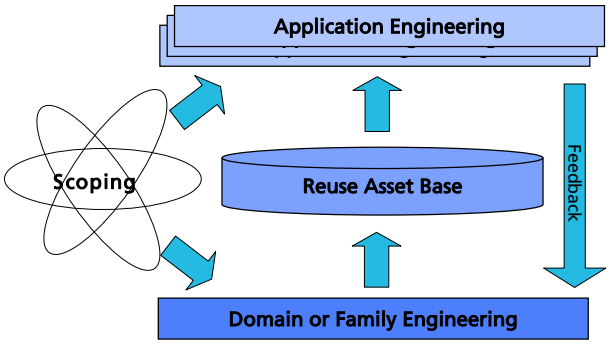


Figure 1. Big picture of product line engineering [Mut02]

While the DE/FE phase aims at developing for reuse (i.e., at building up an SPL's reuse asset base addressing a certain scope), the AE phase aims at developing concrete systems within the defined scope by reusing these assets. Thus, we define AE according to Halmans and Pohl [HP03] as follows:

Definition – Application Engineering (AE)

Application engineering is the phase of product line engineering in which individual, customer-specific software products are developed by selecting and configuring shared assets from an SPL's reuse asset base.

However, as the customization possibilities during AE are restricted due to several architectural decisions already made during DE/FE, the actual development benefits during AE strongly depend on the degree to which customer requirements are already addressed by the existing assets. In this thesis, the requirements that may occur during an AE project are therefore classified into *explicitly anticipated*, *implicitly anticipated*, and *non-anticipated requirements* according to their anticipation during the DE/FE phase[1]. While explicitly anticipated requirements are inexpensive to satisfy because they are implemented as common or variable features within the SPL already, implicitly anticipated requirements typically imply additional development costs, as their details are not known upfront. However, as their core characteristics have already been foreseen during DE/FE, they are known to be basically feasible. In contrast, for non-anticipated requirements, it is typically not possible to estimate their feasibility or costs upfront, as neither their core characteristics nor their details were taken into consideration before the start of an AE project. In

[1] As the definitions of these terms are based on other terms, we will formally define them in chapter 2.

many cases, these requirements therefore do not fit the given SPL architecture and are very expensive or impossible to satisfy.

In each reuse-based development, all three types of requirements may basically occur during an AE project, as no software system can typically be developed by reusing already existing assets only [HPS08]. Of course, the actual distribution of these requirements varies dramatically depending on the domain that is addressed as well as on the maturity of the reuse asset base.

In Figure 2, we therefore introduce two classes of SPLs that result from different distributions of customer requirements. The distribution on the left side of the figure shows an (traditional) SPL in which most requirements that occur during its AE projects are explicitly anticipated. In this case, the major part of a system can be implemented by configuration rather than actual development only. We therefore call these SPLs configurable SPLs. Most closed or embedded system SPLs belong to this class.

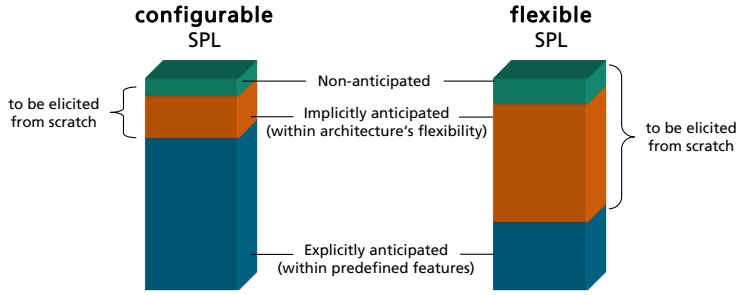


Figure 2. Different distributions of requirements in an SPL

However, besides configurable SPLs, there are also SPLs in which the degree of explicitly anticipated requirements is limited (see distribution on the right side of the figure). In this class, the SPL architecture needs to be flexible in order to also enable the efficient realization of many still unknown requirements. We therefore call such SPLs flexible SPLs.

Definition – Flexible Software Product Line

A flexible software product line is a software product line for which it is either not possible or not economic during DE/FE to explicitly anticipate most requirements that may occur in AE projects.

Even though such SPLs can basically occur in each system category, information system SPLs, in particular, belong to this class, as their development has to cope with specific challenges that are not as prominent in other system classes such as embedded systems. Two examples shall serve to illustrate this fact.

- Information system customers typically have very specific business processes that have to be supported by software in order to enable them to stand out from the competition. Due to the combinatory explosion of potential process variants to be supported, explicitly anticipating them is not feasible. Hence, software vendors can only anticipate them implicitly without defining them in detail.

- Information system customers often have a multitude of legacy and proprietary systems with which a novel system should be integrated. In many cases, however, neither the existence of these systems nor their detailed characteristics can be anticipated during DE/FE. Hence, software vendors can just define basic adapter types, but the implementation details cannot be determined before a detailed requirements analysis has taken place.

Thus, even though a multitude of common and configurable standard functionalities can already be provided by an SPL's reuse asset base, many specific needs have to be addressed by additional development during AE. Even in a mature medium-sized software organization, for instance, we found that only about 60% of customer requirements are explicitly known before an AE project starts (see Appendix I). Hence, a complete scoping or domain-, respectively family, analysis is not feasible here.

The next subsection introduces the practical problem that currently exists during AE in such flexible SPLs in the area of information systems (IS).

1.2 Problem Statement

While much research effort has been spent on how to build up SPLs, the AE phase has not received sufficient attention yet [PKG+08] [RGD07] [RD07]. Thus, even though SPLs have been recognized as a key concept for gaining a competitive advantage in development, building new applications based on an SPL is a still time-consuming and expensive task and often not as easy as proclaimed [DSB05]. This holds especially true for flexible SPLs in the IS area, which are in the focus of this thesis.

In order to enable a better understanding of the origin of this low efficiency, the state of the art in AE is described below. Even if a commonly accepted AE method is still missing [RGD09], many applied approaches share the ideas of the generic process model described by Deelstra et al. [DSB05]. This approach distinguishes an initial configuration phase and a subsequent (tuning) iteration phase.

In the initial configuration phase (see left side of Figure 3), a first version of a system is built based on the already existing reuse assets. For that purpose, the explicitly anticipated requirements of the SPL are instantiated with the customer by using variability models (VM), decision models, or corresponding questionnaires. An example is a requirement concerning the database to be connected. This requirement can typically be predefined in such models, and is therefore easy to elicit and negotiate. Thus, as these models make explicit what is already implemented, they allow customers to state directly which predefined feature they would like to have. A significant fit can therefore be achieved quite fast (see left side of Figure 4).

Definition – Fit

The fit (or, more precisely, the realization fit) between customer requirements and an SPL is the percentage of requirements within a set of elicited requirements that can be economically satisfied with this SPL.

However, the more implicitly anticipated or even non-anticipated requirements must also be addressed in order to satisfy a customer, the more tuning iterations with "from scratch" elicitation and development are also needed [DSB05] [ORR+09] (see middle of Figure 3).

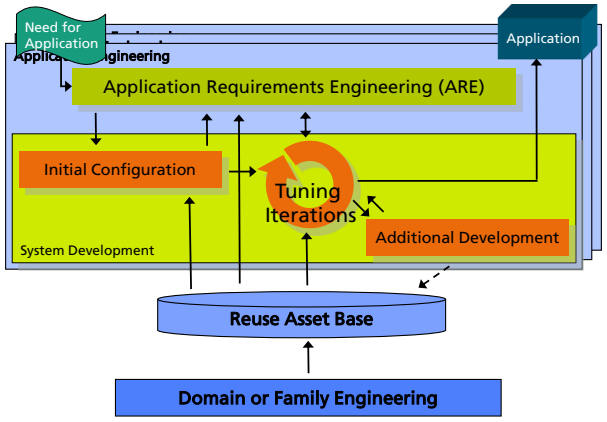


Figure 3. Typical application engineering phases [DSB05]

Unfortunately, existing SPL approaches are rather suited for configurable SPLs, and therefore support only the configuration phase explicitly. Especially when they are to be used for unforeseen, customer-specific requirements, they are very inflexible and insufficient [PKG+08]. Hence, systematic alignment of such requirements with available assets is not supported yet [ORR+09] [DS07] [GP07], leading to the situation that a discussion of customer requirements during the tuning iterations bears significant risks. Especially as selecting an SPL also means accepting a certain set of (architectural) constraints, it becomes apparent that not all customer requirements can be satisfied as initially stated. Rather, trade-

offs between ideal requirements and rapid development must be made in order to retain the profitability of the SPL [ORR+09] and to achieve the best possible fit (which is usually lower than 100% (see Figure 4)).

However, achieving this best possible fit is hard because information about the feasibility and the costs of implicitly anticipated and non-anticipated requirements is neither formalized nor available in current requirements engineering (RE) or SPL approaches (see related work in chapter 3). An example is a business process with specific functional and non-functional properties. In this case, information about its feasibility cannot be represented in decision models or variability models upfront because it is (as mentioned above) not possible to describe each potential process as a variant.

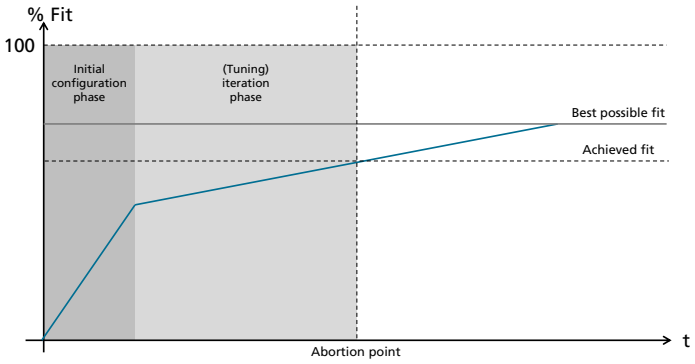


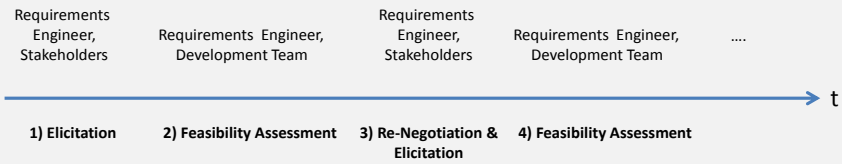
Figure 4. Low elicitation effectiveness in AE

Requirements elicitation and negotiation therefore often become an error-prone and project-delaying task, and still relies on SPL experts (e.g., architects) to predict the impact of requirements that can only be realized with additional development [ORR+09]. In many IS projects, customers are therefore allowed to state any requirements without major restrictions during elicitation. As the resulting requirements may then easily contravene the given architectural constraints or the development needs, it is then often up to the AE team *"to bridge the gap between requirements and implementations"* [BBG+00]. This leads to either a decreased degree of reuse or tedious rework, respectively re-negotiations. The following example should illustrate this:

Motivating Example – Elicitation according to state of the art

A requirements engineer from an organization that develops customer-specific information systems based on a comprehensive business process management suite (BPMS) is eliciting requirements with stakeholders from a customer company. For this purpose, the requirements engineer uses a questionnaire derived from the variability model of the BPMS.
While the requirements concerning other variation points have already been

elicited in previous elicitation sessions, the requirements engineer is now interviewing the stakeholders with regard to the user administration system via which the roles and rights should be imported in the BPMS (1). Based on the supported variants defined in the variability model, he asks the stakeholders whether the BPMS should import user data from Active Directory or another LDAP system. The stakeholders confirm that they would like to import user data from their Active Directory, but that it is also required that the business roles can additionally be important from the ERP system. The requirements engineer looks at the description of the BPMS, as this case is not covered in his questionnaire. The only information he finds is that the SPL provides adapters to SAP via which certain data can be imported, and he asks whether such a system is in place. The stakeholders explain that they do not use SAP but a proprietary system, from which data can only be manually exported as CSV files. The requirements engineer informs the stakeholders that such an export mechanism should be feasible, as, to his knowledge, CSV is a trivial data format with which the BPMS should be able to cope.



However, when the requirements engineer presents the requirements elicited in the last elicitation session to the development team, the SPL expert is not satisfied with the elicitation results at all (2). He informs the requirements engineer that the BPMS does not include a mechanism to parse CSV files, because an explicit design decision had been that XML should be the only exchange format to be used. Furthermore, he tells that it is very costly to define mapping mechanisms to cope with proprietary data structures and that it is complicate to combine user data from different sources. The SPL expert therefore asks the requirements engineer whether it would be possible to reject or at least re-negotiate this requirement, as otherwise 75.000 € extra costs and three months delivery delay would be probably needed. The requirements engineer realizes that he did not have the information about such constraints and that the information provided in the product description or the questionnaire were not sufficient in this regard. Then, he schedules a new elicitation sessions with the stakeholders, in which requirements concerning the user data management are re-negotiated before requirements regarding the next aspect can be elicited (3).

As it can be seen in the example, the progress made towards the best possible fit often decreases significantly during the iteration phase in which customer-specific requirements are addressed (see right side of Figure 4). O'Leary et al. [ORR+09] therefore propose minimizing deviations from the explicitly anticipated requirements by convincing customers that an "*80% solution*" will also do. However, in domains that are served by rather flexible SPLs, the missing 20% typically include those requirements that are indispensable for realizing a customer's competitive advantage. Thus, avoiding them, as proposed by O'Leary et al., is not a suitable option in this context, which also explains why (traditional) configurable SPLs are not applicable here.

When summarizing these findings, it becomes apparent that the non-aligned handling of requirements in AE is a major reason why the actual development efficiency is often lower than expected, even though it has been recognized that the success of reuse mainly depends on how requirements are treated [LLC04]. Especially when a high number of customer requirements are implicitly anticipated or even non-anticipated, the achievement of the best possible fit between customer requirements and SPL characteristics during requirements elicitation in AE is either not possible at all (i.e., the AE respectively the iteration phase ends before the best possible fit is achieved) or takes too much time.

In this regard, the aforementioned project analysis at a successful medium-sized software organization has shown that, on average, 28% of all customer meetings in an AE project are only needed there because requirements have to be renegotiated or clarified again (see Appendix I). Thus, we define the practical problem to be addressed as follows:

Practical Problem

The practical problem to be addressed by this thesis is the low elicitation effectiveness and, as a result, the slow achievement of the best possible fit between customer requirements and SPL characteristics during an AE project in the context of flexible SPLs.

1.3 Thesis Contribution

The overall goal of this thesis is to shorten the time to market in AE projects by achieving the best possible fit between customer requirements and a given SPL faster.

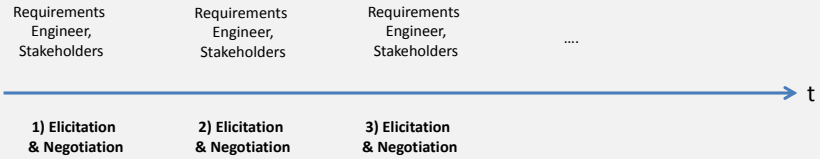
To solve the practical problem mentioned above, the idea is to improve the effectiveness of elicitation through better alignment of customer requirements with SPL characteristics already during the elicitation sessions. This means that the actual needs of the customers are used as a starting point for requirements elicitation (instead of using predefined, explicitly anticipated requirements, variability models, etc.), and that the SPL capabilities and constraints are continuously taken into consideration when these requirements are discussed. Only when these SPL characteristics are sufficiently considered during the AE requirements phase does the chance increase that the requirements will fit the SPL better, and will not need to be reworked in tedious iterations. In particular, requirements that are hard to satisfy can be detected and negotiated much earlier, which reduces the number of costly rework cycles.

The practical contribution of this thesis is therefore the provision of knowledge about a given SPL to AE requirements engineers in order to guide their elicitation work towards requirements that can, to the great-

est extent, be satisfied with a given SPL. Thus, by providing requirements engineers with SPL knowledge, greater progress towards the best possible fit can be made during a specific elicitation session (i.e., higher elicitation effectiveness). A modification of the aforementioned example should illustrate this:

Motivating Example – Elicitation according to the thesis approach

A requirements engineer from an organization that develops customer-specific information systems based on a comprehensive business process management suite (BPMS) is eliciting requirements with stakeholders from a customer company. For this purpose, the requirements engineer uses a questionnaire according to the thesis approach.
While the requirements concerning other topics of interest have already been elicited in previous elicitation sessions, the requirements engineer is now interviewing the stakeholders with regard to the user administration system from which the roles and rights should be imported in the BPMS (1).
Based on his questionnaire, he asks the stakeholders which systems for user management are currently in place and should be used for managing the users of the BPMS. The stakeholders state that they would like to import user data from their Active Directory, but that it is also required that the business roles can additionally be important from the ERP system. Based on the information provided in his questionnaire, the requirements engineer then informs the stakeholders that the connection of the Active Directory is supported by default, but that he needs additional information regarding the ERP system. The stakeholders explain that they do not use SAP but a proprietary system, from which data can only be manually exported as CSV files.



The requirements engineer considers the information provided in the questionnaire regarding adapters to other systems. He finds the information that it is very complicated to connect the customer's ERP system because the BPMS only supports a XML exchange, and because the data mapping with proprietary systems is very costly in general. The stakeholders accept his explanation and state that an interface to Active Directory would be sufficient in a first step. Thus, the requirement to connect the BPMS with the ERP system is rejected directly during the elicitation session. The requirements engineer then proceeds with the elicitation of a next class of requirements (2).

Based on this example, it becomes apparent that the roles that will benefit from the results of this thesis are all the people involved in AE; at least the AE requirements engineers, who are responsible for the elicitation of customer requirements, and the SPL experts, whose explicit involvement is needed today to make feasibility checks. This means that the SPL experts should not need to be contacted each time a customer states a requirement that has not been explicitly defined upfront. Rather, the re-

quirements engineers should be informed well enough to make such an assessment by themselves in most cases.

1.3.1 Research Questions

Making AE requirements engineers aware of SPL characteristics in order to enable more effective elicitation is not an easy task. On the one hand, communicating and representing all capabilities in terms of variability models or decision models (as commonly used) is hard due to the complexity of industrial SPLs [RGD09] [ORR+09]. On the other hand, if the degree of explicitly anticipated requirements is limited (as is the case in flexible SPLs in the IS area), a corresponding formalization is uneconomical or even impossible [DSB05].

The scientific problem to be solved by this thesis therefore deals with the challenge of enabling requirements engineers to use knowledge about a given SPL for guiding the elicitation and negotiation of customer requirements during AE in flexible SPLs more effectively.

Thus, this thesis aims at answering the following research questions:

Research Question 1 – Extraction

How can knowledge about an SPL be systematically extracted and incorporated into application engineering requirements (ARE) processes?

Research Question 2 – Representation

How can knowledge about an SPL be represented appropriately to AE requirements engineers in order to improve their elicitation and negotiation activities?

To answer these research questions, this thesis proposes the following solution described in the next subsection.

1.3.2 Solution Idea

From a scientific point of view, this thesis aims at providing a tool-supported method for the systematic and appropriate incorporation of knowledge about an SPL into a state of the art RE process, respectively supporting instructions.

To achieve this aim, an SPL built during DE/FE must be analyzed systematically. The computer science contribution of this thesis is therefore the algorithmic identification and translation of knowledge about an SPL into suitable application requirements engineering (ARE) instructions. For this purpose, the tailoring method systematically guides method tailors (i.e., persons that are responsible for defining methods and processes in

an SPL organization) in analyzing an SPL's reuse asset base, the selected AE development strategy, and RE best practices for the addressed domain together with SPL experts. Based on this extracted knowledge, precise ARE instructions, which implement an algorithmic elicitation approach, are then automatically generated. In Figure 5, the overall solution idea is depicted graphically.

Regarding the tailoring inputs, the reuse asset base (including the architecture) is needed for supporting SPL experts in identifying the existing SPL capabilities as well as the constraints of the underlying architecture. The development strategy is additionally used for allowing the alignment of the resulting ARE process with the intended development approach, both from a thematic and from a chronological point of view. Finally, RE best practices are taken as input for assuring that the ARE instructions adhere to the state of the art.

The thesis approach aims at identifying and representing SPL characteristics in a descriptive (i.e., constraint-based) way instead of reengineering all requirements that are potentially feasible with an SPL. According to the three alternatives for reusing domain-specific knowledge introduced by Muthig [Mut02], we therefore propose a process-oriented rather than product-oriented approach to express knowledge about an SPL. This means that mainly knowledge about what the ARE process should look like is reused rather than knowledge about the requirements and features that have been explicitly defined during DE/FE. Hence, instead of discussing feature models or decision models with a customer during an elicitation session, we therefore propose a rather "traditional" requirements process, in which explicitly anticipated requirements, implicitly anticipated requirements and non-anticipated requirements are elicited and negotiated in an integrated manner. This notion is based on the work of Guelfi and Perrouin [GP07], who propose that AE should not only rely *"on a fully dictated decision model"*. Of course, the elicitation process still has to be guided systematically based on the capabilities, constraints, and information needs of a given SPL in order to achieve a good fit.

At this point, it is important to highlight that the approach presented in this thesis has an influence on both the DE/FE phase and on the AE phase. With regard to the target audience, the thesis therefore addresses two different stakeholder groups (see Figure 5). During AE, the requirements engineers, who are responsible for the effective reconciliation of requirements with SPL characteristics, are supported by means of ARE instructions as mentioned above. During DE/FE, however, the method tailors, who are responsible for the creation of these instructions, are supported, as they get guidance on how to do this algorithmically. The scientific contribution of this work can therefore be seen in the area of DE/FE, while its practical implications are concerned with AE.

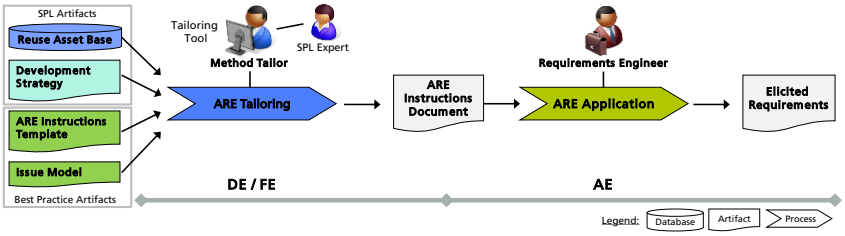


Figure 5. Overall solution idea of thesis

In this regard, the tailoring approach should not be considered as an alternative to any existing SPL approach, but rather as an enhancement to bring more guidance into ARE. In particular, the approach does not replace the scoping or domain analysis. Rather, due to the aforementioned problem of limited anticipation in the area of flexible SPLs, the approach tries to analyze what an SPL is able to support beyond the scope of explicitly anticipated features. Of course, as this SPL analysis requires a deep understanding of what an SPL looks like, it can only be done at the logical end of DE/FE (see Figure 5).

1.3.3 Research Objectives

In order to realize the solution concept, this thesis deals with the following research objectives, which are the subject of the remaining chapters. These objectives are classified into foundations and models, methodological approaches, engineering support, and empirical evaluation.

Foundations and Models

Foundations and models are used to explain the conceptual world on which the solution is built. In this thesis, two research objectives are concerned with the foundations.

- *Conceptual ARE Model.* This model provides the foundation for the entire thesis work and explains how ARE processes are conceptually related to an underlying SPL. By knowing these dependencies, we can easily define which requirements have to be delivered by ARE in order to instantiate an SPL effectively and efficiently.

- *Issue Model.* This model describes RE best practices in terms of the issues to be discussed during elicitation in a certain domain. In the context of this thesis, the issue model has been specialized for the IS area.

Methodological Approaches

Methodological approaches describe the utilization of the foundations in order to answer the research questions. In this thesis, two objectives are concerned with methodological approaches.

- *ARE Instructions Template.* This template provides a generic structure as well as a set of predefined text blocks for representing best practices and important knowledge about an SPL to AE requirements engineers by means of algorithmic elicitation instructions.

- *Tailoring Method.* This method describes a clear sequence of algorithmic activities to be carried out by method tailors during DE/FE in order to derive ARE instructions from a given SPL. For this purpose, the tailoring method makes use of the conceptual ARE model, the best practice model, as well as the ARE instruction template.

Engineering Support

Engineering support aims at realizing the methodological approaches in a way that enables their efficient use in practical settings. In this thesis, one research objective is concerned with engineering support.

- *Tailoring Tool.* This tool (semi-)automates the execution of the tailoring method by generating proposals for the results of different tailoring steps based on the results of a previous step. The main contribution of the tool is the automatic creation of an ARE instructions document based on the intermediate tailoring results.

Empirical Evaluation

Empirical evaluations are used to show the usefulness of the methodological approaches with regard to the research questions or the practical problem. In this thesis, two research objectives are concerned with empirical evaluations.

- *Controlled Experiment.* This study evaluates whether requirements engineers using ARE instructions according to the methodological approaches of this thesis are able to elicit requirements more effectively than when using state of the art material.

- *Case Study.* This study evaluates the feasibility of the tailoring method in order to investigate whether ARE instructions can be effectively defined based on a given SPL.

1.3.4 Scientific and Practical Benefits

According to the aforementioned problems, research questions, and solution idea, different benefits are expected for research and practice when exploiting the results of this thesis (see Figure 7).

With regard to economic implications, which are probably the most important ones for justifying a dissertation in applied research, the thesis aims to provide the following improvement:

Hypothesis 1 – Efficiency of Application Engineering

H_1. *An AE process using ARE instructions defined on the basis of the thesis approach has an at least 15% shorter time to market than an AE process using state of the art instructions.*

From an engineering point of view, this improvement should be made possible by higher effectiveness in requirements elicitation. In this regard, the thesis aims at achieving the following practical benefit:

Hypothesis 2 – Effectiveness of Elicitation

H_2. *ARE instructions defined based on the thesis approach enable requirements engineers to achieve an at least 15% higher realization fit during an elicitation session than when using other instructions.*

Thus, when using the thesis approach, more requirements elicited during an AE project will fit the SPL characteristics at a certain point in time (e.g., the abortion point) than when using a state of the art approach[2] (see Figure 6). As less rework (for either re-negotiations or costly implementations) is then needed, the overall efficiency of the AE phase in terms of duration is expected to increase (see hypothesis 1).

Furthermore, knowing SPL characteristics and corresponding information needs can also help to avoid both the unnecessary elicitation of requirements that are satisfied by default anyway (i.e., common features) and the omission of requirements that must be known for instantiating the SPL. Again, time and effort is expected to be saved. In Appendix J, the calculation basis for the quantified hypotheses is shown.

[2] In our study, we have combined a decision model approach according to PuLSE-I [BGM+00] with a top-down elicitation approach according to TORE [PK04].

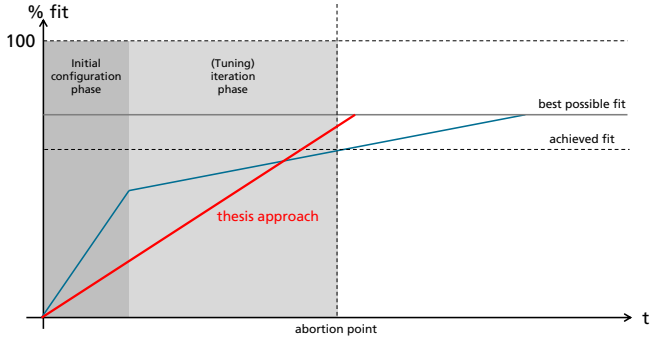


Figure 6. Intended improvements in application engineering

However, in order to actually achieve this elicitation improvement, the required knowledge must be made available to AE requirements engineers. Thus, the incorporation of SPL knowledge into ARE instructions must work effectively. The thesis therefore aims at providing the following benefit:

Hypotheses 3 – Effectiveness of Tailoring

H_3. *An incorporation of SPL knowledge into ARE instructions is possible when using the thesis approach, i.e., at least 80% of method tailors are able to successfully create ARE instructions without major problems.*

Finally, on the scientific level, the thesis clarifies the relationship between SPLs and ARE processes, respectively corresponding instructions. To our knowledge (see related work in chapter 3), such clarifications and explanations have never been made before, which is why this thesis delivers a novelty in this regard. However, a corresponding hypothesis is not formulated, as this is demonstrated in a rather analytical way.

1.3.5 Assumptions and Limitations

The benefits mentioned above are not expected to be realized in every AE context. This holds especially true for rather configurable SPLs that do not need to cope with a multitude of implicitly anticipated or even non-anticipated requirements. This thesis is therefore based on the following assumptions, and intentionally accepts certain limitations:

1. *High flexibility required.* The SPLs to be addressed by this thesis are characterized by a high degree of flexibility, i.e., the degree of explicitly anticipated requirements is limited due to technical or strategic reasons. Nevertheless, the thesis does not aim at supporting the case that a customer states requirements such as changes to the SPL architecture at all costs because this does not play any role in the highly competitive market that is to be served better through SPLs.

Instead, the thesis is based on the assumption that one should insist on a realizable scope, even if not all requirements need to be explicitly known upfront.

2. *Planned flexibility and producibility.* An assumption related to the previous one is that the SPL architecture has been designed for flexibility [Naa09] and producibility according to a certain development strategy [Car08]. In this regard, a further assumption is that the architecture is sufficiently specified in order to extract the required knowledge. However, this thesis deals neither with the definition of such a strategy nor with the design of a flexible architecture. Instead, it just helps to better exploit the corresponding potentials.

3. *Human-based SPL analysis.* An important assumption of this thesis is that a fully automatic analysis of reuse asset bases or SPL architectures is not appropriate in the addressed context. Instead, the thesis requires that SPL experts (e.g., architects) are available who can exhaustively explain a given SPL and the constraints of the architecture (see Figure 5). Otherwise, the thesis approach is not able to identify, analyze and systematically incorporate important knowledge into the requirements process. The thesis contribution is therefore not an automatic SPL analysis, but a method for SPL experts to can externalize their SPL knowledge.

4. *Human-based requirements elicitation.* Similar to the previous assumption, it is also assumed that the actual elicitation and negotiation of requirements remains a human-based task. In this context, however, the approach is not intended to support the "world's best requirements engineers", but rather the majority of people who are challenged with requirements elicitation in practice.

5. *Stable requirements.* The ARE instructions developed by the thesis approach intentionally neglect the whole area of requirements management. Especially the handling of change requests is not covered, even if the inherence of requirements changes is not put into question. Consequently, the avoidance of "normal" rework is not the focus of this thesis.

6. *Problem-driven elicitation.* We assume that requirements in the context of flexible SPLs have to be elicited in a problem-driven (top-down) instead of reuse-driven (bottom-up) manner in order to satisfy the actual customer goals. In this regard, we also assume that such a problem-driven elicitation implicitly assures a sufficient degree of requirements completeness from a customer's point of view.

1.4 Research Approach

In this thesis, we applied a systematic research approach following the design science research process [PTG+06]. As this process aims at observing the world and existing solutions, building new models, and validating them with regard to explicitly stated hypotheses, it can be considered as a scientific method according to the classification of Basili [Bas93]. Below, the concrete procedure of how our research has been carried out is described.

1. *Identification and motivation.* Before starting the thesis research, our experience in many projects with Fraunhofer IESE's industry customers was that it is an almost more difficult to specify requirements that satisfy developers than to specify requirements that satisfy customers [ADE09]. In many requirements specifications, we recognized that design-relevant information is often missing, while superfluous information is described extensively. The biggest issue, however, was the observation that already existing components or concepts were often not considered sufficiently when requirements were elicited. The resulting fit problem between requirements and reusable assets, especially in the IS context, was therefore selected as the practical problem to be addressed. The feedback we got from the research community, from software organizations, as well as from the literature (see chapter 3) has confirmed that this practical problem is actually important and solving it is worthwhile.

2. *Objectives of a solution.* In this step, the main objectives of a solution were derived from the problem definition. An important decision during this activity was the clear determination of what our approach should accomplish [PTG+06]. Based on the practical problem identified before, we therefore started investigating the state of the art in reuse-oriented RE. We quickly found that even recent approaches from the SPL area could not be used to solve the problem because all assumed an explicit anticipation of requirements. We therefore started sketching a novel solution. Our still rather high inexperience in the related research areas supported this aim, as *„ignorance of a topic makes it easier to think out of the box, and to come up with a creative never-though-of solution"* [Ber10]. Thus, we departed from the traditional idea of building explicit requirements-, domain-, variability-, or decisions models, and rather proposed the usage of tailored SPL-aware requirements processes to enable requirements engineers perform elicitation with better information on their hands. This idea was fostered by the positive experience Doerr et al. [DPK04] had made with requirements process tailoring based on information needs. Concrete research questions and hypotheses resulting from this notion were then derived.

3. *Design and development.* The actual design and development of the artifacts that were needed to realize the solution idea started immediately after the research questions had been defined. Among other things, we elaborated the still abstract idea by defining the inputs, outputs and basic activities of the tailoring approach. The presentation of our initial results to the RE community at a Doctoral Symposium [Ada10] and an SPL workshop [ADE+10] was met with motivating acceptance and confirmed that we were on the right track. During the last one and a half years of the thesis work, we then improved all solution artifacts based on the feedback we had received until then and the experience we had made during intermediate studies. Based on this improved formalization, we developed a tool that automated many parts of the tailoring approach.

4. *Demonstration and evaluation.* Regarding the demonstration and evaluation of our work, we performed early and late studies. During an early study, we were just interested to see whether a tailoring of ARE processes was actually feasible as intended. Thus, we performed a case study by using a draft version of the tailoring method in a medium-sized software organization [ADE+10]. We found that our tailoring idea was basically feasible and that the outcome in terms of precise ARE instructions looked promising. However, the experience made in this study challenged us to make significant improvements to the solution artifacts because the degree of formalization as well as the consistency were not sufficient yet. At the end of our thesis research, we then performed late studies. First, we let practitioners from a medium-sized software organization as well as from Fraunhofer IESE use the final tailoring method for developing ARE instructions in order to validate the "effectiveness of tailoring" hypothesis. Second, we performed a controlled experiment with 26 students in order to validate the "effectiveness of elicitation" hypothesis. The results of the late studies confirmed our claims and delivered additional feedback that helped us in making final improvements and identifying open issues for future research.

5. *Communication.* The aforementioned research activities and their results were communicated continuously to the research community mainly in terms of scientific publications (e.g., [AD08], [ADE09], [Ada10], [ADE+10], [Ada11], [Ada11b], [Ada11c], [RAG11], [Ada12], [ARG+12]). However, besides the pure dissemination of the results, communication with the research community was also a fruitful means for the informal validation of our work, as important feedback was received both during the review of our papers and during the presentation at conferences or workshops.

1.5 Outline

In Chapter 2, basic terms and concepts are introduced and formalized. The purpose of this chapter is twofold. First, the set of requirements to be of actual interest in an AE project is defined. Second, the conceptual relationships between an SPL and ARE processes are elaborated and formalized. This chapter therefore provides the foundation for the definition of the methodological approaches of this thesis.

Chapter 3 provides an overview of related work and investigates its strengths and weaknesses. For this purpose, related work both with regard to the practical problem and with regard to the research contributions has been considered. The purpose of this chapter is to give some insights about existing approaches, accepted notions, as well as also problematic and open issues.

In Chapter 4, a template for ARE instructions is introduced and formalized. This formalization is based on best practices from the literature, own previous work, as well as RE experts' input. The chapter precisely defines what the outcome of the thesis tailoring approach should look like, and which form of knowledge representations is appropriate for guiding the elicitation in an (almost) algorithmic manner.

In Chapter 5, an issue model for RE in the IS area is introduced. This model reflects the topics to be typically discussed in IS projects and is therefore used for taking this "best practice" into account during tailoring.

In Chapter 6, the core contribution of the thesis, i.e., the tailoring approach, is introduced and described in a formalized way. The purpose of this tailoring approach is to provide a clear process that allows algorithmic reflection of SPL knowledge in ARE instructions according to the template introduced in chapter 4. In this context, a supporting tool is also presented.

The evaluation of the entire solution is part of Chapter 7. In particular, the design and setup of our late studies, as well as their results and implications are presented.

The thesis closes with a brief summary and an outlook on future work in Chapter 8.

1.6 Summary

Besides software product lines (SPL) in which a high number of requirements can be fulfilled by just configuring the reuse asset base, there are SPLs in which the degree of explicitly anticipated requirements is limited. This is especially the case for information systems (IS), which must fulfill a multitude of very specific requirements in order to enable a customer to stand out from the competition.

In these flexible SPLs, however, the typical benefits concerned with reuse are often fewer than expected. One important reason is the fact that the achievement of the best possible fit between customer requirements and SPL characteristics during application engineering (AE) either does not work at all or takes too much time. This is caused by the insufficient knowledge requirements engineers typically have regarding the given SPL.

Thus, in order to increase the fit between requirements and reuse capabilities, information about an SPL must be considered better by AE requirements engineers during elicitation. The central contribution of this thesis is a tailoring approach that systematically incorporates knowledge about an SPL into a state of the art requirements process. Hence, the requirements process for the AE phase and its guiding artifacts are tailored based on the capabilities, constraints, and information needs caused by a given SPL. The corresponding application requirements engineering (ARE) instructions then assure that the actual elicitation in AE can reconcile customer requirements with SPL characteristics more effectively, and especially in a constructive rather than an analytical way. This finally results in higher efficiency of AE projects (see Figure 7).

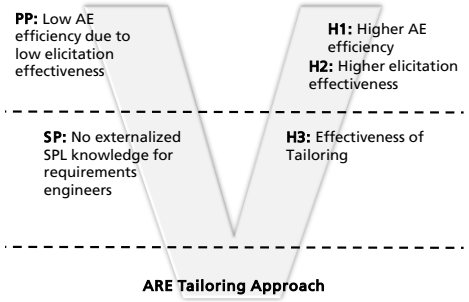


Figure 7. Research V-Model for this this

This thesis describes the scientific components of this concept as well as their evaluation. In particular, a conceptual ARE model, an ARE instruction template, an ARE tailoring method, a supporting tool, and an issue model will be introduced in the subsequent chapters.

2 Foundation

"Those who want to build high towers,
have to linger long at the base."
Anton Bruckner

As shown in the solution idea (see section 1.3.2), this thesis provides an approach that aims at improving the effectiveness of requirements elicitation through tailored ARE processes. To make this happen, the thesis tailoring approach takes an SPL's reuse asset base, a development strategy according to which systems should be derived, and RE best practices as input. Based on this product- and process-oriented knowledge, AER instructions are then generated, which are finally used for eliciting customer requirements in a concrete AE project. This logical input-output sequence is again depicted in Figure 8.

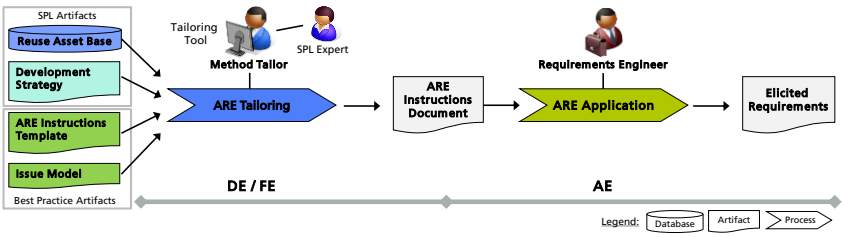


Figure 8. Inputs and outputs of the thesis approach

This chapter clarifies the elements within the input and output artifacts of the thesis approach as well as their conceptual relationships. The purpose of this chapter is to explain which requirements should be the focus of an ARE process and how this process depends on the characteristics of a given SPL. The resulting conceptual ARE model then acts as a foundation for the methodological approaches of this thesis introduced in later chapters.

To make this happen, the following questions are answered below:

- How are ARE processes integrated into the AE phase?
- Which product knowledge about the SPL is important for deriving ARE processes?

- Which process knowledge about the development strategy is important for deriving ARE processes?
- Which RE best practice knowledge is important when deriving ARE processes?
- Which types of requirements must be elicited in an ARE process in order to fit a given SPL?
- Which elements must be part of an ARE process and corresponding instructions?
- How are ARE processes and corresponding instructions conceptually related to the aforementioned artifacts?

The subsections below answer each question one by one. However, it has to be noted that we will only introduce elements that are actually required for the thesis approach. This means that, for instance, a full-fledged description of an SPL and its detailed concepts will not be given, as far as these concepts are not needed for defining ARE processes also.

2.1 Research Approach

The foundation has been developed in several iterations throughout the entire phase of the thesis research (see Figure 9).

In a first step, basic literature mainly from the RE and SPL communities was analyzed in order to identify the central concepts in these two areas. However, as many notions in literature were just "common sense" and not further formalized or explicitly described there, considering tacit expert knowledge was also an important input for the development of the thesis foundation. This tacit knowledge was mainly elicited via intensive discussions with requirements-, architecture-, and SPL experts at Fraunhofer IESE.

Based on this input as well as on experience from previous research, we consolidated the gathered knowledge (step 1). As a result of this step, a first version of the conceptual ARE model was developed.

During a couple of subsequent iterations, this model was then checked for completeness and consistency (step 2). This was done in two different ways. On the one hand, the results were discussed with the aforementioned experts at Fraunhofer IESE as well as with an external supervisor. Their feedback was carefully analyzed and incorporated into a novel version of the model. On the other hand, completeness and consistency were checked by using the foundation during the development of the

other thesis components. In particular, the development of the actual tailoring method, the ARE instructions template, and the corresponding tailoring tool was an excellent means to challenge the existing foundation. Hence, besides expert feedback, insights gathered during these research steps were incorporated into an adapted version (step 3).

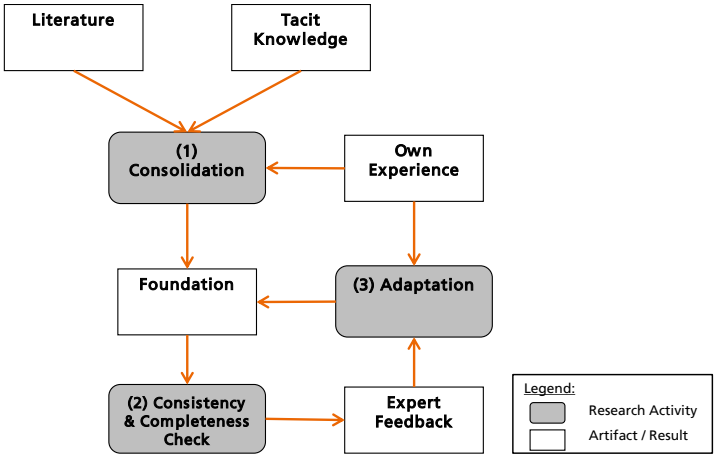


Figure 9. Research approach for the foundation

During the last months of thesis research, no incompleteness or inconsistency was detected anymore and the work converged towards a stable model. In particular, the model enabled us to align all thesis components seamlessly without any workarounds or deviations.

2.2 Application Engineering

To illustrate the role of ARE processes within AE, a high-level overview is given in Figure 10 using the UML notation [OMG11]. As introduced by Deelstra et al. [DSB05], and as depicted in Figure 3, RE and development (often denoted as product derivation) are explicitly distinguished here.

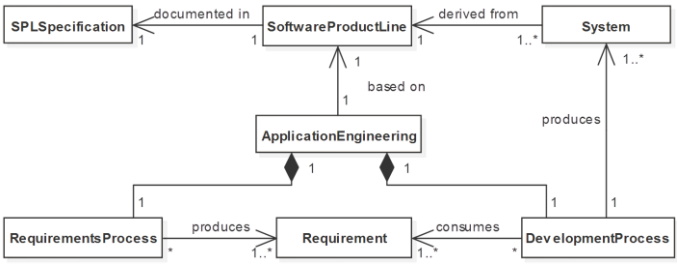


Figure 10. High-level overview of application engineering

The reason for this separation is that development processes in AE are often less generic than in single system development. Rather, they are focused on concrete activities that are needed to derive a customer-specific system from a given SPL via the instantiation of product line architectural elements. In this context, we define development processes and requirements processes according to other authors as follows:

Definition – Development Process

A development process is a structured set of activities, work products, roles, and tools aimed at the development of a customer-specific information system based on an SPL's reuse asset base and a set of requirements. [Mut02][Car11]

Definition – Requirements Process

A requirements process is a structured set of activities, work products, roles, and tools for creating, validating and maintaining requirements that are needed as input for a development process. [SS97][Dav93]

According to these definitions, the development process implies that the customer requirements for a new system are sufficiently known. Thus, the requirements process is responsible for elaborating requirements that specify what each individual system to be derived from the SPL should look like. We define a requirement and, for the sake of completeness, the other elements of Figure 10 as follows:

Definition – Requirement

A requirement is an information about a characteristic or capability a system must have, or about a characteristic of the usage environment a system must consider in order to satisfy a stakeholder goal.

Definition – System

"A system is a set of components interacting with each other to satisfy some global objectives." [Lam09]

Definition – SPL Specification

An SPL specification is the official statement of what to implement in an SPL and therefore contains a complete description of the anticipated variable and common SPL capabilities from a customer's point of view.

However, even though the requirements process and the development process could be completely decoupled when the focus is only on the exchange of requirements, the intent to highly benefit from an SPL approach makes it indispensable to align the two processes more systematically (see chapter 1). This means that the requirements process must be strongly oriented on the development process, as otherwise the requirements process cannot assure that only requirements that are of ac-

tual value for development are elicited. In order to find out how such an ideal alignment should look, a deeper look into the reuse asset base and the development strategy must be taken.

2.3 Reuse Asset Base

A reuse asset base is a logical repository containing all artifacts of an SPL that are potential subject for reuse. Besides compiled components and source code modules, a reuse asset base may also include artifacts such as requirements specifications, architectural designs, process descriptions, development guidelines, etc. In the context of this thesis, however, the reuse asset base is only investigated for the purpose of extracting product-oriented knowledge, i.e., for extracting the current capabilities and constraints of the SPL (see section 1.3.2). Hence, only product-oriented artifacts are of interest here.

2.3.1 Product Line Architecture

The core artifact to be analyzed in order to understand what an SPL is able to do is the product line architecture. Basically, a software architecture is the *"structure of a system, which comprises software elements, the externally visible properties of these elements (capabilities), and the relationships among them"* [BCK03]. A product line architecture is then defined as follows.

Definition – Product Line Architecture (PLA)

"A product line architecture is the generic software architecture for all systems in an SPL providing variation mechanisms that support the diversity among these systems." [NC07]

A central goal of a product line architecture is the description of the common and variable elements in the SPL, and their interconnections [Gom04]. In contrast to single system architectures, a product line architecture therefore addresses explicit variability aspects, and comprises corresponding possibilities in this regard. Furthermore, each product line architecture constrains both the solution space and the problem space that can be addressed by systems derived from the SPL.

In Figure 11, we have elaborated the core elements and related concepts of a product line architecture based on existing literature and expert discussions. According to this figure, each SPL has a product line architecture, which is a specific kind of software architecture as mentioned above. As each software architecture comprises a set of architectural elements [RW05], a product line architecture is also composed of corresponding parts, which we call product line architectural elements below.

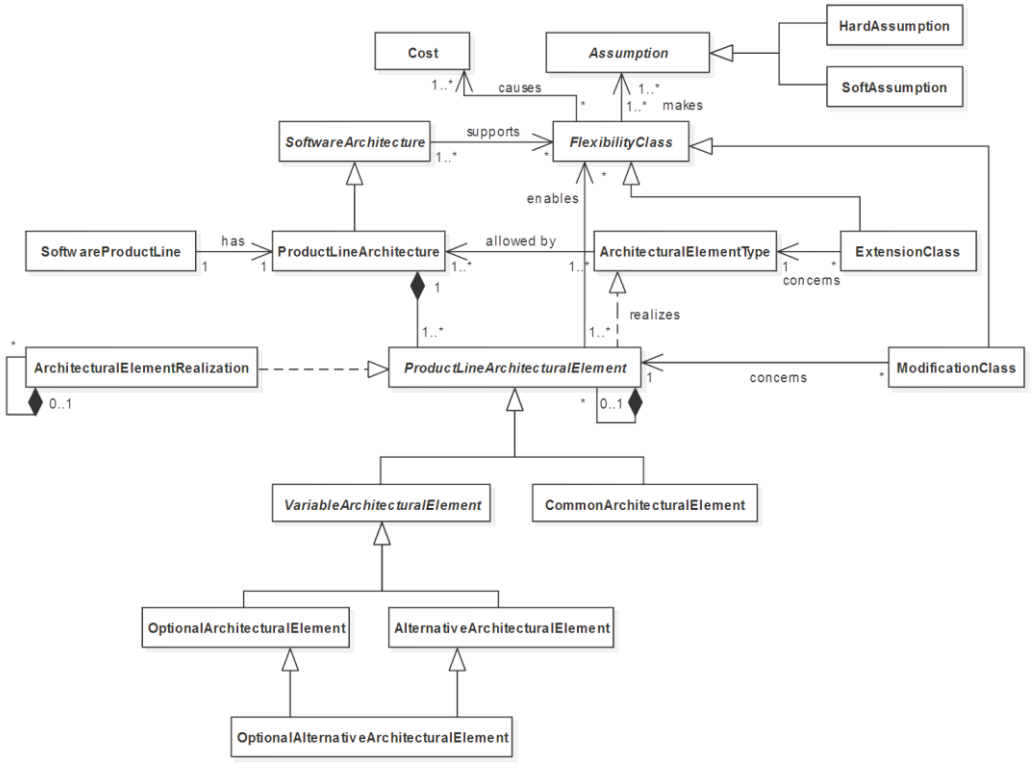


Figure 11. Core elements of a product line architecture

According to Rozanski and Woods [RW05], we define them as follows:

Definition – Product Line Architectural Element

A product line architectural element is a fundamental piece of software from which the systems derived from an SPL are (recursively) constructed.

In the context of this thesis, we do not analyze architectural elements further (see Carbon [Car11] for such an analysis). Rather, we consider them as a piece of software that provides either business-oriented functionality (e.g., specific components for processing incoming documents), or infrastructure-oriented capabilities for crosscutting, technical, or general concerns (e.g., database, workflow engine, adapters).

However, as architectural elements may address very different things, architectural element types are needed to classify and to define the valid entities of which a system should consist. This is especially important when using an SPL, as otherwise there is no assurance that a consistent set of elements is used in the different systems derived from it.

Definition – Architectural Element Type

"An architectural element type is a class of architectural elements recurring in software architectures that follow a certain architectural style." [Car11]

In the S3 reference architecture for service-oriented IS [Ars+07], for instances, service, service components, adapters, business processes, portlets, and the like are proposed as the main architectural element types to be used. Hence, the architectural elements in a concrete system derived from an SPL that is built upon this architectural style realize these specific architectural element types.

However, on the level of both architectural element types and product line architectural elements, many important details that constitute the actual system behavior are still open. Thus, each architectural element needs an internal realization to become real. We define these realizations as follows:

Definition – Architectural Element Realization

An architectural element realization is a concrete implementation of an architectural element with specific functional and non-functional characteristics.

For instance, a product line architectural element "database system" could be realized by either a MySQL or an Oracle database with significant differences in the functional characteristic "query power". In traditional SPL terminology, architectural element realizations therefore reflect the variants of a certain architectural element.

Independent of the architectural element type to which they belong and their realizations, product line architectural elements can be classified further. As the separation of common and variable characteristics is a basic concept in product line engineering [MA02], product line architectural elements can also be classified in this way. We therefore make the following distinction in this thesis:

Definition – Variable Architectural Element

A variable architectural element is a product line architectural element that is either optional in the systems derived from the corresponding SPL or that may have different architectural element realizations in these systems.

Definition – Common Architectural Element

A common architectural element is a product line architectural element that is mandatory in all systems derived from the corresponding SPL and that has always the same architectural element realization.

In the aforementioned example, the database system is a variable element because the concrete realizations may vary between certain sys-

tems. However, if an architectural element is a variable architectural element or a common architectural element is typically a strategic decision made during DE/FE and not further formalized in this thesis. Especially scoping (see [Sch03]), which aims at *"deciding in which parts of a product systematic reuse is economically useful"* [JKL+06], provides a first decision in this regard, even if concrete architectural elements are not discussed yet during the scoping phase.

Variabilities in an SPL are typically further distinguished into optionalities and alternatives, where alternatives also cover multiple choices [MA02]. To address these notions, we categorize variable architectural elements into optional architectural elements, alternative architectural elements, and optional alternative architectural elements.

Definition – Optional Architectural Element

An optional architectural element is a variable architectural element that does not need to be included in each system derived from the corresponding SPL. By default, an optional architectural element is always realized with the same architectural element realization.

Definition – Alternative Architectural Element

An alternative architectural element is a variable architectural element whose architectural element realizations may vary between the systems derived from the corresponding SPL.

Definition – Optional Alternative Architectural Element

An optional alternative architectural element is a variable architectural element that is both optional and alternative, i.e., it does no need to be included in each system, but if it is included, its realization may vary between the systems.

Following these definitions and considering the fact that each product line architectural element is unique within a system, it becomes apparent that the multiple-choice concept cannot be realized on the level of a single product line architectural element. Rather, additional architectural elements of the same architectural element type are needed in this case, where each element is realized differently. If, for instance, a system should use different database systems (multiple choice), there would be more than one database element in the derived system architecture.

2.3.2 Flexibility Classes and Assumptions

As specific architectural elements may occur in an AE project, a product line architecture must be able to cope with architectural elements and corresponding realizations not anticipated explicitly during DE/FE. The decision about whether this is (economically) feasible or not, mainly de-

pends on the flexibility classes the architecture supports. In the context of this thesis, we define a flexibility class as follows:

Definition – Flexibility Class

"A flexibility class is an aggregated set of coherent flexibility requirements a software architecture should be able to deal with." [NM10]

A flexibility class therefore describes what can be added or modified in a concrete AE project at which costs. In particular, based on our explicit distinction between architectural element types and product line architectural elements in this thesis, flexibility classes can cover changes on the entire architecture as well as changes on the detailed implementation. Thus, we introduce the flexibility of extending new elements from the flexibility of modifying existing elements here:

Definition – Extension Class

An extension class is a flexibility class that enables the addition of customer-specific architectural elements of a certain architectural element type including corresponding architectural element realizations.

Definition – Modification Class

A modification class is a flexibility class that enables the creation or modification of new architectural element realizations for a product line architectural element.

An example of a modification class could be that a foreseen database system interface has to be re-implemented in an alternative way in order fit a proprietary application. An example of an extension class would be the development of additional database system interfaces, as the product line architecture only comprises one database interface by default.

Thus, when a product line architecture supports a set of flexibility classes, architectural elements are expected to be (easily) modifiable or extensible during AE, even if these elements or realizations were not anticipated explicitly during DE/FE.

However, implementing any desired requirement is not always possible, as some realizations may require the architecture to be changed significantly. Only architectural element realizations that do not require such changes are considered to be (economically) feasible when using an SPL. If, for instance, a selected element realization cannot guarantee a certain response time, the entire communication mechanism of the architecture might not work. As already mentioned, a product line architecture therefore constrains both the solution space and the problem space that can be addressed with an SPL.

In order to keep the flexibility within a (economically) feasible scope, flexibility classes therefore make certain assumptions about the requirements that should be satisfied. We define an assumption as follows:

Definition – Assumption

"An assumption is a proposition about customer requirements that may have a detrimental effect on the development effort when not coming true." [RR99]

Assumptions state the limitations of the supported flexibility. In the example of a certain work place to be supported, an assumption could be that all work places have broadband Internet access. Thus, assumptions allow determining the set of implicitly anticipated requirements in a descriptive manner, and also help to decide under which conditions a non-anticipated requirement is (economically) feasible or not. In this regard, we distinguish hard and soft assumptions here.

Definition – Hard Assumption

A hard assumption is an assumption from which it is known upfront that it will cause (economic) non-feasibility in every case of not coming true.

Definition – Soft Assumption

A soft assumption is an assumption for which it is not known upfront whether it will always cause (economic) non-feasibility when not coming true.

2.4 Development Strategy

While the reuse asset base and the product line architecture are considered as the primary sources for extracting product-oriented SPL knowledge (i.e., capabilities and constraints), the development strategy is seen as the main driver for process-oriented SPL knowledge.

In general, a development strategy describes how a software organization intends to develop systems based on a given SPL during AE projects. Hence, we define this strategy as follows:

Definition – Development Strategy

A development strategy is a generic plan of action including basic principles of how an SPL organization would like to build systems during AE.

A development strategy could, for instance, describe that systems should be developed layer by layer, or that systems should be developed incrementally, where a specific business process to be reflected in the system could define a specific development increment [Car11].

2.4.1 Development Process

In most cases, the development strategy is implicitly manifested in the development process, and documented as a production plan that describes how applications are to be developed by reusing assets from the SPL [CN01]. Thus, the inputs, activities, roles, and outputs when deriving customer-specific systems are clearly defined.

However, as the product line architecture constrains the development of a system, an efficient development process should always be derived from the given architecture instead of being defined in a (traditional) phase-oriented way [Car11]. This means that development processes should not be defined using generic practices according to different software engineering disciplines (e.g., requirements analysis, designing, coding, etc.), but should be based on concrete tasks to be applied in order to assemble or implement the product line architectural elements in a project. Thus, the product line architecture is an important driver for the determination of the development process and its detailed activities.

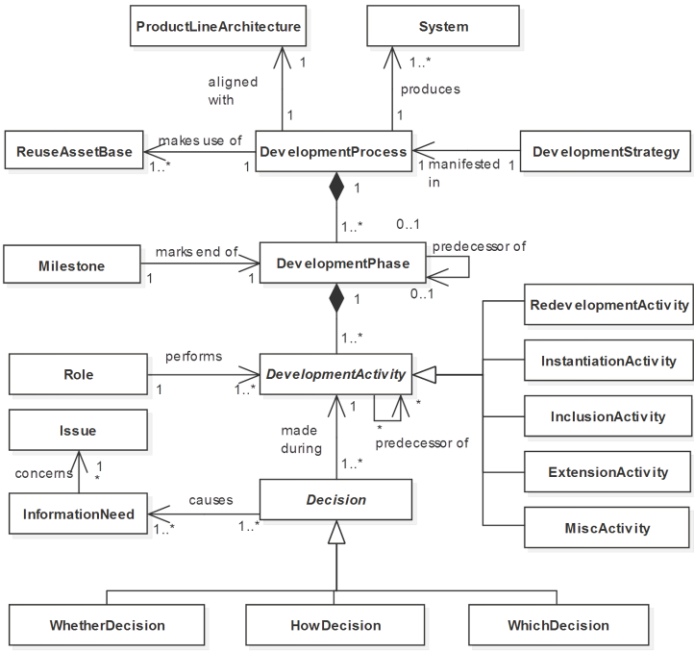


Figure 12. Core elements of a development process

Based on the literature and expert discussions, we therefore elaborated the core elements and concepts that are either part of a development process or related to it (see Figure 12). As already mentioned above, the purpose of the development process is the production of customer-specific systems by making use of an SPL's reuse asset base. For this purpose, the development process should be closely aligned with the

product line architecture in order to streamline the development activities in this regard.

Just like any other process, a development process also comprises an ordered set of activities with responsible roles that are organized in sequential phases reaching different milestones.

Definition – Development Phase

"A development phase is a fixed period of time wherein certain development activities are performed." [Car11]

Definition – Development Activity

A development activity is a procedure that creates an intermediate result relevant for the overall development of a system.

Definition – Role

"A role is a class of persons based on a logical set of their responsibilities, rights, and tasks." [Poh07]

Definition – Milestone

"A milestone is a scheduled event to measure progress." [IEEE98d]

As an architecture should define the work to be done in a development process [Car11], we classify development activities based on the product line architectural element or the architectural element types with which an activity is concerned. Thus, we distinguish the development activities that may occur in an AE development process into inclusion, instantiation, redevelopment, extension, and miscellaneous activities as follows.

Definition – Inclusion Activity

An inclusion activity is a development activity in which an optional architectural element is reused in the architecture of a derived system.

Definition – Instantiation Activity

An instantiation activity is a development activity in which an existing architectural element realization of an included variable architectural element is reused for implementing this element in a derived system.

Definition – Redevelopment Activity

A redevelopment activity is a development activity in which a new or modified architectural element realization is created for a product line architectural element included in a derived system.

Definition – Extension Activity

An extension activity is a development activity in which a specific architectural element of a certain architectural element type is created from scratch including the development of a corresponding architectural element realization in a derived system.

Definition – Miscellaneous Activity

A miscellaneous activity is a development activity that deals with a task to be done during a development process, except for inclusion, instantiation, redevelopment, and extension.

The concrete alignment of development activities with architectural elements is discussed in a later section of this thesis (see section 6.3.7).

2.4.2 Decisions and Information Needs

As in each software development project, AE projects also require creativity and human-based decision making in order to perform the aforementioned development activities properly. Decisions therefore play an important role in software development and need to be supported in the best possible way.

Definition – Decision

"A decision is a choice made between alternatives in a situation of uncertainty." [BC12]

Hence, decisions typically answer *"what should I do"* questions and determine a (future) behavior. In the context of AE, decisions are typically concerned with the overall question of what should be done in order to derive a customer-specific system from a given SPL. Based on the aforementioned development activities, we introduce the following decisions to be made in a development process:

Definition – Whether Decision

A whether-decision is a decision that determines either whether or not a certain optional architectural element is needed in a derived system, or whether or not a common architectural element has to be implemented in a customer-specific way.

Definition – Which Decision

A which-decision is a decision that either determines an existing architectural element realization that should be reused in a derived system for implementing an alternative architectural element, or a decision that determines architectural elements of a certain architectural element type that are additionally needed.

Definition – How Decision

A how-decision is a decision that determines how a specific architectural element realization for a certain architectural element should look.

The concrete alignment of decisions with affected architectural elements is discussed in a later section of this thesis (see section 6.3.8). However, all these decisions have in common that they depend on the required characteristics a certain architectural element (realization) must provide in order to fulfill the customer requirements. If, for instance, a certain complexity of queries is required, only database realizations whose property "query power" can assure the required value may be chosen.

Thus, for being able to make a decision, information must be available in order to know what a customer actually wants or needs. To decide whether an interface is needed, for instance, one has to know with which external applications the system is supposed to interact. Furthermore, to decide how to implement this interface, additional information concerning the data structures, etc. is needed. Hence, each decision causes information needs that must be satisfied before the decision can be made. We therefore define an information need as follows:

Definition – Information Need

An information need is the necessity to have information about a certain issue in order to be able to make a decision.

Information needs typically occur when there is a gap between the available information and the information that is indispensable to provide a correct solution. The simplest information need (which is often the one used in traditional ARE) is the need to know which possible architectural element realization (i.e., variant) a stakeholder wants. However, as customers typically know what they want, but not what they really need [Dav93], decision-making is usually more complex, and requires additional information about the system context and the intended use. For example, deciding which database system should be used depends on information about required response time, required query functions, data to be stored, as well as organizational information about budget, existing licenses, etc. Thus, each information need is concerned with a certain element of the real world, which we denote as an issue:

Definition – Issue

An issue is an inherent element that is either part of a system or part of the system's usage environment.

2.5 RE Best Practices

While the reuse asset base and the product line architecture are sources for product-oriented SPL knowledge, the development strategy has been introduced as a source for extracting process-oriented SPL knowledge. However, even though these sources are indispensable for deriving an ARE process that is able to fit the SPL, it is important not to neglect best practices that have been established in RE independent of the underlying development approach.

In general, best practices can cover both actual practices in terms of methods, techniques, or activities, and recommendations regarding what to elicit and how to proceed in a logical order. While the former type of best practices is the subject of almost every textbook on RE (e.g., Sommerville and Sawyer [SS97], or Robertson and Robertson [RR99]), the latter aspect is rather neglected in research but of special interest in this thesis. The reason is that we aim to establish problem-driven elicitation in ARE (see section 1.3.2), which means that we have to prescribe the order of issues to be discussed in an ARE process.

Basically, in each software development project, requirements form the basis for communication, contracting, development, integration, and maintenance, but also for employee satisfaction or rationalization [Rup07], and *"are the things you should discover before starting to build your product"* [RR99]. Even if there is no universal definition of requirements [Wie05], *"requirements (basically) express the needs and constraints placed on a product that contribute to the solution of some real world problem"* [Swe04]. Hence, requirements are descriptions of what a *"product must do or a quality that a product must have"* [RR99] in order to achieve a certain goal. As mentioned in one of the previous subsections, the purpose of a requirement is the specification of a system.

By means of intensive expert discussions, we have elaborated the interconnections of the concept "requirement" with other important concepts in RE (see Figure 13). According to this figure, the source of a requirement is a stakeholder who has a certain goal to be satisfied by a system. In this thesis, we use these terms as follows:

Definition – Stakeholder

"A stakeholder is a person or organization who will be affected by a system or who has a direct or indirect influence on a system's requirements." [KS98]

Definition – Goal

A goal is a target state in the future that is worthwhile being achieved or kept and whose satisfaction requires the cooperation of a system and its environment. [Lam04] [Rup07]

In contrast to other approaches, mainly from the area of goal orientation (see, for instance, the work of Lamsweerde [Lam04]), goals are explicitly not considered as requirements in this thesis. Rather, they describe an intended state that is to be achieved when a system is put in place.

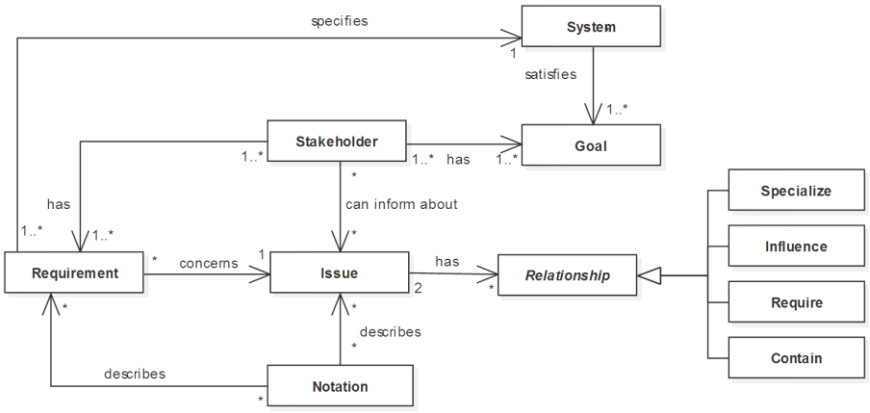


Figure 13. Basic terms of requirements engineering

Regarding its content, a requirement is always concerned with an issue according to the aforementioned definition. These issues cover functional and non-functional system aspects (e.g., human system activities), but also elements of the usage environment for which a system must be designed in order to provide appropriate support (e.g., users, work places, devices, data, business processes, etc.). An example of a requirement concerned with the issue "human system activity" could be: *"The system must support the purchase of a ticket that should work as follows: […]"*. Hence, each requirement addresses an issue that has to be supported or implemented by the system; in this example the concrete human system activity "Buy Ticket".

For each kind of issue, there is typically a stakeholder (group) that can provide information about the details of an issue. This is indispensable when defining the corresponding requirements. Furthermore, each issue and its related requirements can be described with certain notations (e.g., BPMN for business processes) in order to provide more clarification than just spoken words. Thus, notations to be used for clarifying requirements and issues are also a certain kind of best practice.

Definition – Notation

A notation is a series of signs or symbols used to represent quantities or elements in a specialized system [FD12].

Furthermore, as issues reflect elements of the real world, there are relationships among them, which may influence the order in which different issues should be discussed in an ARE process. The knowledge about

these relationships can therefore be seen as another kind of best practice to consider. Based on the work of Goknil et al. [GKB08] [VMT07], the following relationships between issues are distinguished in this thesis:

Definition – Contain (Relationship)

A contain relationship is a relationship between issues that expresses that the contained issue is a structural part of the containing issue [GKB08].

Definition – Require (Relationship)

A require relationship is a relationship between issues that expresses that the required issue is not a structural part of the requiring issue but needed for (correctly) fulfilling, implementing, or executing these instances [GKB08].

Definition – Influence (Relationship)

An influence relationship is a relationship between issues that expresses that the existence of the influenced issue is affected by the existence of the influencing issue [VMT07].

Definition – Specialize (Relationship)

A specialize relationship is a relationship between issues that expresses that the specialized issue is a specific type of the specializing issue enabling a classification of different issues.

To sum up, the issues with which requirements can be concerned, their relationships, the stakeholders who can provide information about them, and the notations that are suitable for their clarification should be described explicitly in order to serve as concrete "RE best practices" input for the thesis tailoring approach. For this purpose, we have developed an issue model that covers the typical elements of interest in the IS area (see chapter 5).

2.6 Requirements in Application Engineering

2.6.1 Relevant Requirements

As shown above, information needs describe the necessity of having information about certain issues in order to make decisions during the development process. We call the issues with which an information need is concerned "relevant issues", and consider them as those for which requirements must be elicited before a certain development phase can start. In particular, all other issues do not need to be addressed in a requirements process, as the requirements concerned with them only con-

tain information that does not influence any decision at all. Hence, we define a relevant issue as follows:

Definition – Relevant Issue

A relevant issue (for a certain development process) is an issue with which at least one information need is concerned (in this development process).

However, besides information needs, the relationships between issues may also determine whether a certain issue is relevant or not (see right part of Figure 14). If, for instance, business objects have been determined as a relevant issue according to the aforementioned definition, business activities may also be relevant, as (due to a potential "require"-relationship) concrete business objects could not be identified without knowing the business activities for which these objects are needed.

Furthermore, it is apparent that the relevance of an issue with regard to the development process does not imply that all requirements concerned with this issue are also (economically) feasible. Rather, the feasibility of a requirement depends on the flexibility and the existing elements of the product line architecture, the way systems should be developed according to the intended development strategy, as well as the technology used for this development. For instance, when the best possible realization of the architectural element "database system" only supports a response time of 0.1 seconds for non-nested queries, requirements that ask for 0.01 seconds are not feasible within the given SPL scope, as otherwise the entire architecture would have to be adapted.

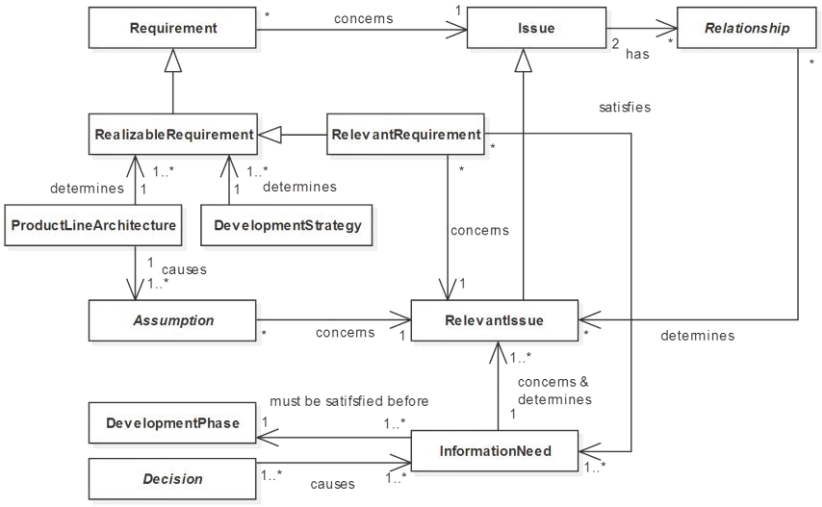


Figure 14. Origin of realizable and relevant requirements

In Figure 14, we have therefore integrated some parts of the views presented above in order to show how different elements from the reuse

asset base and from the development strategy influence the feasibility of requirements in an AE project. In this regard, we define a realizable requirement as follows:

Definition – Realizable Requirement

A (economically) realizable requirement is a requirement whose satisfaction does not require a violation of the product line architecture, any principles of the development strategy, or the constraints of the development technology used.

We define a realizable requirement in this way because a violation of a product line architecture, a strategic principle, or a technical constraint will usually result in unjustifiable extra costs and significant project delay. Thus, as an SPL intentionally aims to avoiding high costs and delays, only realizable requirements shall be accepted during AE in order to actually benefit from this approach.

When combining the notion of relevant issues with the notion of realizable requirements, it is apparent that a development process should ideally only get realizable requirements that are concerned with relevant issues as input. Requirements that are realizable but do not affect any decision are unnecessary for development, while requirements that are not realizable but concerned with a relevant issue are problematic. Thus, in the best case, only relevant requirements should be elicited in AE.

Definition – Relevant Requirement

A relevant requirement is a realizable requirement that is concerned with a relevant issue.

2.6.2 Anticipated Requirements

In SPLs, feasibility is often related to anticipation. In rather configurable SPLs, for instance, almost all (explicitly) anticipated requirements are already satisfied by corresponding realizations during DE/FE.

However, as introduced in chapter 1, AE must also deal with implicitly anticipated requirements and non-anticipated requirements. Below, we now introduce these terms, and also explain their relationship to the sets of requirements known from traditional product line engineering such as common requirements, variable requirements, and specific requirements.

Definition – Explicitly Anticipated Requirement

An explicitly anticipated requirement is a realizable requirement that has been explicitly addressed by product line architectural elements or corresponding realizations during the DE/FE phase already.

Definition – Implicitly Anticipated Requirement

An implicitly anticipated requirement is a realizable requirement that is not an explicitly anticipated requirement, but that belongs to a supported flexibility class and fulfills all given assumptions.

Definition – Non-Anticipated Requirement

A non-anticipated requirement is a requirement that is neither an implicitly anticipated requirement nor an explicitly anticipated requirement.

Definition – Common Requirement

A common requirement is an explicitly anticipated requirement that is satisfied in each system that is derived from an SPL by default.

Definition – Variable Requirement

A variable requirement is an explicitly anticipated requirement that is not a common requirement, i.e., it is only satisfied in some derived systems.

Definition – Specific Requirement

A specific requirement is a requirement that is not an explicitly anticipated requirement, i.e., either an implicitly anticipated requirement or a non-anticipated requirement.

According to these definitions, anticipated requirements are always realizable requirements because their anticipation during the DE/FE phase has led to architectural constructs that support their economic satisfaction. An anticipated requirement that is not feasible does not exist according to these definitions. However, a non-anticipated requirement does not automatically imply that it is not feasible. Rather, non-anticipated requirements can be non-realizable and incidentally realizable as well.

2.6.3 Elicited Requirements

While the aforementioned sets of requirements just cluster the infinite space of requirements that may occur during an AE project, the intersections of these requirements with typical elicitation results (called elicited requirements) are discussed below. The purpose of this discussion is to explain the practical problem of this thesis more formally.

In Figure 15, a complete and disjunctive classification of elicited requirements is shown.

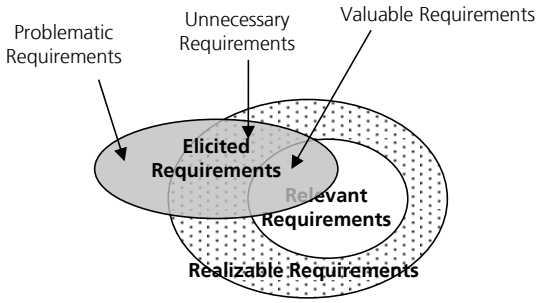


Figure 15. Typical types of elicited requirements

Below, we define these types and explain their implications on the development process.

Definition – Problematic Requirement

A problematic requirement is a non-realizable requirement that has been elicited in a requirements process.

Definition – Unnecessary Requirement

An unnecessary requirement is a realizable but non-relevant requirement that has been elicited in a requirements process.

Definition – Valuable Requirement

A valuable requirement is a relevant requirement that has been elicited in a requirements process.

According to these definitions, problematic requirements can only be requirements that were not anticipated during DE/FE and that are not incidentally realizable. Problematic requirements typically result from not considering the SPL characteristics and thus, from giving stakeholders too much freedom when defining their requirements. However, as state of the art approaches only distinguish between explicitly anticipated requirements and specific requirements, detecting problematic requirements during a requirements process is not systematically possible yet.

Besides problematic requirements, unnecessary requirements can often be found in practice as well. The term "unnecessary" implies that the requirements are not needed because they address, for instance, only elements that are included in all developed systems by default anyway (common architectural elements). Unnecessary requirements therefore provide information that no one on the developer side needs. Thus, the elicitation and specification of these requirements is wasted project effort. However, state of the art approaches typically also elicit those requirements because they follow generic best practices, and are not focused on the actual information needs of a certain development process.

Valuable requirements are finally those elicited requirements that are actually important for a development process: they are needed for making decisions and feasible within the given SPL scope. However, the existence of valuable requirements in a system specification does not imply that all decisions can actually be made. Thus, omitted requirements (missing requirements) are another class of requirements that may affect the efficiency of AE. Basically, requirements can be missing with regard to the satisfaction of customer goals and with regard to the fulfillment of the developers' information needs. However, as we make the assumption that a problem-driven elicitation approach has been chosen and that this approach is able to assure goal satisfaction (see section 1.3.5) implicitly, missing requirements are only defined from the development perspective here. In particular, when developers do not get all information they actually need for making decisions, they have to make their own assumptions or perform re-elicitation, both with risks for the project.

Definition – Missing requirement

A missing requirement is a relevant requirement that has not been elicited in a requirements process, but that is needed for satisfying an information need in the development process because no elicited requirement is able to satisfy this need.

2.7 Requirements (ARE) Process

In the previous subsections, both the elements of the input artifacts for the tailoring, and the possible results of an ARE process have been elaborated. As we now know which product- and process-oriented SPL knowledge and RE best practices have to be reflected, and are aware of what the ideal outcome of an ARE process should be, we can now introduce the elements of an ARE process in this section.

As introduced in Figure 10, AE includes both a development process and a requirements process, which are basically aligned via the production and consumption of requirements. The purpose of the requirements process is therefore the provision of a set of requirements that must exist for development. To make this happen, a requirements process has to prescribe all activities that are necessary for the elaboration of these requirements (see Figure 16). Hence, this process will have a similar structure as the development process introduced before.

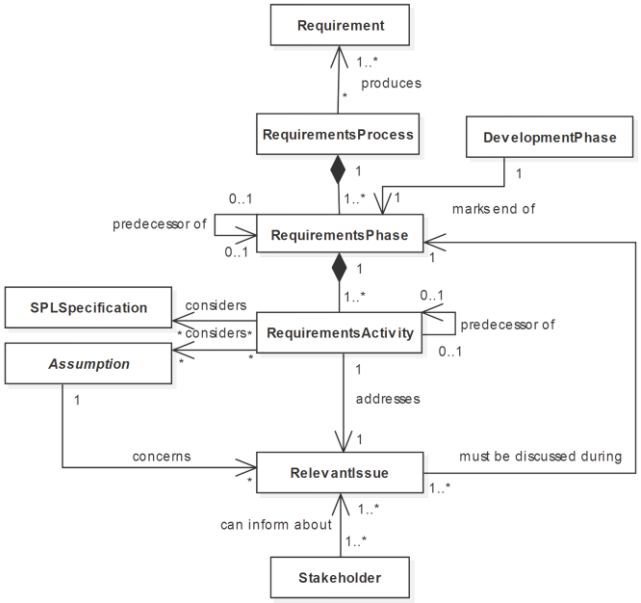


Figure 16. Structure of requirements process and its interplay

In this analogy, we define requirements phase and requirements activity as follows:

Definition – Requirements Phase

A requirements phase is a fixed period of time wherein certain requirements activities are performed.

Definition – Requirements Activity

A requirements activity is a procedure that creates a set of requirements concerning a certain issue that is relevant for development.

While the activities and phases of the development process are mainly driven by the product line architecture as well as the underlying development strategy, the activities and phases of the requirements process are determined by the development phases and the information needs that exist within these phases.

Thus, even if each requirements process is instantiated at the beginning of an AE project, it is a continuous *"no end"* [RR99] activity and, due to the above-mentioned product orientation, not just a front-end step that ends when development starts. Rather, the development phases determine different phases of the requirements processes, while the information needs of the development phases define the issues to be addressed in each of them (see Figure 14 and Figure 16).

To illustrate this procedural view, the above-mentioned database example is taken again. During a development phase "Infrastructure setup", which is assumed to be the first development phase according to a given development strategy (e.g., "Install the infrastructure components before starting to implement the business logic"), the role "database engineer" has to perform the development activity "integrate database system". Hence, before the development phase "Infrastructure setup" can start, a requirements phase must be finished in which relevant requirements for the database integration were elicited.

Besides the relevant issues, the SPL characteristics must also be considered explicitly in each requirements activity in order to achieve this aim. In particular, the consideration of SPL characteristics is indispensable to initiate negotiation or specific analysis as soon as the expectations of a customer seem to contravene the given constraints. However, whether these SPL characteristics have to be considered in the form of an SPL specification or in the form of simple assumptions (see Figure 16) depends on the anticipation of requirements concerning these issues (i.e., explicitly anticipated vs. implicitly anticipated). In chapter 4, which deals with the definition of an ARE instructions template, we will explain under which circumstances which artifact should be chosen.

2.8 Summary

When developing ARE processes, different artifacts must be considered for extracting the required product-oriented and process-oriented SPL knowledge as well as RE best practices. In this chapter, we have therefore elaborated the elements of these artifacts and their relationships, which are a prerequisite for systematically developing the actual thesis approach in chapter 6.

Regarding product-oriented knowledge, the flexibility classes and their assumptions, in particular, have been identified as being important in an ARE process. However, to elaborate them in a systematic manner, the elements and element types with which they are concerned must first be clarified in the given product line architecture.

With regard to process knowledge, the decisions and information needs within the intended development process have been recognized as elements affecting ARE. However, to identify them, the development activities have to be processed systematically first, which again requires strong consideration of the product line architecture. This is because the development activities do not depend on generic tasks only, but mainly on the product line architectural elements that have to be instantiated in an AE project.

In the context of RE best practices, the notion of issues has been introduced as an important concept. Issues describe the inherent elements of the real world with which requirements are concerned. Knowing them and their relationships is therefore a crucial basis for deriving logical elicitation sequences. In chapter 5, an issue model for IS is therefore introduced.

Regarding the requirements that exist in the context of AE, different sets have been defined. In this context, relevant requirements have been identified as the requirements which an ARE process should strive to elaborate ideally, as these requirements are feasible and relevant for making development decisions. Hence, when eliciting a complete set of relevant requirements only, no rework or re-negotiation should be needed.

An ARE process that is able to achieve these requirements in a constructive manner must therefore include activities that address all issues with which an information need is concerned. Furthermore, the assumptions that are made on these issues, as well as explicitly anticipated requirements from the SPL specification have to be considered in the corresponding activities. In chapter 4, an ARE instructions template is introduced that supports the reflection of this procedure.

However, before introducing these solution components, we will first analyze related work in the next chapter.

3 Related Work

„We often see something for hundred or thousand times before we really see it for the first time."
Christian Morgenstern

This chapter gives an overview of related work, both with regard to the underlying practical problem of this thesis and with regard to its research questions. The purpose of this chapter is to show which similar approaches have already been proposed, and to which degree they are able to meet the goals of this thesis.

3.1 Research Approach

The related work described in this chapter has been identified and analyzed according to the systematic literature review approach proposed by Kitchenham [Kit04]. Thus, the first step was the definition of a review protocol defining the review goals, review questions, search strategy, as well as the exclusion and assessment criteria[3] (see Appendix A). According to these protocols, the questions to be answered by the literature review were:

1. Which work exists that aims at providing effective RE for the AE phase?

2. Which work exists that aims at providing effective tailoring or reengineering of RE processes?

3. Which work exists that aims at providing effective guidance for requirements elicitation (in interviews)?

In order to answer these questions, the defined and tested search strings were applied on the digital libraries IEEE Xplore, ACM, and Science Direct (step 2). The reason for using only these libraries was that they in-

[3] The assessment criteria were derived from the goals of the thesis introduced in chapter 1.

clude most of the relevant work in the RE area and also cover, for instance, the LNCS proceedings from Springer in which many related papers are published. Thus, a direct search at Springer Link or other sources was not necessary. Unfortunately, we found that many hits did not address the context we were looking for. Thus, by reading the publications' abstracts and titles, and comparing them with the exclusion criteria defined in advance, many proposed papers were sorted out directly.

In a third step (step 3), the remaining papers were then read and classified into basic papers (describing more general issues regarding the topics of interest), and candidate papers (describing a real approach addressing one of the research questions). The candidate papers were then investigated according to the defined assessment criteria (see Appendix A) and briefly summarized. Wherever possible, we used assessments from existing literature reviews and did not analyze each paper from scratch. In a fourth step, we then checked the reference sections of the aforementioned papers in order to find further resources that might be interesting (step 4). Furthermore, suitable papers and books that were found by chance during other research activities were also taken into consideration. These new hits were then also investigated according to the defined criteria.

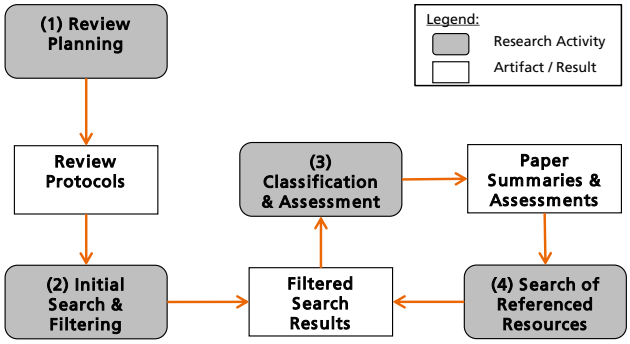


Figure 17. Research approach for related work review

However, this chapter does not claim to provide a complete literature review even though corresponding methods were applied. Rather, this chapter aims at providing an initial basis for understanding how the thesis fits into, respectively may enhances the state of the art.

3.2 Application Requirements Engineering

In this section, we will attempt to describe existing work aimed at providing effective RE for the AE phase.

While much research effort has been expended on how to build SPLs during DE/FE, the AE phase has not received sufficient attention yet [PKG+08] [RGD07] [RD07]. Especially for ARE, only some initial work exists so far. Thus, even a systematic literature review on RE for SPLs done by Alves et al. [ANA+10] in 2009 does not discuss any ARE method, even though this topic was not explicitly excluded.

In the literature review of Rabiser et al. [RGD09], which investigated the state of the art in product derivation, this underrepresented role of ARE was also noted. With regard to the scope of this thesis, especially the following issues were identified as being unresolved in the 13 AE approaches addressed in their review:

- **Inadequate knowledge.** Application requirements engineering heavily relies on SPL expert involvement as the tacit knowledge regarding supported variability / flexibility and available reusable assets can hardly be captured completely in explicit models. In the Kobra approach [ABB+02], for instance, it is mentioned that the requirements engineers *"should lead the discussion to ensure that the information needed for the cost-effective instantiation is obtained."* However, as elaborated in the review done by Bühne et al. [BHL+06], existing approaches do not sufficiently explain how requirements engineers should be provided with the required knowledge beyond the predefined variants.

- **Weak RE support.** Even though it is widely acknowledged that many (non-explicitly anticipated) requirements may arise during AE, capturing and aligning these requirements with existing SPL capabilities is a difficult task. In particular, customers are too much influenced by predefined SPL requirements instead of being enabled to state their real needs and wishes [DS07]. This is mainly caused by the fact that negotiation and elicitation support (i.e., guidance) is still weak. It is therefore difficult to predict the costs of customer-specific requirements [ORR+09] and to assure a consistent specification [DS07] at the same time.

In most existing AE approaches, the need for RE is just highlighted without any information on how this should be done (e.g., [DSB05]). Basically, only the tasks of communicating the variability (see, e.g., [HP03]), selecting variants, specifying the system requirements, and supporting trade-off decisions are typically proposed as being important in this context [Poh07]. In the COVAMOF [SDH06] or PuLSE-I [BGM+00] approach, for instance, it is even assumed that the customer requirements are already available and that they have been elicited *"like it is done in single systems"* [BGM+00]. Thus, these AE approaches do not include RE at all.

Due to the low availability of ARE approaches, we have therefore only found three publications that seem to address ARE sufficiently. In order

to assess their suitability for the problems to be solved by this thesis, the following criteria were checked:

Problem Orientation. Does the approach support problem-oriented elicitation?

Elicitation and Negotiation. Does the approach include elicitation and negotiation activities?

Customer-specific Requirements. Does the approach explain how to deal with customer-specific requirements beyond the predefined variants?

Customization. Is the approach customizable based on the given reuse asset base?

Validation. Has the approach been empirically validated?

Precision. Does the approach provide precise guidance?

Applicability. Is the approach applicable for information systems?

Below, we briefly summarize and assess each of these approaches.

3.2.1 SARE

The "Scenario-based Application Requirements Engineering" approach (which we denote as SARE due to a missing official abbreviation[4]) proposed by Bühne et al. [BHL+06] is quite a comprehensive approach for the development of application-specific requirements specifications. The approach is based on a variability model (VM) with associated usage scenarios and covers the elicitation, negotiation, documentation and validation of requirements in AE projects. While the VM is used to guide the elicitation and communicate the basic variants of the SPL, the associated scenarios are used to give customers an idea of the later usage when a certain variant will be implemented.

The entire RE process according to this approach works as follows: Initial stakeholder requirements are elicited in a first step. Based on this, the requirements engineers instantiate a scenario via the VM and communicate this scenario (and related requirements) back to the stakeholders. If the scenario fulfills their expectations, the selected variants are documented. Otherwise, the delta is recorded and used for a subsequent negotiation step. During this negotiation, the requirements engineers propose alternative variants to the stakeholders, who are then asked to make a trade-off decision. This means that the stakeholders either accept a certain alternative or insist on a customer-specific solution. The corresponding evaluation results (e.g., impact on costs or time) are then estimated and communicated by the requirements engineers. After elici-

[4] If no official abbreviation exists, we will introduce our own abbreviations in this thesis.

tation and negotiation, the finally agreed requirements are then documented and validated. In these steps, the VM and the predefined scenarios are used again.

As the entire ARE process is driven by the VM (and not by problems or goals to be solved, respectively achieved) the approach is rather solution- than problem-oriented. However, elicitation and negotiation are clearly supported, and the handling of customer-specific requirements is also addressed explicitly. Furthermore, as both the VM as well as the associated scenarios are part of the underlying SPL, the approach is customizable in this regard. However, the preciseness of the approach, at least in the available publication, is very low. Guidelines for requirements engineers that support them in performing the elicitation, negotiation, documentation and validation activities are not mentioned in the paper.

Finally, the approach has been developed, applied, and validated in the context of automotive systems only. Thus, it is not possible to claim that it is also applicable in the IS area.

3.2.2 RED-PL

The RED-PL approach proposed by Djebbi and Salinesi [DS07] aims at deriving consistent application requirements from an SPL requirements model during AE projects. However, in contrast to SARE, the approach explicitly aims to provide users with the possibility to express their real needs and wishes using traditional RE techniques. These needs are then matched with SPL capabilities, followed by a corresponding negotiation step.

The entire RE process according to this approach works as follows. In a first step, requirements are elicited from customers using any kind of traditional (top-down) approach. These requirements are then matched with SPL requirements. Based on defined constraints between these SPL requirements, an optimal and consistent set of application requirements is derived. If conflicts exist between customer requirements and (predefined) SPL requirements, negotiation and (re-)elicitation take place. Thus, these three activities are performed in an iterative manner.

As the approach explicitly aims at consuming requirements that have been elaborated using traditional RE techniques, it seems to support problem-oriented elicitation. Thus, the approach is apparently able to deal with customer-specific requirements. However, as the actual elicitation is not part of the approach, and only the matching and negotiation (called arbitration) steps are supported and elaborated in the available publication, the entire ARE process cannot be customized based on a given SPL. As a consequence, no precise guidance is provided either, at least not for the elicitation activities.

The RED-PL approach has been validated in an industrial case study in the medical device domain. It is therefore not possible to claim that it is also applicable in the IS area.

3.2.3 DOPLER-UCon

The DOPLER-UCon approach proposed by Rabiser and Dhungana [RD07] aims at integrating product configuration and RE in AE. For this purpose, the approach comprises a tool-supported method that covers all steps of product derivation ranging from the enhancement of VMs with sales-relevant knowledge via product simulation to product deployment. Within the approach, requirements elicitation and negotiation activities are closely intertwined with product configuration and simulation.

The entire RE process according to this approach works as follows: In a first step, initial requirements are taken from customers and used for pre-configuring and simulating an application. This simulation gives the customers the opportunity to directly review the proposed solution and provide corresponding feedback. Based on this feedback, changes to the configuration are then made or additional wishes and requirements are elicited from scratch, if necessary. During this refinement, the development engineers are responsible for checking which of these customer-specific requirements a) can be realized with the existing assets, b) will imply additional development, or c) are not feasible at all. The tool supports these tasks by tracking influence relationships to existing assets and decisions. The negotiated and agreed set of requirements is finally documented and taken as input for the actual application development and deployment.

As the approach starts with product configuration and simulation and offers the possibility to elicit and negotiate customer-specific requirements only on demand, it is basically not a problem-oriented approach. Rather, a specialized VM (called derivation model) is the driver of the requirements process. Even though the approach explicitly addresses and supports the elicitation and negotiation of customer-specific requirements, how this should be done is not clearly explained, at least not in the available publication. However, as the approach takes a tailored VM of a given SPL as input, it is highly customizable in this regard. In particular, the approach can be tailored not only based on SPL characteristics, but also based on the specific characteristics of a certain customer project.

The DOPLER-Ucon approach has been validated in plant automation projects. Thus, it is not possible to claim that it is also applicable in the IS area.

3.2.4 Assessment Summary

In this section, the assessed approaches are briefly summarized (see Table 1).

	Problem orientation	Elicitation and negotiation	Customer-specific requirements	Customization	Validation	Precision	Applicability
SARE	-	+	+	+	+	-	?
RED-PL	+	O	+	O	+	-	?
DOPLER-Ucon	-	+	+	+	+	-	?

+: fulfilled, O: partially or implicitly fulfilled, -: not fulfilled, ?: unclear

Table 1. Assessment summary of existing ARE approaches

Even though all considered ARE approaches support the handling of customer-specific requirements, elaborating them in a problem-oriented (and state of the art) manner is not common yet. Similar to approaches that do not deal with RE at all, state of the art ARE is typically organized around predefined decision-, feature-, and domain models rather than being focused on the actual needs to be addressed.

Furthermore, the existing ARE approaches have only been developed and applied for embedded systems so far. Thus, it is unclear whether the concepts proposed by these approaches also work for IS.

Finally, the preciseness of the considered ARE approaches is very low – similar to most approaches in single system development. Thus, they just describe what has to be done without explaining how. Clear instructions describing how elicitation and negotiation can be performed concretely, for instance, are not given. As the corresponding publications also do not mention the existence of more precise method guidelines, we assume that the approaches are actually not described precisely to allow successful execution by non-experts. This is interesting insofar as Rabiser et al., for instance, mentions that better *"guidance and support are needed to increase efficiency and to deal with the complexity of application engineering"* [RGD07].

Elaborating and aligning real customer requirements with the capabilities of a given SPL and finding a compromise between reuse and customer satisfaction is therefore an ARE problem that has not be resolved satisfactorily yet. Hence, this thesis tries to provide a solution for this aim.

3.3 Requirements Process Tailoring

In this section, we try to identify existing work that aims at providing effective tailoring or reengineering of RE processes. In this regard, approaches that tailor RE processes by incorporating domain-specific process knowledge as well as those that tailor RE processes by incorporating domain-specific product knowledge are considered. While the former capture knowledge in decisions or tasks to be made, the latter capture knowledge in artifacts to be reused explicitly during a process [Mut02].

Basically, *"every project needs a different RE process for the simple reason that every project is different"* [RR99]. Therefore, RE processes should always be tailored to their actual usage context and strongly individualized with regard to *"the type of applications being developed, the size and culture of the companies involved, and the software acquisition processes used"* [Som05]. Tailoring must therefore deal with determining how a process will produce the deliverables especially by whom, in which order, in which form, at which location, and with which quality gates, in order to meet certain context characteristics[5].

In the context of SPLs, AE processes are typically guided and supported by tailored production plans defined during the DE/FE phase. A production plan describes how applications are developed from the core assets of an SPL and is a guide for application engineers [CN01]. However, as production plans remain very informal and merely describe the inputs, activities, roles and outputs when deriving products from an SPL, they do not provide precise guidance on how to perform individual steps, at least not within ARE. In particular, specific knowledge that is important for the early phases such as sales or RE is not captured and represented in them [RGD07].

In particular, checking and assuring completeness remains a huge challenge also in a tailored requirements process, as completeness is considered to be the most difficult specification characteristic in RE [ZG03]. Especially because *"we cannot specify everything"* and need only specify *"what the developers cannot guess"* [Lau02], absolute completeness is neither necessary nor justifiable from an economic point of view. Hence, tailoring must also deal with determining the information that is actually required for later project steps, and assure that ideally, only this information is actually elicited during an RE process. For the same reason, feasibility issues must also be considered. This means that tailoring does not only have to deal with determining required information, but also with identifying (technical) constraints that limit the solution space to be addressed.

[5] A good overview of software process tailoring approaches in general is given by Alegria et al. [ABQ+11].

Thus, with regard to ARE processes, no specific tailoring approaches exist yet. Indeed, we have identified many publications that aim at reengineering requirements from legacy systems or related documentations as input for DE/FE. However, reengineering or tailoring approaches for ARE processes were not found. Below, we have therefore only listed publications that deal with RE tailoring or requirements reengineering outside the area of SPLs. In order to assess their suitability for the problems to be solved by this thesis, the following criteria were checked:

Requirements Process. Does the approach lead to a requirements process?

Best Practices. Does the approach incorporate and reflect state of the art RE and consider best practices (process knowledge)?

Information Needs. Does the approach address the incorporation and reflection of information needs (process knowledge)?

Development Strategy. Does the approach consider the development strategy (process knowledge)?

Capabilities and Constraints. Does the approach address the incorporation and reflection of existing capabilities and constraints (product knowledge)?

Validation. Has the approach been empirically validated?

Precision. Does the approach provide precise guidance or even automation support?

Applicability. Is the approach applicable for information systems?

Below, we briefly summarize and assess each of these approaches.

3.3.1 REPKB

The methodology for RE process development based on an RE process knowledge base (REPKB) proposed by Jiang et al. [JEF04], provides a framework that aims at developing most suitable RE processes for a given context. For this purpose, the approach comprises a process knowledge base containing knowledge about experience, templates, best practices, etc. Furthermore, a decision support system based on case-based reasoning, a process development methodology, and evaluation models are provided.

The entire tailoring process according to this approach works as follows. In a first step, the characteristics of a certain project or development situation are identified. These characteristics and constraints determine the selection of RE process building blocks, templates, and techniques as well as the guidelines for the actual process development when using the case-based reasoning component. The recommended entities found

in the knowledge base are then presented to the method tailors, who then compare the different alternatives. Based on the decisions made, the selected process building blocks or templates (reference processes) are then combined in order to develop a suitable RE process. This process is finally evaluated and stored as a new template in the REPKB.

As the goal of this approach is the definition of RE processes, and the process template as well as the process building blocks (e.g., activities, techniques, etc.) reflect best practice in this regard, the first two assessment criteria are apparently fulfilled. However, the approach does not include any step that deals with the incorporation and reflection of information needs existing in a given (project) context. With regard to the development strategy, the approach provides no specific support either. The consideration of project context characteristics may partially address this aim. However, it is not done explicitly, and at least no alignment of RE process steps and development process steps is performed. In addition, with regard to the incorporation and reflection of reuse capabilities and constraints, the approach does not provide any guidance. The process building blocks dealing with reuse are not further explained.

The REPKB approach is not limited to a certain domain and is therefore also applicable for IS. Furthermore, it seems to be validated, even though only an informal case study has been published. Precise guidelines for process definition, process tailoring, as well as technique selection are mentioned as being available.

3.3.2 REPI-IM

The RE process improvement based on an information model proposed by Doerr et al. [DPK04], which we abbreviate with REPI-IM here, provides an approach for the tailoring of (existing) RE processes based on the responsibilities and information needs of the involved people. The idea is to define an information model that captures the documents created and used by the project stakeholders, and to derive the RE process based on the exchange of these documents. Furthermore, the approach allows determining the content a document should have, and, thus, the elicitation and specification activities that are needed in a process.

The entire tailoring according to this approach works as follows. In a first step, the typical stakeholders involved in a project are identified and described. Furthermore, an initial identification of problems within the current RE process takes places. In a two-day workshop, these problems are then elaborated together with the identified stakeholders. Furthermore, a first information model is created, which explains which requirements are provided by whom on which level of abstraction. Taking into account the existing problems, this initial information model is then redefined. In particular, responsibilities and exchange relationships are defined, and the document details (e.g., the required content, quality criteria, etc.) are

defined. These decisions then result in a clear definition of document templates, process flows, tasks, and responsibilities within the RE process.

As the approach deals with the improvement of RE processes, the first assessment criterion is apparently fulfilled. Best practices are not considered in the approach, except for those practices that are already implemented in an organization that uses this approach. Rather, REPI-IM aims at improving the quality of the documents as well as their exchange relationships. The development strategy is only implicitly covered by considering how documents are exchanged today. Capabilities and constraints that might restrict the feasibility of requirements are not explicitly elaborated in the approach either. Only procedural aspects and specification quality criteria are incorporated into the document templates.

The approach has been successfully applied in the smart traffic domain. However, the general nature of the approach seems to allow its usage for IS, too. Unfortunately, the precision of the approach is low, and success appears to depend mainly on the skills of the workshop moderator.

3.3.3 EVECR

The approach for extracting viewpoints for eliciting customer requirements based on an analysis of specification change records proposed by Aoyama et al. [AUY+07] provides a concept for deriving elicitation checklists. The motivation for this work is that existing elicitation guidelines are not precise enough to gather all information that is important for avoiding misconceptions, which often lead to late change requests. Previous change requests should therefore be analyzed in order to determine check items that should be considered more thoroughly when eliciting and describing requirements in future projects.

The entire tailoring process according to this approach works as follows. In a first step, existing change requests are rephrased in order to increase their understandability, also for third parties. Then the reasons for the change request are analyzed and validated, if necessary (a typical reason is that some aspects of the system environment were not considered sufficiently during elicitation). In the third step, the process in which the reasons for the changes occurred should be identified. If this is an elicitation activity, then the viewpoint (i.e., the issue in the terminology of this thesis) that was discussed in it must be determined. If this is a design activity, then the implementation aspect (e.g., a certain component) is identified. Based on the identified reasons for the change, check items are defined and annotated to the viewpoints in order to provide more guidance. Requirements engineers are then better informed about the questions to ask or the facts to consider when eliciting the corresponding requirements.

Even though this approach deals with requirements elicitation, it does not lead to requirements processes, but just to a list of check items to be used during elicitation. In this regard, the approach does not include best practices either, as it only changes the artifacts to be used in elicitation. Hence, the existing (best) practices that are already applied in the organization may remain stable, which also holds true for the development strategy. Even though the approach does not elaborate all information needs to be satisfied, it identifies and addresses those information needs that have not been satisfied sufficiently before. With regard to the incorporation of capabilities and constraints, the approach does not mention any support. However, besides check items related to information needs, check items could theoretically also refer to constraints against which the requirements must be checked.

The approach has been validated by means of a simulation, but not in a real project so far. As it comes from the IS area, it is apparently applicable there. The precision of the approach is, as far as the publication can tell, low.

3.3.4 DOPLER

The approach for adapting and augmenting variability models (VM), which is part of the DOPLER framework proposed by Rabiser et al. [RGD07], aims at providing more information to people involved in early phases (sales and ARE) of an AE project. For this purpose, the VM of a given SPL should be pruned to the actually relevant size, while additional knowledge is incorporated in order to provide sales people with better information. In particular, guidance and hints should be added to each decision a sales person or requirements engineer has to make. This should support them in explaining the consequences of a certain requirement to their customers. Furthermore, the approach aims to provide additional sales recommendations when interacting with the customer.

The entire tailoring process according to this approach works as follows. In a first step, the VM is considered in order to see what the supported variabilities currently are. In a second step, the roles in the AE project are defined in order to determine their responsibilities with regard to the decisions to be made. Then, pre-defined products that may be a good basis for the discussion with a customer should be selected. Based on this, the whole VM should be pruned down to those variation points that are potentially of actual interest for a certain customer. Finally, the model is enriched with additional knowledge and proper guidance in order to support the sales people or requirements engineers in the best possible way when discussing the variation points with the customers.

The DOPLER approach does not really result in an RE process but provides information that can be used in it. Hence, the approach does not

incorporate best practice in this regard either. The information needs of the developers are also not addressed, and the development strategy is not considered. However, with regard to capabilities and constraints, the approach tries to incorporate them and reflects them on a non-technical basis. So far, these constraints and capabilities are only based on the explicitly anticipated variants described in the VM.

The approach was introduced in the domain of plant automation, but no validation is described. Whether it is applicable for IS is also unclear. The precision of the steps is very low. In particular, it remains completely unclear how the required sales knowledge should be extracted and incorporated.

3.3.5 MDE

The model-driven engineering-based (MDE) approach proposed by Alegria et al. [ABQ+11] is not specifically intended for RE processes, even though a case study is presented in this context. The idea of this approach is to use concepts from model-driven engineering and variability modeling when instantiating project-specific processes. As a basis for this aim, organizational processes must be defined with explicit variabilities, and context models have to be created for each project setting.

A prerequisite for making this approach work is that a general process model and transformation rules have been defined upfront. This can either be a reference process for a certain development discipline (e.g., RE) or for a certain organization. Thus, only the context model has to be defined individually for each concrete project or development context.

As this tailoring approach is a generic one, it may also result in RE processes for IS. However, due to this generality, a reference process that comprises best practices is not part of the approach. Furthermore, the technical rather than methodological nature of the approach also neglects the incorporation and reflection of information needs, development strategies, as well as capabilities and constraints. With regard to the precision of the approach, there is a divided result. On the one hand, the approach is precise enough to automatically generate processes. On the other hand, the steps such as the definition of the context model are not clearly described, at least not in the available publication.

The approach has been validated in the context of industrial RE processes and has therefore been proven to work.

3.3.6 Assessment Summary

In this section, the assessed approaches are briefly summarized (see Table 1).

	Requirements Process	Best Practices	Information Needs	Development Strategy	Capabilities and Constraints	Validation	Precision	Applicability
REPKB	+	+	-	O	-	+	+	+
REPI-IM	+	-	+	O	-	+	-	+
EVECR	O	-	O	O	O	O	-	+
DOPLER	-	-	-	O	O	?	-	?
MDE	+	-	-	-	-	+	O	+

+: fulfilled, O: partially or implicitly fulfilled, -: not fulfilled, ?: unclear

Table 2. Assessment summary of existing tailoring approaches

The investigated approaches dealing with the tailoring of RE processes (or, at least, of supporting artifacts) are all validated and seem to be applicable for IS.

However, it can be seen that most of them take neither the actual information needs of subsequent development roles nor the existing capabilities or constraints into consideration. The development strategy according to which systems should be developed is not explicitly addressed either and rather assumed to be met implicitly.

With regard to RE best practices, only one of the assessed approaches makes use of corresponding "building blocks", while the others do not take into account best practices beyond those already implemented. Furthermore, the precision of most approaches is low; hence, the success when applying them mainly depends on the persons who carry them out.

Incorporating and representing important process and product knowledge into an RE process is therefore still an insufficiently supported task, even outside the SPL area. Hence, this thesis tries to provide a better solution in this regard.

3.4 Elicitation Instructions

In this section, we try to identify existing work that aims at providing effective guidance for requirements elicitation in interviews.

Basically, requirements elicitation *"is the process of discovering the requirements for a system by communication with customers, users and others who have a stake in the system development."* [SS97]. However, elicitation does not only mean asking people what they would like to

have, but also finding out what they really need in order to solve their problems. Thus, problem orientation was also taken as an assessment criterion for the ARE approaches described above.

Even though there are many different techniques for requirements elicitation (see, for instance, [Lau02] for an overview), the most important and most straightforward one is still the interview [LW00]. Davis et al. [DDH+06] have even shown that interviews are probably the most effective way of elicitation. For interviews, even analyst experience does not appear to be a relevant factor. Of course, a prepared set of questions (closed interview) should be available in order to provide appropriate guidance.

However, the search and even the manual browsing for elicitation / interview instructions or guidelines in the aforementioned digital libraries, the Internet, or respective conference proceedings did not return any approaches that explain how to proceed concretely during an elicitation session. Even though the search terms (see Appendix A) led to many "hits", existing publications typically mention only very generic issues (if at all), such as "*come prepared to the interview*" and do not really provide strategies, guidelines, or even detailed statements on how to proceed. So far, there are only guidelines that explain how to transcribe the elicited requirements in a certain format, or how to derive requirements from goals or vice versa (see, for instance, [RSB98]).

Even though we know that there is some work in other disciplines such as social science or journalism dealing with systematic interviews, we did not find any publication that fulfilled our inclusion criteria (see Appendix A). In particular, we found no resource in which a concrete guideline for elicitation was presented (apart from very generic or example guidelines often shown in RE textbooks, such as in [LW00]). Neither did we find templates or strategies on how to structure an interview guideline, at least for the purpose of requirements elicitation. Thus, in contrast to the previous section, an assessment of existing work according to defined criteria is not possible.

3.5 Summary

Knowing the state of the art in a certain area is a prerequisite for sound research. In this chapter, existing work that is either related to the practical problem of this thesis or to its concrete research questions has therefore analyzed.

In the area of application requirements engineering (ARE), only a few approaches exist so far. While AE (or product derivation) is still an underresearched topic in the SPL community, ARE has received even less attention. The few existing approaches in this area still lack sufficient precision

in terms of clear how-to instructions. Furthermore, they do not promote problem-oriented elicitation either. This means that the variability or decision models of the underlying SPL are the driver for requirements development rather than the actual problems or goals of a customer. Another observation is that all existing ARE approaches have just been evaluated in the embedded domain so far. Whether they are also applicable for IS remains unclear.

A novel approach like the one to be developed by this thesis should therefore provide more precise guidance on how to perform (problem-oriented) RE in AE. Furthermore, the approach should be developed for and evaluated with IS in order to provide suitable support here. In this regard, however, it is important that the approach does not neglect established SPL concepts such as variability or decision models, but rather tries to integrate them wherever possible.

In the area of RE process tailoring, a couple of approaches were found. However, only one approach explicitly addresses information needs as input for tailoring. Its idea of taking the responsibilities of development roles as a trigger for the identification of information needs is a promising concept to be further investigated in this thesis.

With regard to capabilities and constraints, no approach deals with the systematic incorporation and reflection in RE processes. Even though the idea of using change requests ("bad experience") as a means for identifying items to be considered during elicitation is basically suitable in this regard, it is not within the scope of the analyzed approach yet. However, for this thesis, this notion might be interesting. The development strategy according to which systems should be developed is not explicitly addressed by the considered approaches either and is rather assumed to be met implicitly. The same holds true for RE best practices, which are only considered explicitly by one assessed approach. This approach is also the only one that provides precise guidance for the tailoring process. All other approaches are very informal and highly rely on the involved people.

The approach to be developed by this thesis must therefore provide much more precision and automation support. The experience made by [ABQ+11] that automatic generation of processes based on the elicited context characteristics is possible is a fruitful insight in this regard. Furthermore, RE best practices, the defined development strategy, as well as the given capabilities and constraints must be explicitly addressed. So far, this is apparently not the state of the art, yet.

In the area of elicitation guidance, no related work was found. Even though it is indispensable to prepare and use questions and guidelines to manage an elicitation session, no existing work describes what such guidelines / instructions should look like. Some RE textbooks provide examples of interview questions, but neglect to investigate more thorough-

ly how such questions or instructions should guide requirements engineers through an interview.

The approach to be developed by this thesis should therefore explain what elicitation instructions should look like and how they should be instantiated based on the characteristics of a given SPL. For this purpose, recapitulating and formalizing practical elicitation experience seems to be the preferred research strategy rather than combining non-tested ideas. In the next chapter, this thesis will therefore introduce a template for elicitation instructions suitable for the context of ARE, and in the following chapter the tailoring approach will be presented.

At this point, it has to be noted that systems built based upon components off the shelf (COTS) would also have been an important area for related work. However, these approaches are not sufficient either, as they just deal with the selection [Alv03] or adaptation [AFC+05] of COTS components, and do not provide any guidance on how requirements are to be elicited and negotiated in order to fit existing assets.

4 A Template for ARE Instructions

"Who does not forbid doing the wrong, command it."
Marcus Aurelius

As already described in the solution idea (see section 1.3.2), the overall purpose of this thesis approach is to tailor ARE instructions based upon an SPL's reuse asset base, the intended development strategy, and RE best practices in order to provide requirements engineers with better knowledge about a given SPL.

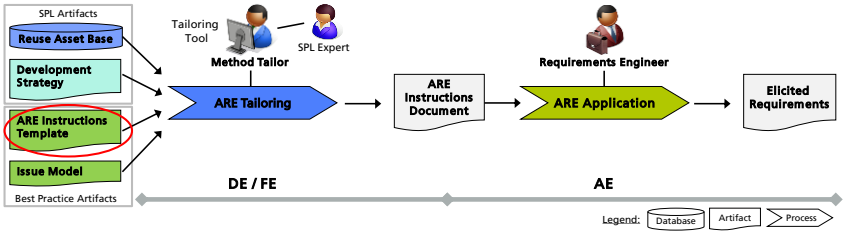


Figure 18. ARE instructions template within thesis approach

To support this aim, this chapter introduces a template for ARE elicitation instructions including a set of predefined text blocks. The purpose of this chapter is to explain what ARE instructions should look like ideally in order to improve the elicitation and negotiation activities of requirements engineers. In particular, the chapter explains how SPL knowledge can be represented in an appropriate way in order to provide information about capabilities and constraints. Thus, the resulting template provides a consolidation of RE best practices and can therefore be used as input for the automatic generation of ARE instructions during tailoring (see Figure 18).

4.1 Research Approach

For the definition of a template for ARE elicitation instructions (we denote them as ARE instructions in the following), different research steps were carried out (see Figure 19).

In a first step (step 1), hypothetic requirements regarding the content that ARE instructions should provide were derived from the foundation described in chapter 2. As we assumed this model to be complete (at least for the purpose of this thesis), we also assumed the derived hypothetic requirements to be complete in this regard. Furthermore, based upon a precise elicitation instructions document that we had developed some years ago in an industry project at Fraunhofer IESE [ARC08], we additionally derived a couple of hypothetic requirements regarding the general "nature" of elicitation instructions beyond their content. The reason for additionally using this document was that it had enabled people with low RE experience to perform rather good elicitation merely by following the described procedure. In addition, we performed a literature review, but we found that similar work has not been proposed yet (see section 3.4).

In a second step, we then performed a survey with eight experienced requirements engineers in order to elicit their requirements on ARE instructions (step 2). The survey was done by means of a questionnaire with open and closed questions. The open questions were used to gain new insights about the desired contents and properties of suitable ARE instructions. The closed questions (using the scale "totally disagree, …, totally agree") were used to get confirmation for the hypothetic requirements identified before. In this context, a hypothetic requirement was considered as confirmed if the median in the answers was at least "rather agree", and the minimum was not lower than "neither agree nor disagree".

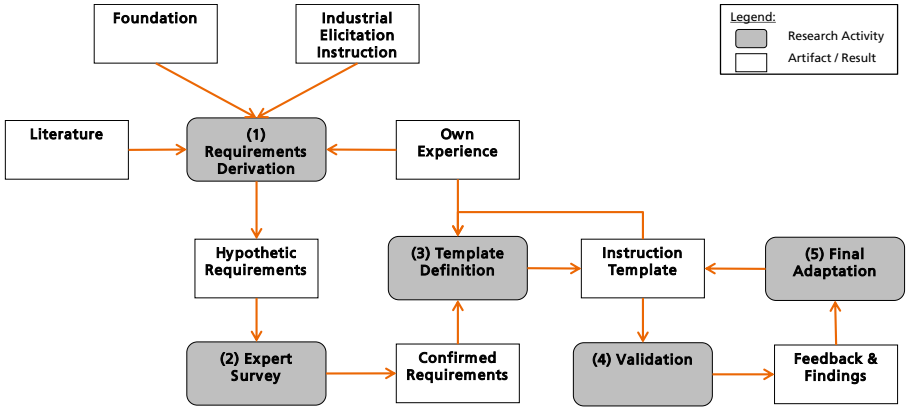


Figure 19. Research approach for ARE instruction template

Based on the confirmed requirements (see Appendix B), the template was then developed in a third step (step 3). A central task here was the definition of text blocks (called "phrases") intended to provide the required information. In order to choose appropriate formulations and determine rules regarding a meaningful order, we recapitulated, discussed, and formalized the aforementioned industrial elicitation instructions with which we had made good experience in a previous project. In a couple of iterations, and in close alignment with the development of the tailoring approach (see chapter 6), a tool for automatically generating concrete ARE instructions based on the template was then developed. During this development, we continuously increased the precision and correctness of our template.

In order to finally validate the template, a two-step evaluation approach was used (step 4). During a first study, eight RE experts used and reviewed an exemplary ARE instructions document according to the template in order to check whether it is basically applicable and useful. During a second study, which was also the final evaluation of the entire thesis (see chapter 7), 13 students used another ARE instructions document according to this template for eliciting requirements in a controlled experiment. The findings of both evaluations were used to make final adjustments to the template (step 5).

4.2 Template Overview

Requirements elicitation is an essential activity in RE. However, while elicitation practices and techniques as well as the thematic aspects (see issue model in chapter 5) are widely documented in the literature, concrete instructions on how to actually perform requirements elicitation are underrepresented in current work (see in section 3.4). Instead, most *"approaches that address requirements development usually lack sufficiently precise and prescriptive instructions"* [CA07], which is why these activities still depend on the persons carrying them out rather than on the selected techniques.

In order to make elicitation in AE more independent of the person filling the role of the requirements engineers, the use of precise and (almost) algorithmic ARE instructions is therefore proposed in this thesis. These instructions should support AE requirements engineers in achieving a good set of requirements by enabling them to ask the right questions in the right order, and to initiate appropriate reactions when requirements seem to contravene the given SPL. Thus, if AE requirements engineers have such instructions, they are expected to perform the elicitation more effectively, even though not all customer requirements were explicitly anticipated during DE/FE already. This is especially important for rather flexible SPLs in which not all requirements can be documented upfront.

To realize this notion, ARE instructions have to describe the issues for which requirements have to be elicited, in which order, and which assumptions must be considered during this elicitation. In contrast to the decision model or feature model questionnaires traditionally applied in AE, the really necessary requirements can then be systematically derived based on the given business problems (problem-oriented requirements). Thus, requirements that are already part of the SPL and those which are not are addressed in an integrated manner. For this purpose, a constraint-based rather than an enumerative description of the SPL is provided as part of the ARE instructions document. This notion also addresses the challenge that an explicit variability expression is often limited in rather flexible SPLs.

For defining the structure and content of such ARE instructions, requirements on ARE instructions were elicited from RE experts. These requirements basically state that an ARE instructions document should provide sound descriptions for an "ideal" elicitation sequence including predefined templates, input and output definitions (i.e., the issues to be elicited) as well as domain-specific questions. Furthermore, the instructions should make clear which stakeholders are to be involved when discussing a certain issue. Finally, the instructions should provide information when finished and provide support for handling typical pitfalls, especially with regard to technical or economic constraints.

Of course, we are aware that ARE instructions that fulfill these requirements are basically very rigid, and that it may be complicated to keep all of them in mind during real customer conversations. However, we do not expect the ARE instructions to be used straightforward. Lauesen [Lau02], for instance, stresses that an elicitation instruction *"must not be followed point by point, and that the requirements engineers should rather follow the interviewee. However, it is important not to get sidetracked."* Thus, we consider ARE instructions (even though they are algorithmically formulated) as an abstract process to follow, or even just as a mnemonic to advise requirements engineers about necessary elicitation activities and the constraints that must be considered.

4.3 ARE Instructions Template in Detail

This section explains how an ARE instructions document may reflect important information about a certain SPL and RE best practices. For this purpose, we first introduce the overall structure of the template before defining the elicitation strategy to be reflected in it. In a third step, single elicitation instructions that provide clear statements to requirements engineers on what to do will then be elaborated. Furthermore, hints are provided that give information about important aspects requirements engineers should be aware of during elicitation.

4.3.1 Basic Structure

The overall purpose of an ARE instructions document is to guide the requirements process, respectively the requirements elicitation, which is the focus of this thesis. Thus, the general structure of envisioned elicitation processes must be covered appropriately in an ARE instructions document.

As already shown in chapter 2, a requirements process basically consists of requirements phases in which several requirements activities are performed. The requirements activities within a requirements phase elaborate the requirements that are needed before a corresponding development phase can start. Each requirements activity addresses exactly one (relevant) issue and may consider the SPL specification or certain assumptions in order to adhere to the given SPL.

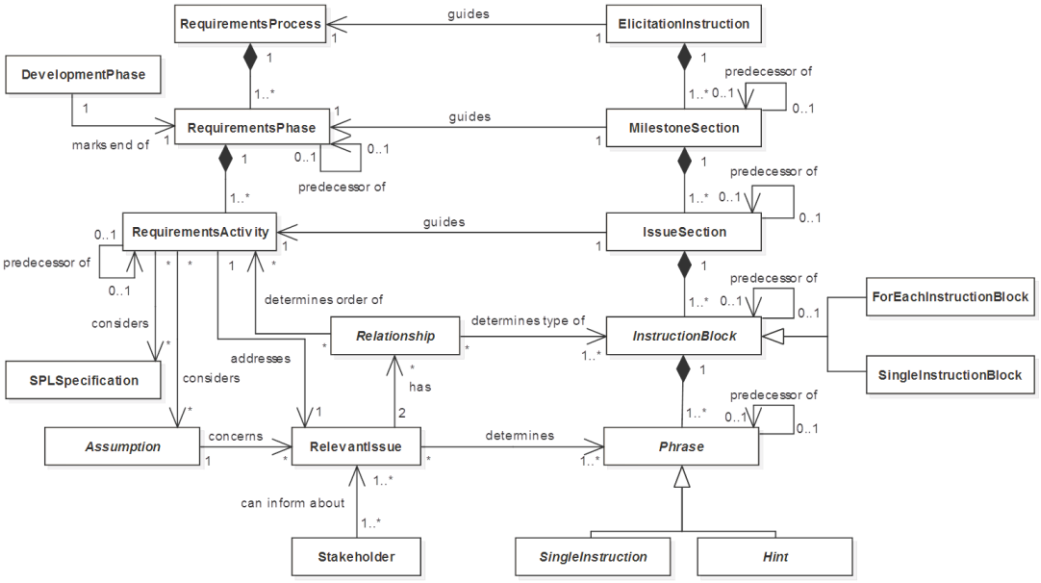


Figure 20. Structure and dependency of elicitation instructions

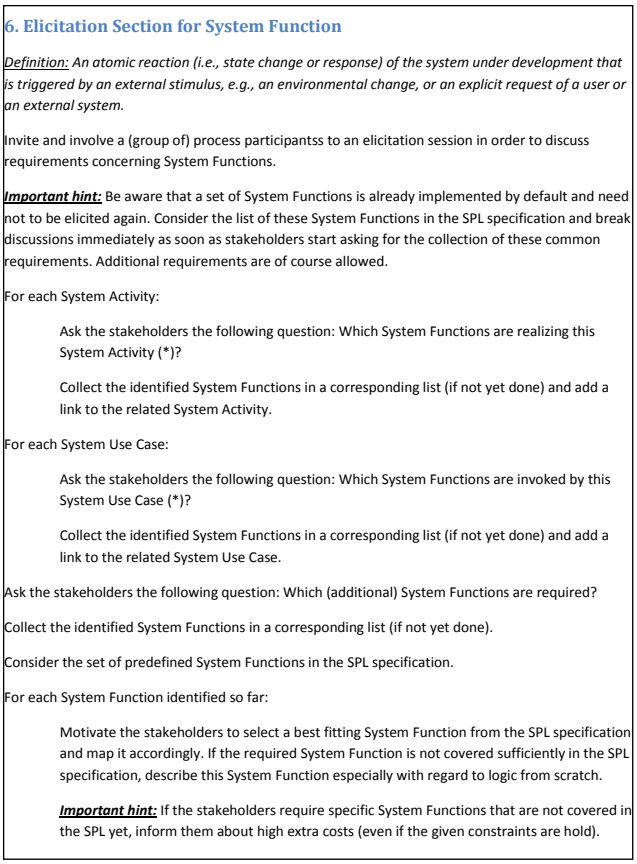
In order to provide appropriate support, the template of an ARE instructions document (see left part of Figure 20) therefore reflects this structure of a requirements process. Thus, for each requirements phase within a requirements process, an elicitation instruction (i.e., an ARE instructions document) should provide a corresponding milestone section. The purpose of a milestone section is to collect all instructions for the requirements activities that must be performed in the corresponding requirements phase. The idea behind this structuring is that requirements concerning different issues are typically needed at different points in time during subsequent development (see section 2.4.1). If, for instance, a phase "business analysis" is part of a requirements process, the corre-

sponding milestone section has to guide all requirements activities that are needed to elaborate business-relevant issues such as business goals, business objects, business rules, business processes, etc., as otherwise the related development phase cannot start. Each milestone section is further subdivided into issue sections concerning one specific issue (e.g., one section for business processes, one for business objects, …). Thus, a requirements activity always deals with the elicitation of requirements regarding one specific class of elements.

Within each issue section, concrete guidance on how to elicit and analyze requirements concerning a specific class of issues is then given. For this purpose, an issue section (see Figure 21 for an example) contains precise phrases (i.e., single instructions and hints), which are organized into so-called instruction blocks. While the phrases comprise concrete statements for the requirements engineers on what to do or what to consider, the instruction blocks group these phrases in order to align different sub-activities such as asking, describing, classifying, clarifying, etc. more logically.

Instruction blocks are either for-each instruction blocks or single instruction blocks. Whether a for-each or a single instruction block is chosen depends on the relationship an issue has to other issues, respectively the issues' singularity. For instance, the instruction "*Ask the stakeholders the following question: Which organizational units are performing this business process? (at least one)*" may be replicated several times, as there is typically not only one business process for which the performing units are to be identified. Thus, by using a for-each instruction block, this single instruction can be embedded into a loop such as *"For each business process identified before: …"*.

Through this hierarchical separation using milestone sections, issue sections, and instruction blocks, an ARE instructions document provides a modularized structure that allows making breaks. Breaks can be made, for instance, after all requirements concerning a certain issue class have been elicited (i.e., after each instruction section), or after all requirements for a certain milestone have been covered (i.e., after each milestone section).



6. Elicitation Section for System Function

Definition: *An atomic reaction (i.e., state change or response) of the system under development that is triggered by an external stimulus, e.g., an environmental change, or an explicit request of a user or an external system.*

Invite and involve a (group of) process participantss to an elicitation session in order to discuss requirements concerning System Functions.

Important hint: Be aware that a set of System Functions is already implemented by default and need not to be elicited again. Consider the list of these System Functions in the SPL specification and break discussions immediately as soon as stakeholders start asking for the collection of these common requirements. Additional requirements are of course allowed.

For each System Activity:

Ask the stakeholders the following question: Which System Functions are realizing this System Activity (*)?

Collect the identified System Functions in a corresponding list (if not yet done) and add a link to the related System Activity.

For each System Use Case:

Ask the stakeholders the following question: Which System Functions are invoked by this System Use Case (*)?

Collect the identified System Functions in a corresponding list (if not yet done) and add a link to the related System Use Case.

Ask the stakeholders the following question: Which (additional) System Functions are required?

Collect the identified System Functions in a corresponding list (if not yet done).

Consider the set of predefined System Functions in the SPL specification.

For each System Function identified so far:

Motivate the stakeholders to select a best fitting System Function from the SPL specification and map it accordingly. If the required System Function is not covered sufficiently in the SPL specification, describe this System Function especially with regard to logic from scratch.

Important hint: If the stakeholders require specific System Functions that are not covered in the SPL yet, inform them about high extra costs (even if the given constraints are hold).

Figure 21. Example of an issue section

4.3.2 Implemented Elicitation Strategy

As a hierarchical structure does not specify the order of sections, blocks, and phrases, this subsection introduces the elicitation strategy according to which different issues should be processed ideally during an ARE process. This proposal is based upon the concepts expressed in the conceptual ARE model (see chapter 2), as well as additional notions explained below.

The basic idea of this thesis regarding an elicitation strategy is to consider requirements processes as algorithms that prescribe a systematic and repeatable way on how the issues of interest should be processed within ARE. In this regard, especially the relationships between the relevant issues as well as the development phases prior to which certain issues must be discussed are important for determining the concrete way of elicitation. Based on the conceptual ARE model from chapter 2, we list a set of principles to be reflected in an ARE instructions document:

- Ask only for requirements concerning *Relevant Issues,* as all other issues do not influence any *Decision* during AE anyway.

- Consider the *Development Phases* and their relationships in order to determine the order of the *Requirements Phases*.

- Consider the *Relationships* between *Relevant Issues* in order to define a detailed order of the *Requirements Activities*.

- Consider the *Relationships* between *Relevant Issues* in order to see whether one requirement may influence another one.

- Involve appropriate *Stakeholders* as they can provide information about the *Relevant Issues*.

- Consider the existing *Assumptions* made with regard to the *Relevant Issues* in order to know what is feasible and what is not.

- Consider the *Explicitly Anticipated Requirements* already described in the *SPL Specification* in order to avoid re-eliciting everything from scratch.

From these principles, it becomes apparent that the order of milestone sections, issue sections, instruction blocks, and concrete phrases within an instruction block mainly depends on the relevant issues, respectively their relationships with other elements of the conceptual ARE model. In this regard, the order of milestones sections, for instance, can be determined very easily because only the simple sequential order of requirements phases must be reflected. This sequential order holds even true if a certain requirements phase is performed iteratively.

Rule 1 – Order of Milestone Sections

The milestone sections within an ARE instructions document occur in the same sequential order as the corresponding requirements phases in the ARE process. Thus, if requirements phase 1 is the predecessor of requirements phase 2, then the milestone section that guides requirements phase 1 is the predecessor of the milestone section that guides requirements phase 2.

Besides the order of milestone sections, the order of issue sections can also be determined in this way; namely by simply considering the order of the requirements activities.

However, determining the order of requirements activities (respectively the order in which certain issue classes have to be discussed) requires thorough consideration of the conceptual relationships among the relevant issues they address. This is much more difficult, as the order may depend on more than one relationship here. For instance, when discuss-

ing business activities, it must be considered that these activities have relationships to business processes, business objects, business roles, and business rules. Finding a meaningful order that avoids discussing one issue several times is therefore a challenging task. In order to cope with this problem, we have therefore developed the following algorithm (see rule 2). The basic idea of this algorithm is that all issues of a certain class that depend from issues of another class (e.g., via "contained in" relationships) should not be discussed before all these issues have been discussed also. Otherwise fly backs might be necessary, which could lead to a loss of control within the elicitation process. Thus, all independent issues should be addressed first, before the dependent issues are to be processed in a depth-first way. In this regard, depth-first means that the specializing and contained issues are addressed first, followed by required or influenced issues. The goal of this strategy is to minimize context switches. For instance, when discussing business activities, it makes sense to proceed with a classification (specialization) of these activities rather than with a discussion of the system functions required by them.

Rule 2 – Order of Requirements Activities (i.e., Issue Sections) within a certain Requirements Phase (i.e., Milestone Section)

1) Discuss in a random order all issues that do not have any relationship to another issue.

2) Discuss in a random order all issues that are not required by, not contained in, not influenced by, and not a specialization of another issue. If there is none, discuss at least those issues in a random order that are influenced by an already discussed issue, but have no further required / contained / influenced / specialization relationships.

3) Discuss all issues that are required by, contained in, influenced by, or a specialization of an already discussed issue, and that are neither required by, contained in, influenced by, nor a specialization of an issue that has not been discussed yet. If there is more than one, discuss them in the following suborder:

1) issues that specialize an already discussed issue

2) issues that are contained in an already discussed issue

3) issues that are required by an already discussed issue

4) issues that are influenced by an already discussed issue.

If there is more than one issue in each suborder, discuss them in the order in which the specialized / containing / requiring / influencing issue has appeared. Adapt the order continuously and repeat this procedure until all issues related to a certain milestone have been discussed.

By organizing an ARE instructions document according to these rules, it is easy to recognize when finished, respectively when a certain milestone

has been reached successfully (fulfills requirement R.S.4 of elicitation instructions shown in Appendix B, i.e., *"make clear when elicitation is finished").* In particular, when developing ARE instructions based upon these rules, it can be constructively assured that all requirements are available before the elicitation of related requirements starts.

Of course, there is a risk that the order of milestone sections may contravene the required order of issue sections. If, for instance, business objects should be discussed in an early requirements phase, but the business processes in which they are required are discussed in a late phase, a conflict exists that must be resolved. However, we will deal with appropriate conflict resolution during tailoring (see chapter 6) and not in this chapter.

Regarding the order of phrases and the embedding of instruction blocks within an issue section, a clear strategy has also been defined. Here, we basically propose two different principles:

Principle 1 – Consideration of Relationships

The requirements concerning a certain issue are elicited by considering their relationships to the already elicited requirements of another issue.

Principle 2 – Identification before Definition

Requirements concerning a certain issue are identified before being defined in detail.

The rationale for the first principle is that we assume that stakeholders can name requirements concerning a certain issue better when they consider the context of this issue. For instance, asking which role is responsible for a certain business activity will probably lead to a more reliable answer than when letting stakeholders enumerate all roles in isolation. The rational for the second principle is that we assume that it is more effective and efficient to let stakeholders' minds wander instead of interrupting them with another question.

Thus, each issue section should first contain the phrases that aim at identifying and collecting all requirements concerning the issue without defining them in detail. For instance, before the characteristics of the business processes to be supported are elaborated, an initial list of all these processes has to be developed first. At the beginning of each issue section, one or more instruction blocks should therefore be implemented, where each instruction block covers one (contained in or required by) relationship the issue class of interest has to another issue. For instance, a business object that is required by business processes and business activities would have two instruction blocks reflecting these relationships. In this regard, an issue's relationships and the corresponding instruction

blocks should be covered in the following order for applying the depth-first approach consistently.

Rule 3 – Order of Instruction Blocks that express Relationships

The instruction blocks concerning the related issues within an issue section occur in the same sequential order as the issue sections of the related issues in the ARE instructions document. Thus, if the issue section of related issue 1 is the (direct or indirect) predecessor of the issues section of related issue 2 in the ARE instructions document, the instruction block that concerns related issue 1 is the predecessor of the instruction block that concerns related issue 2.

Whether a for-each instruction block or a single instruction block is chosen depends on the cardinality, respectively the singularity, of the related issue class. In a normal case, a for-each instruction block is chosen to implement a relationship; a single instruction block is only used if the related issue is a singleton. Examples of a singleton issue are the system to be developed, the project in which the system is developed, or the physical environment of the system.

In addition to the instruction blocks covering the relationships, a further single instruction block is to be used if requirements concerning the issue class of interest may exist without having a relationship. An example is a user role that does not use the system for performing certain business activities, but that is responsible for administrative tasks, which might not be explicitly discussed. Furthermore, an additional for-each instruction block can also be introduced when a decomposition of collected issues is necessary to complete the identification of corresponding requirements.

The idea of identifying requirements concerning a certain issue through the iterative consideration of requirements concerning related issues allows achieving a high degree of completeness in a constructive manner. Thus, besides knowing whether all relevant issues have been discussed, requirements engineers can also get good indications of whether the corresponding requirements have been elicited completely (fulfills R.S.4, i.e., *"make clear when elicitation is finished"*). Of course, the identification of requirements through the consideration of related requirements implies that the considered set of related requirements is complete. This prerequisite can only be fulfilled if all issue sections dealing with the elicitation of requirements concerning a related issue are finished before another issue is discussed.

In order to finish each issue section, the requirements concerning the issue class of interest (relevant issue) identified and collected before must therefore be described, visualized, or reused. Thus, after all requirements concerning the issue class of interest have then been collected in the aforementioned instruction blocks, a single instruction block that may contain several hints or visualization instructions is now included in the

issue section. The purpose of this block is to inform about possible constraints that should be considered when describing each requirement in more detail. Each issue section then closes with an instruction block that contains individual instructions for describing, classifying, or selecting the requirements. Whether a for-each or single instruction block is chosen depends again on the singularity of the issue class of interest.

Regarding the selection and instantiation of concrete phrases within an issue section, the properties of the issue to be discussed, respectively the properties of its related issues, have to be taken into consideration. The most important properties in this regard are the status of an issue and the degree of freedom provided by the underlying SPL. While the former expresses whether and how many instances an issue class may have (normal = n, singleton = 1, abstract = 0), the latter expresses whether requirements concerning an issue class are already predefined in the SPL, respectively restricted by the SPL architecture or strategy. In Figure 21, for instance, the degree of freedom states that a couple of system functions are already covered in the SPL specification, but that additional, customer-specific system functions may be specified too. Hence, corresponding hints and single instructions that inform requirements engineers about this fact are included in the issue section.

In this regard, we define and apply the following rule with regard to the consideration of different artifacts based on the degree of freedom.

Rule 4 – Artifacts to Consider

1) If all requirements concerning an issue class are common within the SPL anyway, no consideration of SPL characteristics is necessary during ARE, of course.

2) If the requirements concerning an issue class are all variable (i.e., explicitly anticipated), only the SPL specification in which these requirements are described needs to be considered.

3) If all requirements concerning an issue class are not explicitly anticipated, i.e., if they are implicitly anticipated or even non-anticipated, the assumptions, if any, made by the flexibility classes have to be considered.

4) In all hybrid forms, assumptions and the SPL specification both need to be considered, for instance, if a set of requirements concerning an issue class is implemented as a commonality, while further requirements concerning this issue class may be added specifically due to certain flexibility classes,.

This rule is also depicted as a decision table (see Table 3). Hence, depending on whether requirements concerning a certain issue are explicitly anticipated (either as a common or as variable requirement), assumptions, or the SPL specification, or both have to be considered during the corresponding requirements activity. In this regard, we intentionally exclude the case that requirements should be elicited for an issue for which all related requirements are common anyway.

Requirements concerning issue are …			**Requirements activity has to consider…**	
common	*variable*	*specific*	*Assumptions*	*SPL specification*
y			(no activity necessary)	
	y			c
		y	c	
y	y			c
y		y	c	c
	y	y	c	c
y	y	y	c	c

y: requirements (will) exist, c: artifact must be considered

Table 3. Artifacts to be considered

In Appendix D, we present an algorithmic description for determining all internals of the issue sections according to the aforementioned explanations more clearly. This algorithm is based on the rules and principles mentioned above. In particular, this pseudo code explains (in a simplified manner) under which conditions (based on the attributes or relationships of a relevant issue), a certain single instruction or hint is incorporated into the instructions document. The refined implementation of this algorithm, which also takes technical aspects regarding the correct generation of ARE instructions into consideration, is shown in Appendix C.

In the next subsections, we will now introduce these single instructions and hints and explain the conditions under which they are displayed in an ARE instructions document.

4.3.3 Single Instructions

Based on the confirmed hypothetic requirements on elicitation instructions (see Appendix B) as well as the elements in our conceptual AE model (see chapter 2), we have identified a set of eight single elicitation instructions to be part of the ARE instructions template (see Figure 22). The overall purpose of these instructions is to support the requirements elicitation through the provision of predefined actions that are typically needed in this context.

The concrete text blocks for each instruction are based on the formulations used in an industrial elicitation instructions document developed some years ago at our institute (see research approach in section 4.1). We used these formulations as a basis because this document had enabled people with low RE experience to perform rather good elicitation merely by following the contained instructions. However, in order to fit the thesis context, slight modifications were needed in order to provide information that was not provided by the industrial document.

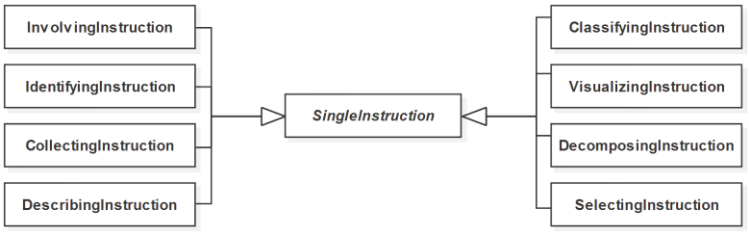


Figure 22. Taxonomy of single (elicitation) instructions

Below, we describe each single instruction including its purpose and the idea it implements in more detail.

4.3.3.1 Involving Instruction

The purpose of this instruction is to involve the stakeholders that are needed for a certain elicitation step (fulfills R.C.3, i.e., *"name the required stakeholders"*). This instruction is used for assuring that the right stakeholder group is available when requirements concerning a certain issue are discussed, as only these people are assumed to be able to provide the required information (see Figure 20). Indeed, the concrete stakeholders to be involved cannot be predefined upfront and must typically be identified together with a contact person of the customer organization in each AE project. However, the skills these stakeholders should have can be determined in advance based on the issue class to be discussed. If, for instance, the business processes are of interest, line managers, but probably not IT people, are an important stakeholder group to be involved.

Thus, the involving instruction is aligned with the issue class of interest (relevant issue), which is the only variation factor besides the stakeholder group. The resulting template for involving instructions then looks as follows:

Template 1

Invite and involve a (group of) <stakeholder group> to an elicitation session in order to discuss requirements concerning <issue class of interest>s.

Example: *"Invite and involve a (group of) line managers to an elicitation session in order to discuss requirements concerning the business processes."*

4.3.3.2 Identifying Instruction

The purpose of this instruction is to find out what a stakeholder actually wants or needs. As Davis et al. [DDH+06] have shown that interviews are probably the most effective way of elicitation, this instruction prompts requirements engineers to actively pose direct questions to the stakeholders. However, as this instruction just aims at identifying requirements without defining them in detail, only "which"-questions should be used here. Furthermore, *"in order to avoid prejudicing the stakeholders' answers, context-free questions should be used"* [LW00], i.e., information about a possible solution should not be part of an interview question. The identifying instruction is based on our notion of relevant issues and is therefore needed to fulfill requirement R.C.1 in Appendix B (i.e., *"name the issues to be discussed"*).

Thus, for each relevant issue, at least one identifying instruction exists, as this instruction aims at identifying all requirements that concern a certain issue (see Figure 20). In this regard, the instruction implements our elicitation strategy to identify requirements through the consideration of (already identified) requirements whose issues have a relationship to the issue currently being discussed. For instance, when a set of business processes has already been identified, the next identifying instruction could ask for the organizational units that perform these business processes. The assumption behind this strategy is that stakeholders can name requirements concerning a certain issue better when they consider the context of this issue (i.e., the conceptual relationships). In this example, clear orientation on the business processes may help to elicit the involved organizational units more easily, and maybe also more completely than pure enumeration of all possible units in a greenfield manner. Hence, as the relationships between issues are explicitly considered by this instruction, requirement R.C.12 (i.e., *"inform about dependencies"*) regarding elicitation instructions is also fulfilled implicitly.

The variation factors that determine the actual formulation of an identifying instruction include the issue class of interest (relevant issue) for which requirements are to be identified, other relevant issues related to this issue class, and their relationships including their cardinality. The resulting templates for identifying instructions look as follows:

Template 1

Ask the stakeholders the following question: Which <issue class of interest>s are <relationship> this <referencing issue class> (<cardinality>)?

Template 2

Ask the stakeholders the following question: Which (additional) <issue class of interest>s are required?

The first template should be used when an issue is in a relationship with another issue discussed before. The second template is (additionally) chosen when a requirement concerning the issue class of interest can also exist without having a relationship to a requirement that concerns a related issue. According to the elicitation patterns of Scheinholtz and Wilmont [SW11], we use moderately open, neutral, primary interview questions when following this instruction template. This means that the posed question allows considerable freedom with only a certain amount of restrictions, while no overt direction is given by the requirements engineer.

Example (T1): "*Ask the stakeholders the following question: Which organizational units are performing this business process? (at least one)*"

Example (T2): *"Ask the stakeholders the following question: Which (additional) organizational units are required?"*

In this concrete example, only template 1 makes sense. As there is typically not only one business process for which the related units can be identified in this case, this instruction must be replicated several times as part of a for-each instruction block as introduced above.

4.3.3.3 Collecting Instruction

The purpose of this instruction is to collect all identified requirements in an enumerative manner. Collecting instructions are needed for quick and temporary documentation (mainly in terms of bullet lists or short notes) in order to handle the mass of gathered information efficiently during an elicitation session. However, collecting instructions just focus on the enumeration of requirements in terms of keywords without describing any details. In the organizational unit example above, for instance, collecting instructions would ask for an enumeration of the names of all units to be supported without noting details such as responsibilities, etc. This notion to focus only on enumeration reflects our elicitation strategy that the details for each requirement should not be defined before a quite stable set of requirements has been achieved. The reason is that interrupting inquiries regarding certain details can be reduced in this way, which is especially worthwhile when stakeholders are currently going to let their minds wander. Furthermore, we experienced that a detailed description provided only once and late during an elicitation session is a fruitful means to avoid the complete definition of requirements that are later discarded anyway. One step during collecting, which is important, is the recording of relationships to other requirements, as otherwise late reconstruction might cause costly inquiries.

The variation factors for the collecting instruction are similar to the variation factors of the identifying instruction; namely the issue class of interest for which requirements are to be collected (relevant issue) and the is-

sues related to this issue. The resulting templates for collecting instructions therefore look as follows:

Template 1

Collect the identified <issue class of interest>s in a corresponding list (if not yet done) and add a link to the related <referencing issue class>.

Template 2

Collect the identified <issue class of interest>s in a corresponding list (if not yet done).

The second template is chosen when a requirement is collected that has no relationship to another requirement, or for which the relationship is just a specialize- or influence-relationship (see Figure 13), as we consider these relationships important for elaborating requirements rather than for understanding. In contrast, the first template is recommended in all other cases.

Example (T1): *"Collect the identified organizational units in a corresponding list (if not yet done) and add a link to the related business process".*

Example (T2): *"Collect the identified organizational units in a corresponding list (if not yet done)."*

4.3.3.4 Describing Instruction

While identifying and collecting instructions focus on gathering requirements without defining any details, the purpose of describing instructions is exactly to elicit and record this information. Describing instructions should therefore help requirements engineers to motivate the stakeholders to provide detailed information about a requirement according to the attributes of the issue the requirement is concerned with. This instruction therefore fulfills the requirement on ARE instructions to inform requirements engineers about all the concrete details to be elicited (addresses R.C.4, i.e., *"name the details to be elicited"*).

In the organizational unit example, for instance, describing instructions prompt requirements engineers to elaborate all details of interest such as the responsibilities of the organizational unit, its size, etc. However, as already mentioned, the elicitation of such details makes only sense if a stable set of requirements has already been achieved.

The variation factors that determine the actual formulation of a describing instruction are again similar to the previous ones and include the issue class of interest (relevant issue) for which requirements are to be described, the attributes of this issue, the issues related to this issue, and

the corresponding relationships. The resulting templates for describing instructions look as follows:

Template 1

Ask the stakeholders the following question: Could you please describe this <issue class of interest> especially with regard to <attributes of issue class of interest>?

Template 2

Ask the stakeholders the following question: Could you please describe the <issue class of interest> <relationship> this <referencing issue class> especially with regard to <attributes of issue class of interest>?

The first template is chosen when all requirements concerning a certain issue have already been collected and should now be described in more detail. Similar to identifying instructions, an iteration over all requirements is necessary here. In contrast, the second template is recommended when no separate collection has taken place, as in a case where only one requirement is related to another one. If, for instance, a business process could only be performed by exactly one organizational unit, requirements engineers can be guided to directly ask for the details of this unit according to template 2. Thus, depending on this cardinality, respectively on the singularity of an issue, the aforementioned example can lead to two different describing instructions.

Example (T1): "*Ask the stakeholders the following question: Could you please describe this organizational unit especially with regard to responsibilities, size, …*".

Example (T2): "*Ask the stakeholders the following question: Could you please describe the organizational unit performing this business process especially with regard to responsibilities, size, …*".

Apparently, the point of reference must fit in both cases in order to choose the right instruction template. We describe the rules for integrating each single instruction in the overall elicitation instruction in Appendix D.

4.3.3.5 Classifying Instruction

The purpose of this instruction is to support the classification of requirements into more specific groups. Classifying instructions are needed when requirements concerning an abstract (super) issue (e.g., business activity) have been collected. The rationale for this instruction is based upon the observation that requirements concerning different issues are sometimes identified and collected in an integrated way, but need to be

separated before they can be described in detail. When identifying and collecting the (abstract) business activities within a business process, for instance, no distinction is typically made as to whether these activities are pure human activities, system-supported activities, or system-automated activities. However, as in software development only system-supported and system-automated activities are typically of interest, a corresponding distinction must be made before proceeding further.

The variation factors for classifying instructions include the (super) issue class of interest (relevant issue) as well as the list of issues that specialize the former one. The resulting template for classifying instructions looks as follows:

Template 1

Discuss with the stakeholders if this <issue class of interest> is a <list of specialized issues> and categorize it accordingly.

Example: *"Discuss with the stakeholders if this business activity is a human activity, a system-supported activity, or a system-automated activity and categorize it accordingly".*

4.3.3.6 Visualizing Instruction

In elicitation sessions, requirements are often visualized using certain notations because visualization helps to clarify details or relationships much better than just spoken words (see element "notation" in Figure 13). The visualizing instruction therefore aims at motivating requirements engineers to use graphical representations during elicitation sessions.

The variation factors for visualization instructions include the notation to be used and the issue class of interest (relevant issue) whose requirements are to be visualized. Hence, the resulting templates look as follows:

Template 1

Draw a <specific notation> to clarify the details of this <issue class of interest>.

Template 2

Draw a <specific notation> to clarify the interplay between all <issue class of interest>s.

The first template should be used when details about a single requirement are to be visualized. A prominent example is the visualization of a business process (flow) by using an event-driven process chain. The second template should be used when the interplay between all require-

ments concerning a certain issue is of interest. A prominent example here is an organization chart in which the dependency between all organizational units is reflected.

Example (T1): *"Draw an event-driven process chain to clarify the details of this business process."*

Example (T2): *"Draw an organization chart to clarify the interplay between all organizational units."*

4.3.3.7 Decomposing Instruction

The purpose of this instruction is to prompt requirements engineers to decompose hierarchical structures in order to elaborate the included requirements. The rationale for decomposing instruction is based upon the fact that the issues, with which requirements are concerned are sometimes too coarse-grained to provide sufficient information for development. This instruction is therefore needed for dealing with issues that have contain-relationships with themselves. A prominent example is an organizational unit, which can be recursively decomposed into other organizational units.

The variation factor for decomposing instructions only includes the issue class of interest (relevant issue). Hence, the resulting template looks as follows:

Template 1

Decompose the hierarchy of this <issue class of interest> until no further decomposition is possible. Collect the identified <issue class of interest>s in a corresponding list (if not yet done) and add a link to the parent <issue class of interest>.

Example: *"Decompose the hierarchy of this organizational unit until no further decomposition is possible. Collect the identified organizational units in a corresponding list (if not yet done) and add a link to the parent organizational unit."*

4.3.3.8 Selecting Instruction

The purpose of the selection instruction is to foster the reuse of explicitly anticipated requirements wherever possible. This instruction prompts requirements engineers to consider the SPL specification and variability model (see Figure 20), and to motivate the stakeholders to choose explicitly anticipated requirements instead of allowing them to state requirements from scratch. In particular, when configuration is the preferred strategy for AE, selecting instructions are necessary. In cases

where several features have already been incorporated in the reuse asset base, selecting available implementations instead of reinventing them also makes sense. An example is the selection of certain adapters to databases, ERP systems, web servers, etc., for which a repeated definition of requirements is costly.

However, a systematic identification and collection of requirements using the aforementioned instructions is also needed in such a strongly reuse-oriented case. If, for instance, a set of existing functions were to be presented to stakeholders without investigating which tasks they will perform with the system to be derived from the SPL, it is quite likely that they would choose functions they will never use and vice versa. Thus, our approach tries to assure that the elaborated requirements reflect what the stakeholders actually need, and not only what they believe they need.

The variation factors for selecting instructions include the issue class of interest, and the attributes of this relevant issue. The resulting templates for selection instructions look as follows:

Template 1

Let the stakeholders select the best fitting <issue class of interest> from the SPL specification and map it accordingly. Reject all elicited <issue class of interest>s that cannot be mapped.

Template 2

Motivate the stakeholders to select the best fitting <issue class of interest> from the SPL specification and map it accordingly. If the required <issue class of interest> is not covered sufficiently in the SPL specification, describe this <issue class of interest> especially with regard *to <attributes of issue class of interest> from scratch.*

Template 1 is to be used when only explicitly anticipated requirements concerning a certain issue are allowed to be stated. In contrast, template 2 should be used when explicit reuse of SPL requirements is mandatory wherever possible, but additional requirements are also welcome. The adapters above are a good example of this latter case. When appropriate adapters already exist, it does not make sense to redefine them. However, if additional adapters are needed and also possible due to the architecture's flexibility, the elicitation of corresponding requirements must be supported, of course.

Example (T1): *"Let the stakeholders select the best fitting adapter from the SPL specification and map it accordingly. Reject all elicited adapters that cannot be mapped."*

Example (T2): "*Motivate the stakeholders to select the best fitting adapter from the SPL specification and map it accordingly. If the required adapter is not covered sufficiently in the SPL specification, described this adapter especially with regard to … from scratch.*"

4.3.4 Hints

While the aforementioned instructions support the requirements elicitation through the predefined description of single actions that are typically needed, the hints contain information that requirements engineers should be aware of. This is especially needed to avoid the elicitation of non-fitting, superfluous, or missing requirements and thus serves to accelerate the alignment of customer requirements with SPL characteristics. As an example, knowledge about the commonalities in the reuse asset base is important to avoid elicitation of unnecessary requirements.

The single instructions introduced in the previous section are widely independent of the development context in which they are used. Even though they have a strong focus on the relevant issues, they are not specifically focused on a given SPL, except for the selecting instruction. In particular, information about the feasibility of requirements is not reflected in the single instructions yet. Thus, an explicit consideration of the SPL characteristics that goes beyond the SPL specification is necessary to avoid violating the product line architecture.

As already introduced and also required by the requirements on ARE instructions (see Appendix B) ARE instructions must therefore provide information about restricted issues (R.C.8), the corresponding assumptions (R.C.9), and the technical aspects that basically influence feasibility (R.C.10). Without this information, the requirements engineers cannot systematically assure that the capabilities of an SPL are actually met.

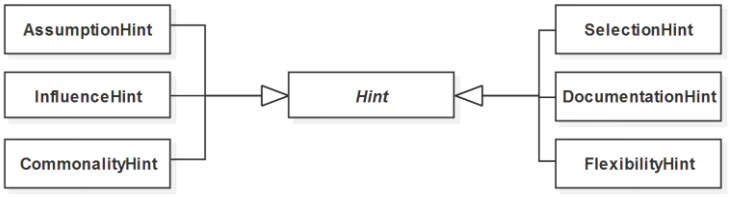


Figure 23. Taxonomy of (elicitation) hints

In this subsection, we therefore explain how additional knowledge about an SPL can be incorporated into and represented in the elicitation instructions in order to fulfill the requirements on ARE instructions to be specific for a certain development or project context (R.N.4) and to reflect technical constraints in a non-technical way (R.N.11). We address this with the notion of hints (see Figure 23), which complement single

instructions. Similar to single instructions, these hints have also been derived from the properties and relationships of relevant issues according to the conceptual ARE model as well as from the elicited requirements on ARE instructions. Below, we describe each hint including its purpose and the idea it implements in more detail.

4.3.4.1 Assumption Hint

The assumption hint is probably the most important hint to reflect SPL characteristics and constraints in ARE instructions without the need to specify all possible requirements in an explicit manner. Assumption hints describe the assumptions the product line architecture makes about a certain issue in order to exploit the supported flexibility (see Figure 20).

The purpose of this hint is therefore the description of the constraints a requirement must meet so that it can be assessed as being realizable by requirements engineers during elicitation already. Hence, this hint fulfills the RE experts' requirements R.C.5 (i.e., *"name the criteria against which requirements have to be checked"*), R.C.8 (i.e., *"inform whether requirements are restricted by architecture"*), and R.C.9 (i.e., *"name the properties that a requirement must fulfill"*) from Appendix B. For instance, when specific documents should be automatically processed by a system, assumptions could constrain the number of pages or the number of words these documents may contain. As assumption hints describe limitations to the "internals" of a requirement and not just its name, they must be combined with describing instructions.

The variation factors of this hint include the issue class of interest (relevant issue) on which assumptions are defined as well as the constraints that result from these assumptions. The resulting template for assumption hints looks as follows:

Template 1

Important hint: Be aware that there are constraints defined for <issue class of interest> requirements. Hence, the <issue class of interest>s stakeholders may ask for are restricted as follows: <constraints>. If the stakeholders require something that contravenes these constraints, inform them about possible (significant) extra costs and tell them that an expert check must be done before you can accept this requirement.

Template 2

Important hint: Be aware that there are constraints defined for <issue class of interest> requirements that are hard! Hence, the <issue class of interest>s stakeholders may ask for are restricted as follows: <constraints>. If the stakeholders require something that contravenes these constraints, inform them that this is not possible technically.

It is evident that the first template is used for soft assumptions, while the second template is used when an assumption is hard and therefore not negotiable.

Example (T1): "*Important hint: Be aware that there are constraints defined for document requirements. Hence, the documents stakeholders may ask for are restricted as follows: pages<10, words<10000. If the stakeholders require something that contravenes these constraints, inform them about possible (significant) extra costs and that an expert check must be done before you can accept this requirement.*"

Example (T2): "*Important hint: Be aware that there are constraints defined for document requirements. Hence, the documents stakeholders may ask for are restricted as follows: pages<10, words<10000. If the stakeholders require something that contravenes these constraints, inform them that this technically not possible.*"

In this context, however, it must be noted that constraints can not only restrict the direct properties of an issue. Constraints can also describe that only a limited set of existing functions is to be used, for instances, when defining a business activity.

Additional example: "*Important hint: Be aware that there are constraints defined for the business activity requirements. Hence, the business activities, stakeholders may ask for, are restricted as follows: only functions provided by services described in the SPL specification may be invoked. ….*"

4.3.4.2 Influence Hint

The purpose of this hint is to inform about influence-relationships that exist between different issues, and that may also apply to the corresponding requirements (addresses R.C.12, i.e., *"inform about dependencies"*). This hint is needed for considering issues that are not related in a "hard" sense, i.e., not via contain-, require- or specialize-relationships (see relationships in Figure 13). An example here is a project goal that may influence certain quality characteristics of a system. An influence hint should therefore provide information about a possible impact another requirement might have on a requirement to be elicited.

The variation factors include the issue class of interest (relevant issue) for which requirements should be elicited and a list of issues that have an influence-relationship to this issue. The resulting templates for influence hints look as follows:

Template 1

Important hint: Consider especially the <influencing issue>s when determining the <issue class of interest>s.

Template 2
Important hint: Consider especially the < influencing issue>s when classifying the <issue class of interest>s.

It is evident that the first template is used together with identifying or describing instructions, while the second template complements classifying instructions, if necessary.

Example (T1): "*Ask the stakeholders the following question: Which quality characteristics are required for the system function? (at least one). Important hint: Consider especially the project goals when determining the quality characteristics."*

Example (T2): "*Discuss with the stakeholders if this business activity is a human activity, system-supported activity, or system-automated activity and categorize it accordingly. Important hint: Consider especially the project goals when classifying the business activities."*

4.3.4.3 Commonality Hint

The purpose of the commonality hint is to provide information about requirements that are implemented by default anyway in order to proactively avoid unnecessary elicitations (addresses R.C.7, i.e., *"name the requirements that are implemented by default"*). In contrast to the avoidance of an entire elicitation step based upon a classification of an issue as "not being relevant", the commonality hint is used for all relevant issues for which a set of common requirements is already described in the SPL specification (see Figure 20), but which may also include additional and still unknown requirements. Commonality hints are therefore typically combined with collecting instructions.

The variation factor for this hint is only the issue class of interest (relevant issue). The resulting template for commonality hints looks as follows:

Template 1

Important hint: Be aware that a set of <issue class of interest>s is already implemented by default and need not be elicited again. Consider the list of these <issue class of interest>s in the SPL specification and break off discussions immediately as soon as stakeholders start asking for the collection of these common requirements. Additional requirements are, of course, allowed.

Example: *"Collect the identified adapters in a corresponding list (if not yet done) and add a link to the related partner systems. Important hint: Be aware that a set of adapters is already implemented by default and need not be elicited again. Consider the list of these adapters in the SPL specification and break off discussions immediately as soon as stakeholders start asking for the collection of these common requirements. Additional requirements are, of course, allowed."*

4.3.4.4 Selection Hint

The purpose of the selection hint is also to support SPL alignment by considering an SPL specification during a certain requirements activity (see Figure 20). However, in contrast to assumption hints, selection hints directly aim at considering predefined requirements in the SPL specification and are therefore to be used together with selection instructions.

The variation factor of this hint includes the issue class of interest (relevant issue) to which existing requirements in the SPL specification concern. The resulting template for selection hints looks as follows:

Template 1

Important hint: Consider the set of existing <issue class of interest>s in the SPL specification.

Example: *"Important hint: Consider the set of existing adapters in the SPL specification."*

4.3.4.5 Flexibility Hint

The purpose of the flexibility hint is to provide information about possible extra costs when stakeholders require specific extensions or modifications (see flexibility classes in Figure 11) even though reuse candidates are already there. Flexibility hints are therefore used together with some selection instructions.

The variation factor of this hint includes the issue class of interest (relevant issue) to which existing requirements in the SPL specification concern as well as the amount of basic costs when exploiting the desired flexibility. The resulting template for selection hints looks as follows:

Template 1

Important hint: If the stakeholders require specific <issue class of interest>s that are not covered in the SPL yet, inform them about <amount> extra costs even if the mentioned assumptions are kept.

Example: *"Important hint: If the stakeholders require specific adapters that are not covered in the SPL yet, inform them about high extra costs even if the mentioned assumptions are kept."*

4.3.4.6 Documentation Hint

As introduced in chapter 2, there are relevant issues that are actually relevant for development, and those that are only implicitly relevant for the elaboration of the former. The purpose of documentation hints is to inform requirements engineers which requirements are worth to spend effort on the description of corresponding details. If for instance, human activities in a business process are only important for the identification of system functions, but not needed to make any decision during development, a complete description of the corresponding requirements is unnecessary.

The variation factor for a documentation hint is therefore the name of the (only implicitly) relevant issue class of interest. The resulting template for documentation hints looks as follows:

Template 1

Important hint: It is not necessary to elicit or describe details about <issue class of interest>s.

Example: *"Important hint: It is not necessary to elicit or describe details about human activities"*.

4.4 Summary

In order to make AE requirements engineers aware of the capabilities and constraints of a given SPL as well as about RE best practices, corresponding process and product knowledge must be provided during elicitation. Unfortunately, existing work that deals with requirements elicitation still depends on the persons who carry them out rather than on the selected elicitation techniques (see related work in section 3.4).

In this section, we have therefore motivated and introduced a template for ARE instructions that prescribes the structure and content such instructions should have in order to support requirements engineers better. This template comprises eight types of instructions as well as six types of hints in order to provide repeatable guidance on what to do and what to consider during requirements elicitation in AE projects.

The template has been derived from our conceptual ARE model introduced in chapter 2 and fulfills a set of requirements that were (addition-

ally) stated by RE experts. With regard to these requirements (see Appendix B), the template provides clear procedural guidance that allows achieving a high degree of completeness and reproducibility. For this purpose, the template provides sound descriptions for an "ideal" elicitation sequence and makes clear which stakeholders are to be involved at which point in time. Finally, it provides information when finished on different levels of abstraction and support for negotiation through the provision of explicit knowledge about SPL characteristics.

With regard to the actual elaboration of requirements, the template implements an algorithmic elicitation strategy according to which all requirements concerning a certain issue are identified by covering the conceptual relationships this issue has to other issues. This step reflects the idea that requirements are likely much easier to identify if they are considered in their context instead of letting stakeholder enumerate them in isolation. Then, as soon as all requirements have been identified in this way, the details of each requirement are elicited according to the attributes of the underlying issue. If applicable, reusable requirements from the SPL specification, or existing assumptions are also considered in this step. Finally, the relationships of the currently discussed issues to other issues, which have not been discussed yet, are then used to initiate the recursive identification of requirements. Through this systematic iteration, an algorithmic guidance is provided for requirements engineers. However, even though the ARE instructions according to this template recommend a detailed process, requirements engineers are free to follow them. Thus, we consider the instructions rather as an abstract process.

In contrast to existing AE approaches, our strategy aims at aligning requirements with SPL characteristics immediately during elicitation. This has the advantage that the fit between customer requirements and SPL characteristics can be increased constructively. Nevertheless, ARE instructions according to this template might not be considered as an alternative to existing AE approaches, but rather as a complement. In particular, an ARE instructions document does not replace the use of predefined, variable requirements or corresponding variability-, respectively decision, models. In contrast, wherever such artifacts exist, the ARE instructions document integrates them via a selecting instruction. Thus, the ARE instructions aim at guiding the core elicitation process, but allow embedding more specific approaches whenever needed. Furthermore, the involvement of SPL experts is still possible when using ARE instructions according to the introduced template. The difference to existing work is that requirements engineers are informed under which conditions these experts have to be contacted. Thus, expert involvement can be reduced.

However, ARE instructions according to this template must be developed for each individual SPL in order to provide an actual benefit in AE. Chapter 6 therefore deals with the question of how ARE instructions can be derived from an SPL in a systematic and repeatable manner.

5 An Issue Model for Information Systems

"Only a fool believes to learn from his experiences.
I prefer to learn from the experiences of others to avoid
own mistakes from the outset."
Otto von Bismarck

As already described in the solution idea (see section 1.3.2), the overall purpose of this thesis approach is to tailor ARE instructions based on an SPL's reuse asset base, the intended development strategy, and RE best practices that are suitable in the addressed domain.

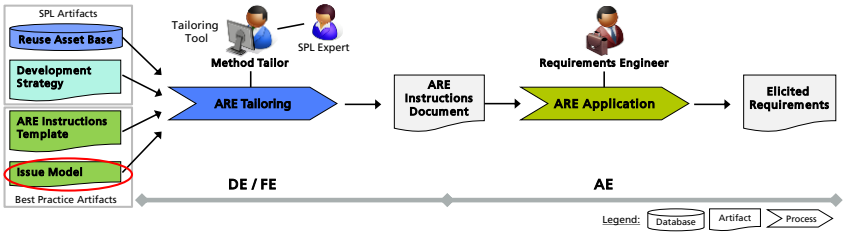


Figure 24. Issue model within this approach

To support this aim, this chapter introduces an issue model for RE in the IS area, as this system class is a representative example for flexible SPLs. The purpose of this chapter is to provide a list of topics to be discussed during elicitation there. Hence, besides the aforementioned template, the resulting model then acts as another RE best practices consolidation that forms a fourth input for the tailoring approach (see Figure 24).

However, similar to other reference models, this issue model does not claim to be perfect in every context either. It addresses typical settings when an IS should be introduced, but needs to be tailored to the specific development context by using the tailoring approach shown in the next chapter.

5.1 Research Approach

The development of the issue model described in this chapter was quite similar to that of the conceptual ARE model (see chapter 2). Thus, in a first step, existing work was consolidated (step 1). For this purpose, we analyzed the most popular specification standards IEEE 830-1998 [IEEE98a], IEEE 1233-1998 [IEEE98b], IEEE 1362-1998 [IEEE98c], and the Volere Template [RR99] and extracted all issues (respectively issue classes) mentioned in them by considering the nouns in the tables of contents. Furthermore, a couple of industrial requirements specifications from Fraunhofer IESE's customers, as well as the following methodological frameworks were also considered in order to enhance and challenge the initial set of issue classes: ARIS House [Sch01], Zachman Framework [Hay03], and Rational Unified Process [Kru00]. The reason for choosing exactly these approaches was that they are widely accepted (and not just academic) frameworks for developing IS. However, in contrast to the specification standards, issue classes were extracted here by using the nouns of the conceptual framework elements (e.g., business process). In the same way, the TORE framework [PK04] [ADE+09] was analyzed. The reason for using this approach developed at our institute was to assure good integration with our previous work.

In the more than 190 issue classes obtained from these resources, redundancies and synonyms were eliminated by using the open card sorting technique [UG12] based on the definitions of the issue classes. Furthermore, classes that were already part of our conceptual ARE model (e.g., goals, stakeholders, etc.) or that did not classify issues according to our definition were sorted out. Thus, we deleted motivations, problems, background, justifications, goals, visions, expected impacts and effects, project-related issues, support and accompanying services, and aspects of the solution space that are typically also contained in the aforementioned standards. The reason was that these "issues" are not needed for elaborating the required capabilities of an IS. Based on RE textbooks such as [Rup07] as well as on experience gathered in previous research, the relationships between the 29 remaining issue classes (see Appendix K for the initial version of the list) were then determined. For this purpose, we considered all issue class pairs one by one and checked whether there was a contain-, require-, specialize-, or influence-relationship among them.

Finally, in order to logically arrange the issue classes, we assigned each issue class to one of the four circles of Alexander's onion model [Alex05]. The reason for using this model as a classification scheme was that its circles are suitable means for putting issues in the right place within the context of an IS, even though this model was initially only proposed for classifying stakeholders. To make this assignment, we used the definition of each circle as proposed by Alexander and decided for each issue class whether the corresponding issues are part of the system to be devel-

oped, part of its direct usage environment, part of the overall business to be support, or part of the wider business environment. As a result, a first version of the issue model was developed.

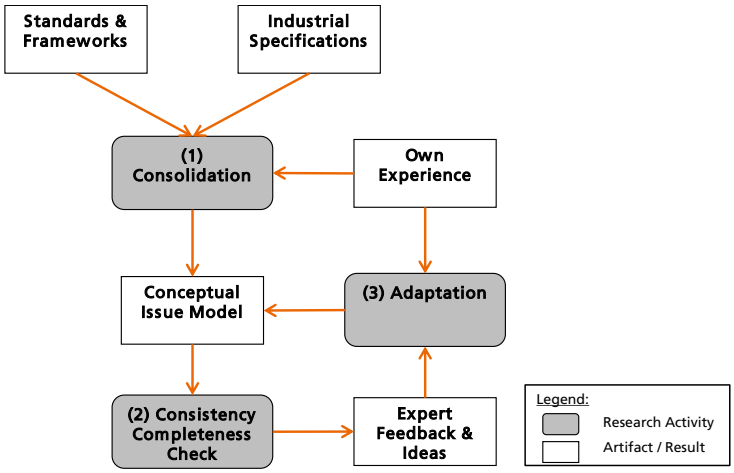


Figure 25. Research approach for issue model

In a couple of subsequent iterations, this model was then checked for completeness and consistency (step 2). This was done in two different ways. First, the model was discussed with four experts from the RE group at Fraunhofer IESE. Their feedback and ideas, which were partially based on a consideration of the V-Model XT [BI12] and the Fraunhofer IESE's SOE Model [ANT10], were incorporated into an improved version of the model. Second, completeness and consistency was checked by assessing concrete ARE instructions documents generated by our tailoring tool (see chapter 6) based on an untailored version of the issue model. Based on these instructions, the involved experts could easily check whether the reflected issue classes and relationships actually covered a realistic elicitation scenario when defining an IS. Thus, besides direct feedback on the model, ideas and observations that emerged during the assessment of the generated ARE instructions were also iteratively incorporated into an adapted version of the issue model (step 3).

Similar to the development of the conceptual ARE model, this model was therefore also developed in several cycles (see Figure 25). These iterations assured that our work converged towards a stable model in the last month of thesis research.

5.2 Model Overview

From a model theory point of view, the issue model bridges the gap between the element "issue" introduced in the conceptual ARE model (see chapter 2), and concrete real-world elements (e.g., "a business travel process") with which an AE project deals. Thus, the issue model contains a set of classes for whose instances requirements typically need to be elicited in an IS project. The reason for addressing issue classes and not concrete issues is that such a model would require a comprehensive domain analysis (and anticipation), which can never be complete, at least not in the context of flexible SPLs (see chapter 1).

Hence, to illustrate the contribution of the issue model, Figure 26 shows an example of how the meta-class "Issue", concrete issue classes such as "business process" and "business object", and corresponding issues (i.e., actual elements of a system or its environment) are arranged within the MOF stack [OMG12]. According to this figure, the issue model is logically assigned to the class level (M1) and therefore comprises important issue classes and their conceptual relationships that might be important in an IS project. This idea is based on the observation that many commonalities exist in RE for IS, even though *"there is no best way to develop enterprise systems"* yet [Gul04].

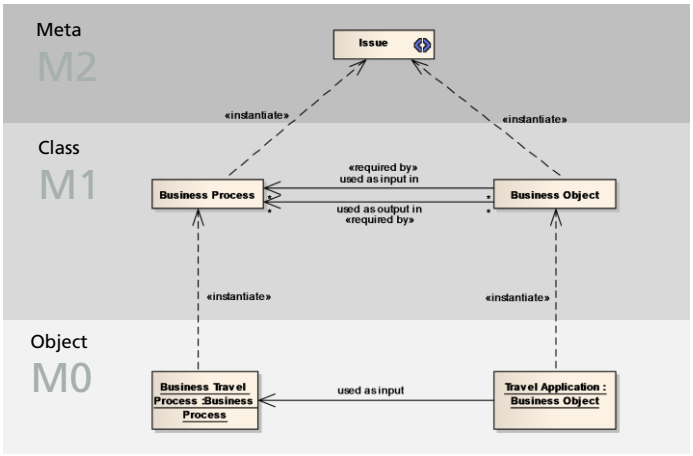


Figure 26. Example of "Issue", issue classes, and issues

In particular, the RE literature (see, for instance, [Rup07]) as well as state of the art development methodologies like the Rational Unified Process [Kru00] have proposed quite similar processes for RE in this area. In particular, the incremental-iterative approach as depicted in the spiral model of Sommerville [Som04], the onion model of Alexander [Ale05], or the idea of separating domain- and product requirements by Lauesen [Lau02], are three concepts that are implicitly or explicitly part of many

requirements approaches today. Especially the latter notion is important because it has been recognized that even systems that fulfill their (product-level) requirements often do not fully satisfy their stakeholders if their goals and tasks have not been understood sufficiently [Lau02]. Thus, the consideration of business issues is nowadays widely accepted and practiced in RE for IS [ST05] [GD07], as knowing the business context is a prerequisite to develop systems that address the actual business needs [GD07].

However, a conceptual foundation is still rare in this regard. Existing approaches are rather informal and often lack a conceptual foundation. Nevertheless, the idea of using conceptual models to clarify aspects of the world to be addressed by an engineering method is not new and is the state of the art in method tailoring [JJM09]. However, to our knowledge, no model exists yet that provides a foundation like the issue model described in this chapter.

5.3 Model View in Detail

This section introduces the elements of the reference model and explains the impact of this model on a meaningful elicitation sequence when following the elicitation strategy as proposed in section 4.3.2. To introduce this model in a stepwise manner, we use the views according to Alexander's onion circles [Ale05] (see Figure 27), which allow a thematic scoping of the issue classes to be considered in different elicitation phases.

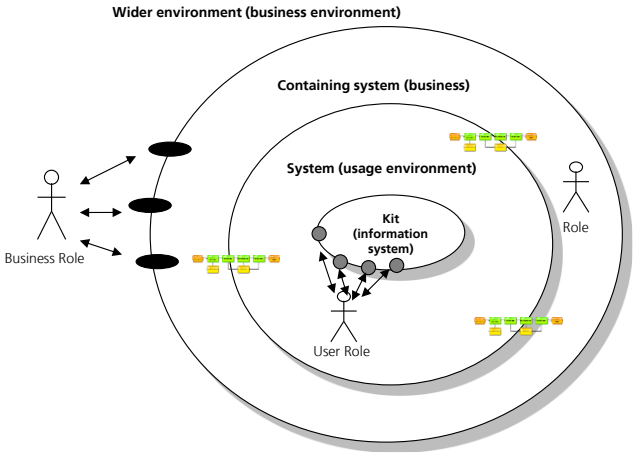


Figure 27. The onion model according to [Ale05]

In this regard, we start with the wider (business) environment view, followed by the containing (business) system view, the system view (i.e.,

the IS in its actual usage environment), and finally the kit view, in which system internals are discussed.

5.3.1 The Wider Environment

The wider environment view reflects the environment of an organization that should be supported by an IS, and therefore contains the issues that are necessary for scoping the boundaries of a project.

Based on our aforementioned research approach, we have elaborated the issue classes that are relevant in the wider environment (see Figure 53). The root of this view is a project in which an IS should be developed or introduced. In such a project, at least one business area (e.g., travel management) for which this system may be relevant is typically considered.

In this regard, the provided business services of the addressed business areas must be known in order to understand what the actual business is about. Business services mainly result from the need of an organization to satisfy external stimuli such as a customer request or an incoming invoice. Hence, all external business events that trigger the considered business areas to react must be known in order to derive corresponding business services (sometimes denoted as business use cases) [RR99]. An appropriate means for the identification of these events is the investigation of the interactions between the considered business area and its external partners [WJR+07], which we denote as business roles in this thesis. However, besides business events and business services also given regulations must also be considered, as they are relevant for determining how a business area may (re)act.

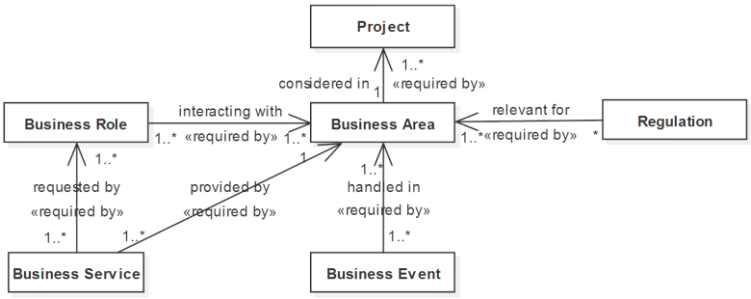


Figure 28. Wider environment view

The reason for starting a requirements elicitation with such a rather business-oriented analysis is the observation that the motivation for an IS can only be made clear if the actual business is well understood [KAP+04].

Below, we define these issue classes as follows:

Definition – Project

"A project is a planned set of interrelated tasks to be executed over a fixed period and within certain cost and other limitations in order to achieve a result." [BC12]

Definition – Business Area

A business area is a part of an organization's operation that is responsible for a certain market segment, respectively for a certain kind of services and goods, or locations, domains, etc. [BC12]

Definition – Business Event

A business event is an external stimulus that triggers a business area to react.

Definition – Business Role

A business role is a role outside a business area that interacts with the business area.

Definition – Business Service

"A business service is a useful work performed by a business area with value for a business role." [ANT10]

Definition – Regulation

A regulation is a given law or standard that can have an impact on the structure or behavior of a business area.

According to the elicitation strategy recommended by this thesis (see section 4.3.2), a meaningful elicitation order of the issues described in this view could look like this: In a first step, project details would be elicited and clarified. Then, the business areas that are in the scope of the project would be described, before the interacting business roles, the business events handled, and the relevant regulations would be addressed. The definition of the wider environment would finally close with the definition of the business services that are provided by the business areas for the business roles identified before.

5.3.2 The Containing System View

The containing system view reflects the internals of the business area and therefore contains the issues that are necessary for defining the internal business organization to be supported by an IS. However, this view just describes the organizational context of an IS, but not the direct

work environment in which this system will be actually operated and used.

In Figure 29, we have elaborated the issue classes that are relevant in the containing system. The roots of this view are all the issues from the wider environment that require the business area to do something internally. In particular, the business services (e.g., travel booking) provided by the business area and the business events to be handled in the business area (e.g., request for reporting) are important in this regard. However, besides the pure business-related triggers, the introduction of the planned IS itself also requires internal reactions. Thus, the planned system administration is considered as an additional root here as well.

For each of these issues, at least one business process that is needed for reacting (e.g., travel application process, monthly report generation process, etc.) shall exist. Each business process, which is a specific kind of a business activity, can either be decomposed recursively into further (sub-) business processes (e.g., travel booking) or merely comprise elementary business activities (e.g., approve travel application) depending on its level of abstraction. Business activities can be performed either by roles (e.g., project managers) or real organizational units (e.g., sales departments), which can also be decomposed recursively.

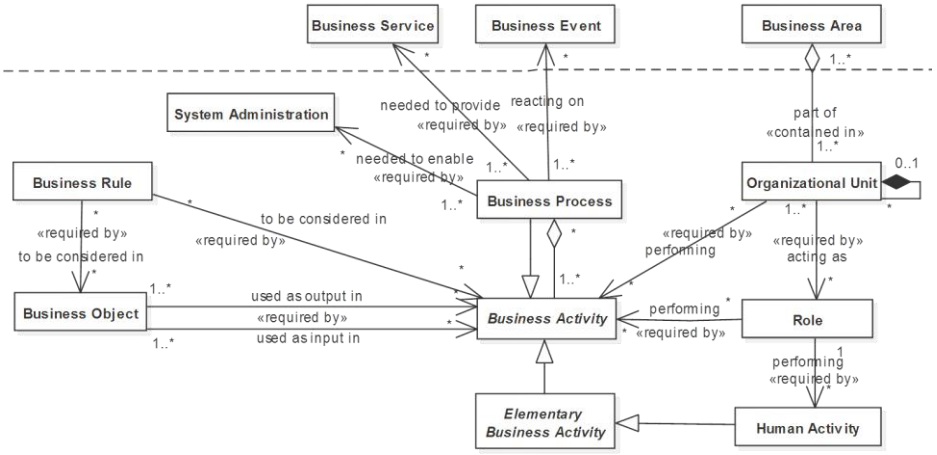


Figure 29. Containing system view

Furthermore, business processes and elementary business activities use business objects (e.g., travel applications, tickets, etc.) as input and output while considering business rules that may govern their execution. A specific kind of elementary business activity are human activities that are performed by a role without any system support.

Below, we define the issue classes of this view as follows:

Definition – System Administration

System administration is the whole set of tasks required for managing the users, assets, and data of an information system.

Definition – Business Activity

A business activity is a work step with clearly identified inputs and outputs that ends in a stable state with value for the business.

Definition – Business Process

A business process is a business activity that comprises a specific ordering of other business activities across time, people, and places.

Definition – Business Object

"A business object is an entity that is handled in or affected by business processes." [SGD+01]

Definition – Business Rule

A business rule is a rule that guides the behavior of an organization. Business rules can either be facts, restrictions (rights and duties), enablers (conditional actions), conclusions (conditional facts), or (conditional) calculations [Wie05].

Definition – Elementary Business Activity

An elementary business activity is an atomic business activity that is performed by a single role or system.

Definition – Human Activity

A human activity is an elementary business activity that is performed by exactly one role without any system support.

Definition – Organizational Unit

An organizational unit is a structural part of an organization that is responsible for a certain area of tasks and topics.

Definition – Role

"A role is a class of real world persons based on a logical set of their responsibilities, rights, and tasks." [Poh07]

According to the elicitation strategy recommended by this thesis (see section 4.3.2), a meaningful elicitation order of the issues described in this view could look like this: First of all, the planned system administra-

tion would be discussed in order to complete the root elements that are needed for the elaboration of the internal procedures. Then, for each root element, the corresponding business processes would be defined and decomposed before their included elementary business activities would be elaborated and classified on the leaf level. For the business processes and elementary business activities, the in- and outgoing business objects would then be identified before the business rules would be determined. The elicitation of business-related issues would finally close with the elaboration and annotation of roles and corresponding organizational units to the process and activities.

5.3.3 The System View

The system view according to the corresponding circle in Alexander's onion model reflects the immediate work environment in which an IS is used, both from a technical and from an organizational perspective. It therefore clarifies the issues that are needed for describing a system's interactions with its environment.

In Figure 30, we have therefore elaborated all issue classes that are relevant for the system. Hence, once an overview of the addressed business areas using the issues of the aforementioned views has been obtained, the concrete expectations with respect to an IS must be defined next. These expectations can be derived from certain business processes [GD07] or from more strategic aims investigated before. At any rate, without clear statements about the business case of an IS, there will exist no basis for determining what the functional scope of the system should be, respectively which parts of the analyzed business processes should be supported or even automated [CFM+02].

In order to define the desired degree of process automation, the system view introduces two further subtypes of elementary business activities, namely system activities (e.g., auto-reply to incoming email) and human system activities (e.g., book hotel). While the former are performed by a system without any human intervention, human system activities are performed by user roles from different workplaces (e.g., office) via certain UI areas (e.g., travel application form). These activities define the concrete way of how user roles will perform certain steps in a business process by using an IS. They are therefore often considered appropriate means for deriving user requirements (often denoted as use cases [Coc00]), as they describe the intended usage rather than solution-oriented system features [Poh07].

However, besides users, partner systems (e.g., SAP) can also interact with a system depending on the system's operation mode (e.g., normal mode, recovery mode, maintenance mode, …). Thus, the interoperation with external partner systems must also be considered [CFM+02], as almost no IS is running in isolation today. This holds especially true for

workflow-oriented IS that aim at automating the execution of business processes by integrating all involved (legacy) applications and users in a holistic manner. Hence, determining the system boundaries, i.e., deciding what is already covered by existing partner systems, requires special attention. The system-system interactions (e.g., synchronization of employee data) and the involved interfaces are therefore important, as interaction data are exchanged (e.g., employee data) between the system and its environment via these interfaces as well as the UI areas.

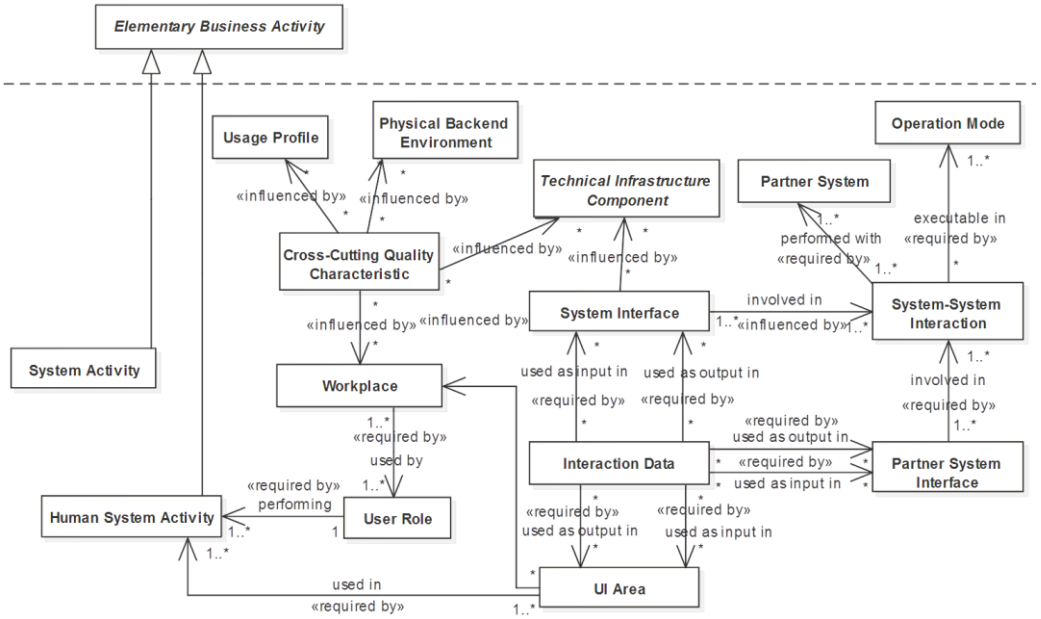


Figure 30. System view

However, due to the critical nature of IS in organizations, non-functional aspects must also be considered as early as possible [ST05]. Besides cross-cutting quality characteristics (in particular those concerning reliability, performance, security, and usability), the technical infrastructure components that are already in place (e.g., existing server hardware, etc.), the physical environment in the backend, (e.g., climate and risk of natural disasters, etc.), the workplaces from which the system will be invoked, and the intended usage profile (e.g., 10000 users between 9 a.m. and 5 p.m.) must therefore be analyzed.

Below, we define the issue classes of this view as follows:

Definition – Human System Activity

A human system activity is an elementary business activity that is performed by a user with a system.

Definition – System Activity

A system activity is an elementary business activity that is performed by exactly one system without any human involvement.

Definition – UI Area

A UI area is a logical part of a system's user interface that allows users to interact with the system in order to carry out certain human system activities.

Definition – User Role

A user role is a role that interacts with a system.

Definition – Workplace

A workplace is a place where a user role works with the system.

Definition – Partner System

A partner system is an external system already available or to be introduced in a parallel project with which the system under development should interact.

Definition – System-System Interaction

A system-system interaction is an interaction sequence between systems for automatically exchanging data.

Definition – System Interface

A system interface is an endpoint provided by the system under development to be invoked by partner systems.

Definition – Partner System Interface

A partner system interface is an endpoint provided by a partner system through which another system can interact with the partner system.

Definition – Operation Mode

An operation mode is a specific state of a system in which a certain (sub)set of capabilities (system functions, quality characteristics) is available.

Definition – Usage Profile

A usage profile is a quantitative description of how a system will be used.

Definition – Interaction Data

Interaction data are (parts of) business objects that are exchanged via an interface or a UI area.

Definition – Physical Backend Environment

A physical backend environment is the location in which a system or certain components are deployed.

Definition – Technical Infrastructure Component

A technical infrastructure component is a piece of external information technology (hardware, software, operation system, middleware, network, etc.) whose services are used by a system to run.

Definition – Cross-cutting Quality Characteristics

A cross-cutting quality characteristic is a non-functional property of a system that concerns the system as a whole

According to the elicitation strategy recommended by this thesis (see section 4.3.2), a meaningful elicitation order of the issues described in this view could like this: In a first step, the technical infrastructure components and the physical backend environment would be discussed before the intended operation modes and usage profile would be elaborated. Then, the partner systems and their system-systems interactions with the system to be introduced would be identified. Based on these results, the required system interfaces and partner system interfaces could be analyzed. In a next step, the user roles for the human system activities within the already analyzed business processes would be determined, including the identification of their workplaces. The UI area to be used in these human system activities as well as all interaction data would then be defined accordingly. The elicitation of issues concerned with the system environment would close with the definition of crosscutting quality characteristics that are also needed to satisfy the stakeholders.

5.3.4 The Kit View

The kit view according to Alexander's onion model reflects the internal issues of the actual system under development to the extent that these issues are already relevant during RE. This view therefore clarifies the issues that need to be known when designing the internals of a system. However, this view does not deal with actual system elements, as it merely aims at addressing the requirements that may exist on their realization.

In Figure 31, we elaborated the issues that are relevant in the kit, at least from an RE point of view. The roots of this view are the human system activities and the system activities introduced above. For both activities, system functions exist that realize (parts) of these activities. During the development of these functions, realization policies have to be considered. In particular, realization policies may also influence the UI style according to which the user interface of a system has to be designed.

Thus, the pure usage-oriented approach that has existed so far is now complemented with realization-specific issues.

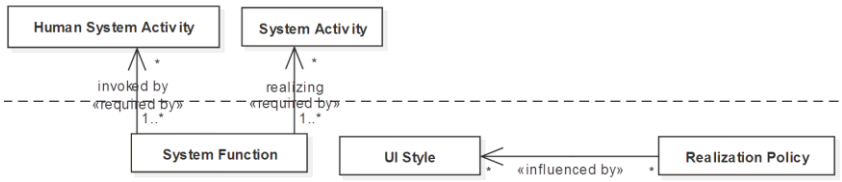


Figure 31. Kit view

Below, we define these issues as follows:

Definition – System Function

A system function is an atomic reaction (i.e., state change or response) of a system that is triggered by an external stimulus, e.g., an environmental change, or by an explicit request of a user or an external system.

Definition – Realization Policy

A realization policy is a constraint for the development of the system under development including security policies, desired architecture styles, COTS or open source to be used, development activities, and development technology.

Definition – UI Style

A UI style is the look and feel or appearance of the user interface respectively the representation rules to be followed.

According to the elicitation strategy recommended by this thesis (see section 4.3.2), a meaningful elicitation order of the issues described in this view could look like this: First, the system functions would be derived based on the human system activities or the system activities identified before. Then, realization policies and, in particular, requirements concerning the UI style would be elaborated in order to provide developers with corresponding design constraints.

5.4 Summary

Issues define the elements for which requirements have to be elicited in order to specify a software system. However, as requirements processes for IS are basically different from requirements processes for other kinds of systems (e.g., embedded systems) [NE00], the issues to be discussed in this domain are very specific and cannot be covered by abstract (meta-) models such as those proposed by [GKB08], [CDS+05], [VMT07],

[WW92], or [AHH11]. However, models such the BPMN specification [OMG12b], the Soffer-Wand ontology [SW04], or the reference models by Scheer [Sch95], which discuss specific business issues in all details, are also no alternative, as they neglect important aspects that are necessary for RE.

In this chapter, an issue model for RE in the IS area has therefore been introduced. The model describes the issues that are typically relevant in IS projects and the relationships among them. As it provides a formalized description of the conceptual world to be discussed during elicitation, it therefore acts as a forth input for our tailoring approach (see chapter 6). In particular, the issues described in the model reflect the elements with which both SPL constraints and information needs can be concerned. This means that different knowledge about an SPL can be assigned directly to a certain issue in order to provide requirements engineers with this knowledge in a corresponding elicitation step. Furthermore, the conceptual relationships among issues defined in the model allow determining a meaningful elicitation sequence based on the rules introduced in section 4.3.2.

However, even though we carefully elaborated the issues using an iterative approach, we are aware that the concrete information needs in a certain development context and, thus, the actually relevant issues, may vary. Especially when systems are built in a reuse-based way, decisions will affect which issues are relevant and which are not (see chapter 2). Furthermore, the point in time at which a certain issue must be discussed is not fixed either. For instance, it may be possible that very technical issues such as the existing infrastructure components must be known earlier in a certain context than the business processes. Thus, it is important to know that the elicitation sequences recommended above are only meaningful within one view. The overall processing order of issues can therefore differ significantly when the issue model has been tailored. In this case, only the relationship stereotypes must still be considered, as, for instance, a contained issue cannot be discussed before a containing issue has been discussed. Thus, this issue model cannot be used out-of-the-box.

The next chapter therefore introduces the ARE tailoring approach and explains how the issue model is used respectively adapted for addressing the actual information needs of an SPL organization. The initial issue model as presented here is an indispensable input for this aim, as it reflects established elicitation procedures. If method tailors know about these, they are not likely to violate best practices when defining tailored ARE instructions.

6 Tailoring ARE Instructions based on an SPL

"Tailors are the smartest people because they take measures over and over again, rather than relying on old information."
George Bernard Shaw

As already described in the solution idea (see section 1.3.2), the overall purpose of this thesis approach is to tailor ARE instructions in an effective and systematic manner in order to provide requirements engineers in AE with better knowledge about a given SPL.

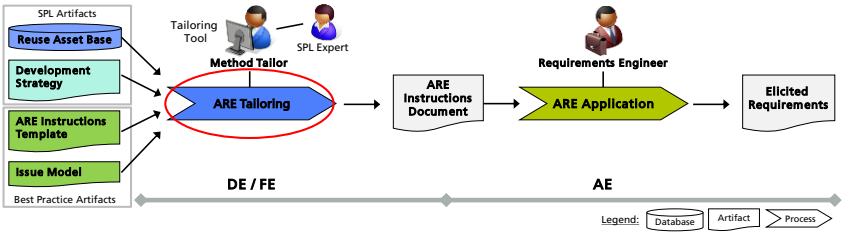


Figure 32. ARE tailoring method within thesis approach

To support this aim, this chapter introduces the main methodological and engineering contribution of this thesis, namely the actual tailoring method and its tool support (see Figure 32). Hence, the purpose of this chapter is to present an algorithmic approach that allows method tailors (i.e., people that are responsible for defining methods and processes in an SPL organization) to extract and incorporate important process and product knowledge from an SPL into an ARE process as well as supporting artifacts.

6.1 Research Approach

The tailoring approach described in this chapter was developed in parallel to other thesis components such as the conceptual ARE model described in chapter 2 and the ARE instruction template described in chapter 4, as these components are strongly intertwined.

In a first step (step 1), an initial version of the conceptual ARE model was therefore used to identify and describe the basic steps of the tailoring approach (see Figure 33). To make this happen, we created a diagram that contained all elements of the conceptual ARE model except those elements that will be a work result of the AE phase (e.g., requirements, system, …). The reason for excluding these elements was based on the intention of the tailoring approach to develop ARE instructions. Thus, elements that will be created during AE can apparently not be processed during the upstream tailoring that has to take place during DE/FE. Furthermore, we also removed all classification elements and all "satellite" elements that just provide optional information about an element of actual importance (e.g., the SPL specification as a means for documenting the software product line). The graph of the remaining elements was then recursively processed by following the elements' relationships starting with the element "Software Product Line". This processing basically included the definition of simple tailoring instructions in the form of *"Analyze the <related element> <association> the <current element>"*, e.g., *"Analyze the Product Line Architectural Elements that are part of the Product Line Architecture"*. These statements were then manually consolidated into more meaningful steps, and extended with additional explanations in order to increase the preciseness of guidance. The concrete foundation of each tailoring step within the ARE model is described in the corresponding subsections below.

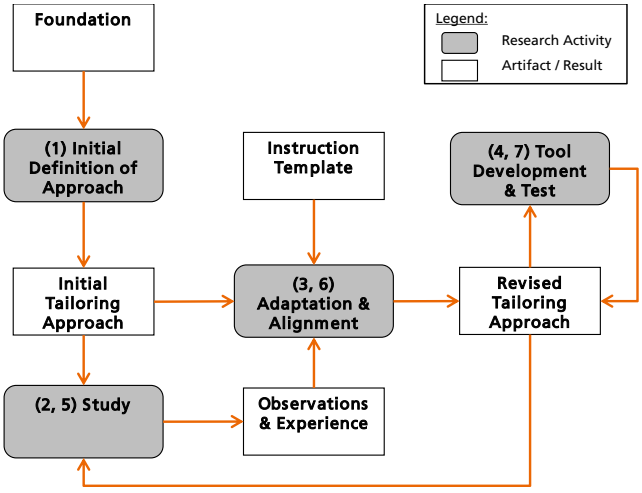


Figure 33. Research approach for tailoring approach

The resulting tailoring approach was then applied in an early feasibility study (step 2), where we tried to derive an ARE instruction by elaborating SPL knowledge with people from a medium-sized software company. Based on the experience made during this study, an improvement of the tailoring method took place (step 3). However, as the tailoring method had to be aligned with the underlying foundation, an adaptation of the

conceptual ARE model was done first. Based on the improvements incorporated in this model, more formalization could then be added in the tailoring steps also, which made the entire tailoring approach more accurate. The improved versions of the tailoring approach and of the underlying ARE model were then aligned with the ARE instructions template in its initial version. During this task, misalignments occurred, which made it necessary to slightly adapt all three thesis components again.

In the next step, a tool was developed for (semi-) automating or at least facilitating the steps of the tailoring method (step 4). During this development, the tailoring steps were again made more precise, either to be implemented in software or to provide better guidance to the tool users. The resulting tool was then used with fictitious examples to test the correctness of the implemented algorithms. Besides the actual tailoring approach, the ARE instructions template, respectively the conceptual ARE model, was also slightly adapted when a need for correction was recognized.

The tested tool as well as a fictitious example were then used by two experienced software architects and two requirements engineers at Fraunhofer IESE as well as by a person from industry in order to derive an ARE instructions document (step 5). The feedback we received from this case study (see evaluation in chapter 7) with regard to the tool-supported approach was used to identify required adaptations (step 6 and 7) to be addressed in future work (see chapter 8).

6.2 Tailoring Overview

While the reuse asset base provides the product knowledge of the SPL, the development strategy and the RE best practices provide knowledge about the processes of an SPL organization. Thus, the tailoring approach aims at addressing the incorporation of SPL knowledge holistically and does not merely focus on product reuse as most other approaches do. However, as shown by Carbon [Car11], products and processes are closely intertwined, at least in an ideal development setting. This means that the elements of a given architecture may influence the processes regarding how to develop systems in an efficient manner. The tailoring approach introduced here therefore takes this dependency into consideration and tries to align certain development activities with the underlying architectural elements wherever possible. Therefore, the tailoring steps dealing with the extraction of process knowledge strongly depend on the underlying product base.

Based on the conceptual ARE model described in chapter 2, the tailoring approach to be performed by method tailors during DE/FE therefore processes the aforementioned inputs as follows (see Figure 34). In the first step, basic information is extracted from the SPL specification in order to

characterize the systems to be derived from it during AE projects. In the second and third steps, the architectural elements, respectively the architectural element types, are then extracted from the product line architecture. These elements and types are used for characterizing the flexibility classes that are supported in step four. In step five, the assumptions made by these flexibility classes are then elaborated taking into consideration the development strategy and product line architecture. Furthermore, the issue model that reflects RE best practices (see chapter 5) is used to align the assumptions from the solution space with elements (i.e., issues) of the problem space.

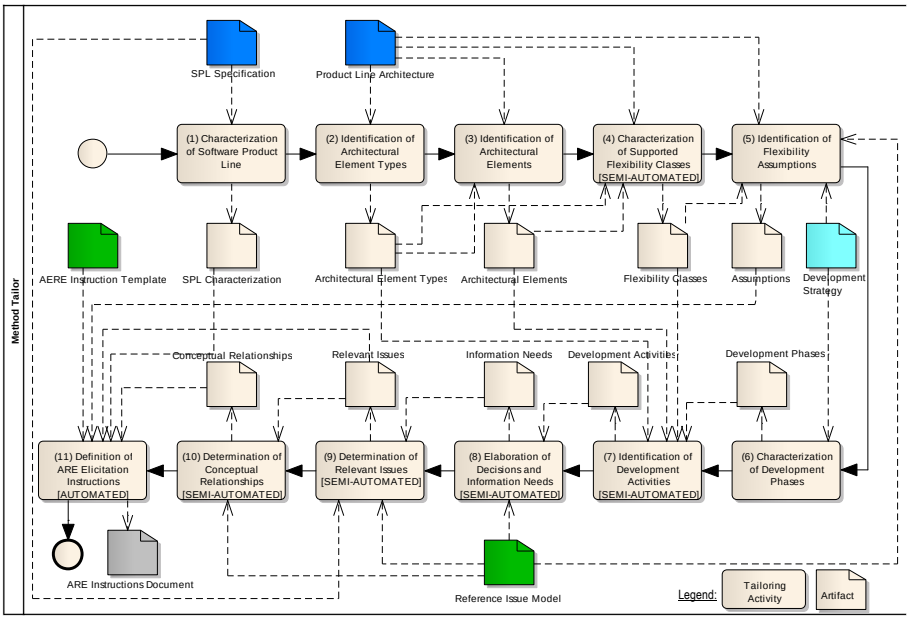


Figure 34. Activities and artifacts of the tailoring approach

Based on the development strategy, the development phases in which a customer-specific system should be developed are then characterized in step six, before the contained development activities are identified in step seven for each of these phases. As the development process is strongly aligned with the product line architecture, the extracted architectural element types, architectural elements, and flexibility classes are considered here.

In step eight, these development activities are used to elaborate the decisions to be made in them as well as the corresponding information needs. In step nine, the information needs are used for determining the relevant issues to be discussed during an ARE process. Again, the best practices reflected in the issue model are taken into consideration. This list of relevant issues as well as the relationships described in the issue model are then used to determine the conceptual relationships between

the issues of interest in step ten. In step eleven, these issues, their relationships, the assumptions concerned with them, the SPL characteristics extracted in the first tailoring step, as well as the ARE instructions template (see chapter 4) are finally used to create an ARE instructions document. These instructions are then ready to use in AE projects.

6.3 Tailoring Steps in Detail

In this subsection, the eleven aforementioned steps of the tailoring approach are introduced in detail. For each step, we describe the purpose and the rationale, the underlying foundation from chapter 2, algorithmic guidance on how method tailors should perform this step, as well as the automation support provided so far.

6.3.1 Characterization of Software Product Line

Purpose. In this step, the basic characteristics of systems derived from a given SPL are described. The purpose of this step is to understand the general nature and intention of these systems in order to get an awareness for the target group, the application domain, and the benefits these systems may have for potential customers in an AE project. Thus, this step aims at extracting the informal background information AE requirements engineers should have when eliciting requirements for such systems.

Foundation. The foundation of this tailoring step is the class "Software Product Line" within the underlying model introduced in chapter 2 (see Figure 35). According to this model, the systems to be developed in AE projects are derived from a given SPL, which is documented in a corresponding SPL specification.

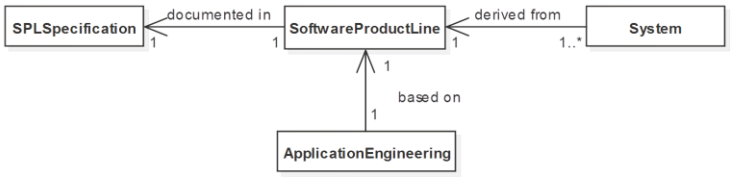


Figure 35. Foundation of tailoring step 1

Input. SPL specification.

Procedure. In order to characterize the SPL for which specific ARE instructions should be tailored, information has to be extracted from the SPL specification. Thus, the following procedure has to be applied in this tailoring step:

Tailoring Step 1 – Analyze the software product line on which the application engineering should be based.

To do so, describe the systems to be derived from the software product line according to the following questions:

- *What is the main purpose of these systems?*
- *Which business domain or market segment is addressed?*
- *Who are the typical customers?*
- *Who are the typical end users?*
- *What are the benefits these systems provide to their audience?*
- *How are these systems typically integrated into their usage environment?*
- *Which main constraints and limitations do these systems have?*

To extract this information, consider the SPL specification in which the software product line is documented. If you cannot find the required information here, feel free to interview an SPL expert.

Output. SPL characterization.

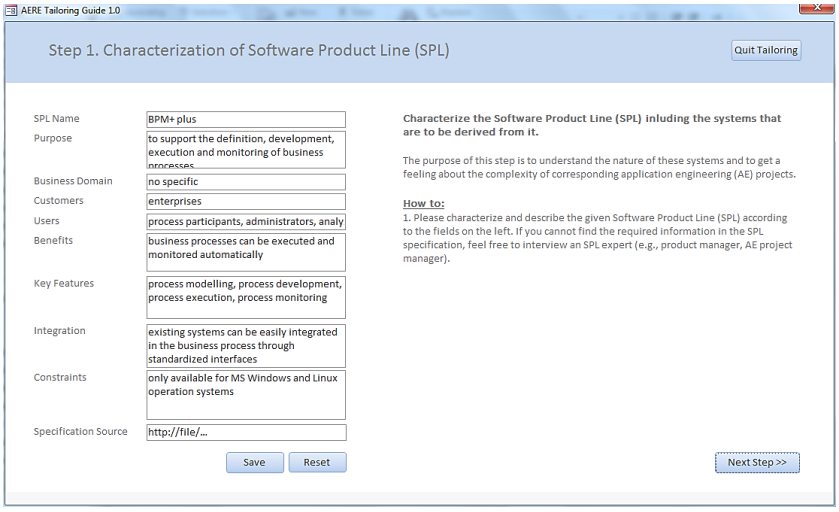


Figure 36. Screenshot of tailoring tool with example (step 1)

Rationales. The reason for asking exactly these questions is based on the observation that the corresponding answers allow people to get a basic understanding of the characteristics of a software system or a

component [FGM07]. The reason for carrying this step out manually is that today's software systems are not described in a semantic way, which would allow extracting this knowledge automatically from either code, architectural documentation, or the requirements [Joh10].

Tool Support. Due to the aforementioned reason, the support of the tailoring tool for this step (see Figure 36) is limited to the description of the aforementioned procedure as well as to the provision of documentation possibilities.

6.3.2 Identification of Architectural Element Types

Purpose. In this step, all architectural element types that are used in the product line architecture are identified and characterized. The purpose of this step is to determine the classes of elements of which both the SPL and also all derived systems are basically composed. Thus, this step defines the valid element types to be used during development, which helps to identify the actual architectural elements, which are already part of the product line architecture, more systematically in the next step. Furthermore, an awareness of valid architectural element types is also needed to support systematic determination of possible customer-specific extensions. This means that for each architectural element type it is checked whether new architectural elements of this type may be specifically extended in a certain AE project or not. However, both the determination of such allowed extensions and the identification of already existing architectural elements are not part of this tailoring step yet, and will be covered in some of the following ones.

Foundation. The foundation of this tailoring step is the class "Architectural Element Type" within the underlying model introduced in chapter 2 (see Figure 37). According to this model, each software product line has a product line architecture comprising different product line architectural elements. The product line architectural elements realize certain architectural element types that are (only) allowed in the product line architecture. Thus, learning about these architectural element types is the prerequisite to understand the product line architecture.

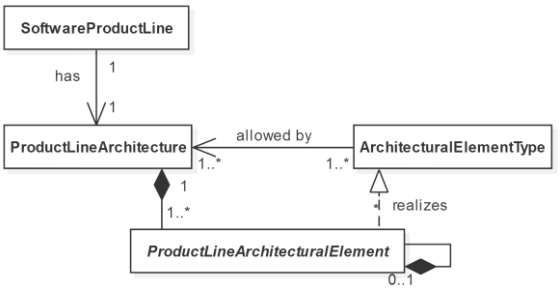


Figure 37. Foundation of tailoring step 2

Input. Product line architecture.

Procedure. In order to identify the architectural element types, the product line architecture has to be analyzed. Thus, the following procedure has to be applied in this tailoring step:

Tailoring Step 2 – Analyze the architectural element types that are allowed in and by the product line architecture.

To do so, describe the architectural element types according to the following questions:

- *What is the name of the architectural element type?*
- *What is the common purpose of the architectural elements belonging to the architectural element types?*

To extract this information, consider the product line architecture in which the architectural element types are defined. If you cannot find the required information here, feel free to interview an SPL expert. To get an idea of what architectural element types might be, consider reference architectures typically used in the domain addressed by the SPL.

Output. (List of) architectural element types.

Rationales. The reason for asking only these two questions is that the architectural element types are just to be used as an anchor for the identification of architectural elements and extension classes that are of higher value for the creation of ARE instructions. Thus, knowing which architectural element types exist is sufficient for achieving the goal of the tailoring approach.

The reason for this manual extraction is that architectures in today's software systems are documented in very different ways. Furthermore, as the scientific contribution of this work is not in the area of architecture analysis or reengineering, providing algorithms that extract architectural element types automatically was not within its scope. However, as future work (see chapter 8), it would be a good idea to integrate this step with upfront architecture analysis, for example based on SAVE [DKL09].

Tool Support. For the reason mentioned above, the support of the tailoring tool for this step (see Figure 38) is limited to the description of the aforementioned procedure as well as the to the provision of documentation possibilities.

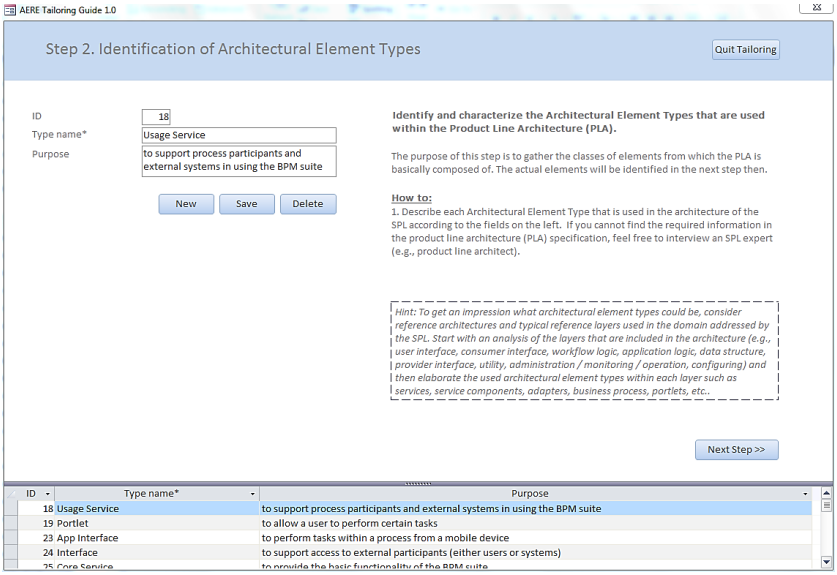


Figure 38. Screenshot of tailoring tool with example (step 2)

6.3.3 Identification of Architectural Elements

Purpose. In this step, all product line architectural elements that realize the architectural element types identified before are identified and characterized. The purpose of this step is to understand of which concrete elements the given product line architecture is currently composed. Thus, the common and variable architectural elements are recursively extracted from the product line architecture. This is a prerequisite for the identification of required system instantiation activities as well as possible customer-specific modifications in later tailoring steps. However, neither the determination of such allowed modifications nor the identification of related system instantiation activities are part of this tailoring step yet and will be addressed in some of the following ones.

Foundation. The foundation of this tailoring step is the class "Product Line Architectural Element" within the underlying model introduced in chapter 2 (see Figure 39). According to this model, product line architectural elements realize the aforementioned architectural element types and are either common architectural elements or variable architectural elements within the product line architecture. Variable architectural elements can be further distinguished into optional, alternative, and optional alternative architectural elements. All product line architectural elements have in common that they may be composed of other product line architectural elements.

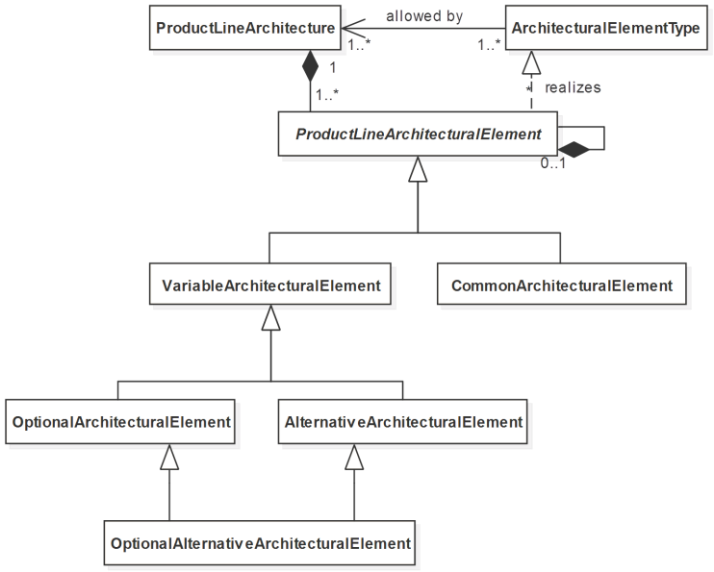


Figure 39. Foundation of tailoring step 3

Input. Product line architecture, (list of) architectural element types.

Procedure. In order to identify the architectural elements, the product line architecture has to be analyzed again. Thus, the following procedure has to be applied in this tailoring step:

Tailoring Step 3 – Analyze the product line architectural elements that are part of the product line architecture.

1. *For each architectural element type identified before, describe the product line architectural elements according to the following questions. Start at the highest decomposition level.*

   - *What is the name of the product line architectural element?*
   - *What is the architectural element type of the product line architectural element?*
   - *What is the purpose of the product line architectural element?*
   - *Is the product line architectural element an alternative within the derived systems (i.e., can it be implemented differently)?*
   - *Is the product line architectural element an optional within the derived systems (i.e., does it need to be part of all derived systems)?*
   - *In which parent product line architectural element is the product line*

architectural element included?

2. *For each alternative architectural element identified in the first step (only for alternatives!), recursively decompose this element into its child elements and answer the above questions again for each identified child element.*

For extracting this information, consider the product line architecture in which the architectural elements are defined. If you cannot find the required information here, feel free to interview an SPL expert.

Output. (List of) architectural elements.

Rationales. The reason for asking these questions is the need to know for which purpose a certain product line architectural element is needed and how it is embedded in the overall product line architecture. Furthermore, the questions concerning variability and optionality are indispensable to elaborate corresponding decisions and information needs in a later tailoring step. In this regard, we consider variability and optionality as being orthogonal (see section 2.3.1 for definitions

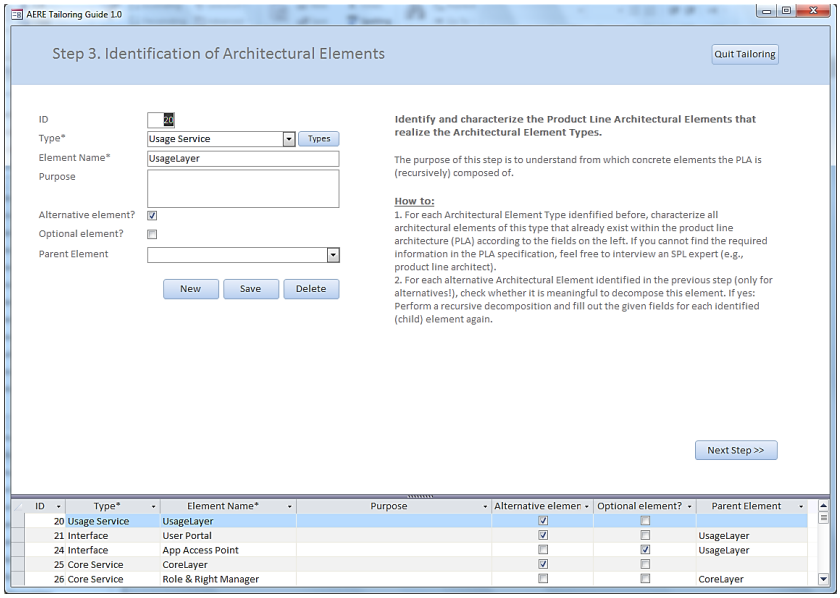


Figure 40. Screenshot of tailoring tool with example (step 3)

The reason for recursively decomposing only alternative elements is based on the fact that the sub-elements of the non-alternative product line architectural elements are always implemented in the same way and do therefore not influence any decision made during AE. Thus, these elements will not lead to certain information needs that must be known for tailoring an ARE process based on the SPL.

Finally, the reason for manually extracting the product line architectural elements is the same as for the architectural element types before.

Tool Support. Due to the aforementioned reason, the support of the tailoring tool for this step (see Figure 40) is limited to the description of the aforementioned procedure as well as the to the provision of documentation possibilities. However, as already mentioned, a combination with architecture analysis tools could be meaningful future work.

6.3.4 Characterization of Supported Flexibility Classes

Purpose. In this step, the flexibility classes that are supported by the product line architecture are identified and characterized. The purpose of this step is to understand which customization possibilities are provided by the given SPL beyond the scope of the variabilities already predefined during the DE/FE phase. Thus, possible extensions and modifications that are either technically or strategically allowed are determined, including an estimation of their costs. This means that for each product line architectural element identified before, it is decided whether this element may also be realized in a novel and still unknown manner during an AE project or not. Furthermore, for each architectural element type, it is decided whether still unforeseen architectural elements of this type may be individually added during an AE project or not. Both aspects are important for if only the predefined variants are considered. The characterization of supported flexibility classes is therefore the key tailoring step for making AE requirements engineers aware of the capabilities of the SPL, respectively its enhanced customization possibilities. However, the determination of the assumptions that must hold in order not to contravene the architecture or the intended development strategy when exploiting these flexibilities is not part of this tailoring step, and will be addressed in the next one.

Foundation. The foundation of this tailoring step is the class "Flexibility Class" within the underlying model introduced in chapter 2 (see Figure 41). According to this model, flexibility classes are either extension classes concerning architectural element types or modification classes concerning architectural elements. They are enabled by architectural elements that provide possibilities for extending further elements or modifying the realization of existing ones.

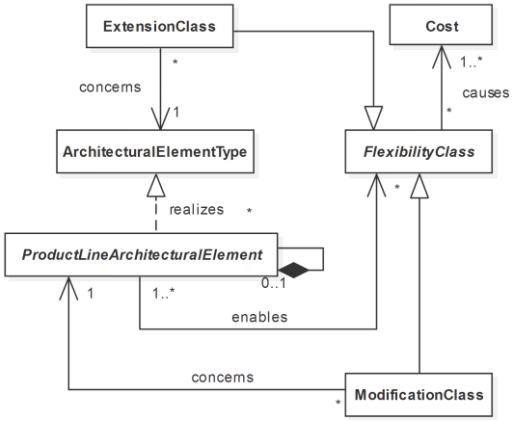


Figure 41. Foundation of tailoring step 4

Input. Product line architecture, (list of) architectural element types, (list of) architectural elements.

Procedure. In order to identify the flexibility classes, the product line architecture must therefore be analyzed again. The following procedure describes how to perform this tailoring step:

Tailoring Step 4 – Analyze the flexibility classes that are enabled by the product line architectural elements.

1. *For each product line architectural element identified before, determine whether there is a modification class concerned with it.*

2. *For each architectural element type identified before, determine whether there is an extension class concerned with it.*

3. *Describe all elaborated flexibility classes according to the following questions:*

   - *What is the name of the flexibility class?*
   - *What is the purpose of the flexibility class (e.g., the extension of new business process components)?*
   - *What is the average development effort for creating a new artifact (i.e., an architectural element or its realization) when exploiting the flexibility class (1 = low, 2 = medium, 3 = high)?*
   - *What is the average impact on other architectural elements when exploiting the flexibility class (1 = low, 2 = medium, 3 = high)?*
   - *Which product line architectural element enables the flexibility class?*

- *With which product line architectural element or which architectural element type is the flexibility concerned?*

To extract this information, consider the product line architecture in which the architectural elements are defined. If you cannot find the required information here, feel free to interview an SPL expert.

Output. (List of) supported flexibility classes.

Rationales. The reason for asking these questions is the need to provide AE requirements engineers with sound information about possible extensions and modifications, their rationales (i.e., the enabling architectural elements), and their exploitation costs. This step therefore requires the prediction of development and integration costs by a means of sound architectural analysis. Thus, product line architects should be involved in this step. Cost estimations based on architectural impact analysis are beyond the scope of this work, but are interesting for future research (see chapter 8).

Nevertheless, we consider it to be sufficient to give only coarse cost estimations (low, medium, high) to stakeholders during an elicitation session. This means that we assume that it is not important to directly state concrete monetary costs or delivery delays when a modification or extension request is stated. Rather, when stakeholders insist on a requirement even though they have been informed about the coarse dimension of its impact, a detailed estimation can still be done afterwards.

Furthermore, the reason to extract flexibility classes from the architecture is based on the assumption that an architecture enables changes that go beyond those expressed in an explicitly-anticipated variability model (VM).

Tool Support. While tool support was limited to the provision of instructions and documentation possibilities in the previous steps, a higher degree of automation support is achieved here (see Figure 42). Before displaying the instructions and forms for this tailoring step, the tool automatically generates a proposal for all possible extensions and modification classes based on the architectural element types and product line architectural elements identified before. Thus, for each architectural element type, the tool generates an extension class using the following template:

Name: *<architectural element type name> Extension*

Purpose: *to enable the realization of customer-specific <architectural element type name>s not covered in the SPL so far.*

Furthermore, for each alternative architectural element, the tool generates a modification class using the template:

Name: *<architectural element name> Modification*

Purpose: *to enable a customer-specific realization of the <architectural element name> for the case that its already foreseen variants are not sufficient.*

Finally, for each common architectural element or optional architectural element (that is not an alternative), the tool generates a modification class as follows:

Name: *<architectural element name> Replacement*

Purpose: *to enable the replacement of the unique <architectural element name> with a customer-specific implementation, if required.*

Thus, when applying this tailoring step by using the tool support, method tailors must still enter the costs and enabling product line architectural elements for each flexibility class (see Figure 42). Of course, the automatically generated list of flexibility classes should first be reduced to those classes that are actually supported.

A further tool support in this step is that the isolated estimations for the development of artifacts when exploiting a flexibility class and for the integration within the entire architecture are automatically combined into one "price". For instance, when the development and the impact were both estimated as being "high", the overall costs when exploiting the corresponding flexibility class will be "very high".

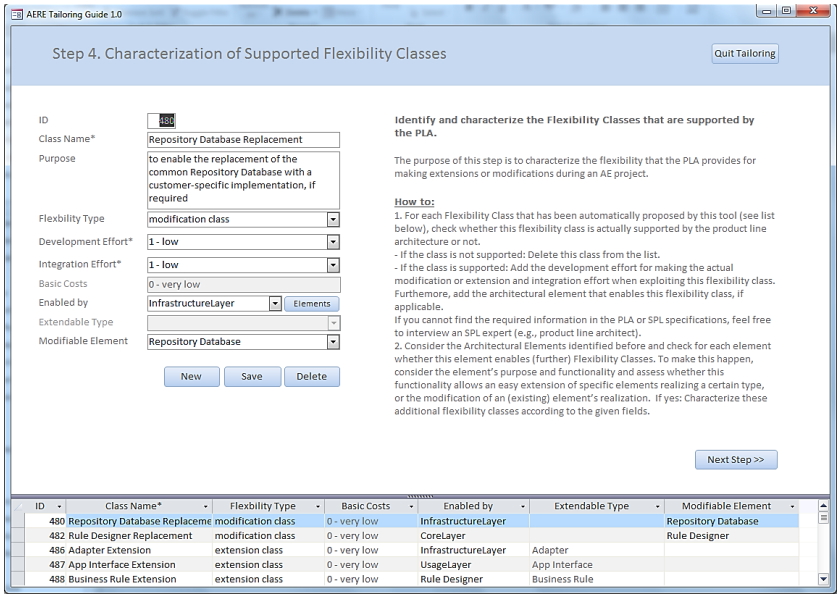


Figure 42. Screenshot of tailoring tool with example (step 4)

6.3.5 Identification of Flexibility Assumptions

Purpose. In this step, the assumptions that must hold in order to exploit the aforementioned flexibility classes are determined. The purpose of this step is to understand under which circumstances customer-specific requirements can be economically realized (even though not anticipated before) without contravening the product line architecture or the intended development strategy. Thus, the set of implicitly anticipated requirements to be supported by the given SPL is declaratively prescribed through these assumptions. This means that for different issues of the application domain, properties are defined that must be fulfilled in order to allow corresponding requirements to be economically feasible. This is important for aligning the capabilities and constraints of a given SPL with elements to be discussed and processed during ARE. The identification of flexibility assumptions is therefore the key step for making AE requirements engineers aware of given SPL constraints.

Foundation. The foundation of this tailoring step is the class "Assumption" within the underlying model introduced in chapter 2 (see Figure 43). According to this model, assumptions are always concerned with an issue, and are either hard assumptions (that must hold) or soft assumptions (that should hold). Assumptions are made by one or more flexibility classes, but their rationales is typically the risk of violating the product line architecture when exploiting these flexibility classes without any restrictions.

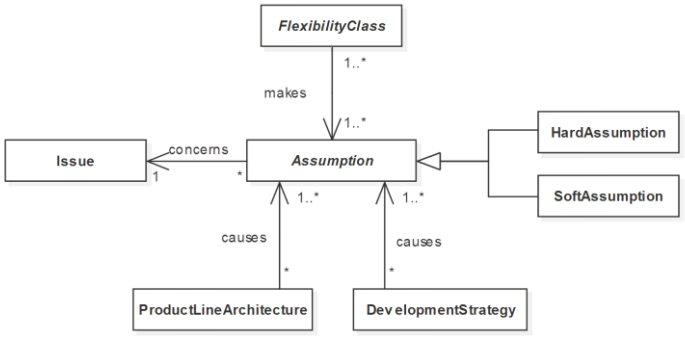


Figure 43. Foundation of tailoring step 5

Input. Product line architecture, development strategy, (list of) supported flexibility classes, reference issue model.

Procedure. In order to identify the flexibility assumptions that must hold in order to be able to exploit the aforementioned flexibility classes without contravening the product line architecture or development strategy, the following procedure has to be applied:

Tailoring Step 5 – Analyze the assumptions that are made by the flexibility classes.

For each flexibility class identified before, describe the assumptions that must hold in order to not contravene the product line architecture or development strategy according to the following questions:

- *What exactly does the assumption express / constrain?*
- *Why does the assumption exist and why must it hold?*
- *Is the assumption a hard assumption (that is known to always lead to economic unfeasibility when it does not hold) or a soft assumption?*
- *Which issue (to be realized or supported by a derived system) is affected by this assumption?*

To extract this information, consider the product line architecture, the development strategy, as well as the issue model introduced in chapter 5. If you cannot find the required information here, feel free to interview an SPL expert.

Output. (List of) assumptions.

Rationales. The reason for asking these questions is the need to understand the assumptions the SPL makes about the requirements that might be stated during AE projects. Particularly to allow AE requirements engineers to know which requirements are basically feasible and which are not (without prescribing all requirements in advance), the assumptions must be clearly stated and, in particular, aligned with the issues with which requirements can be concerned. For instance, when eliciting requirements concerning business processes, assumptions that are made with regard to business processes can be directly taken into consideration. To negotiate convincingly, it is furthermore important for AE requirements engineers to know why these assumptions must hold. Otherwise, stakeholders will probably not be willing to accept when a requirement is put into question. In this regard, it is also important to elaborate whether an assumption is hard (mandatory) or soft (desired but not mandatory). Thus, all aforementioned questions aim at gathering this information.

This step therefore again requires a sound knowledge of the product line architecture, but also of the development strategy. An important source for determining the assumptions are the technologies, protocols, tools, standards, and regulations or policies on which the SPL is based, or from which customer-specific systems should be derived during an AE project. For instance, when using a product line architectural element "Business Process Designer", which implements the BPMN standard, a corresponding assumption concerning the issue "Business Process" would be that the business processes are modeled in BPMN and not in another notation. Thus, product line architects as well as technology experts should

be involved in this step in order to externalize the existing assumptions and assess whether they are hard or soft. In future work, it would be interesting to see to which degree it is possible to derive the assumptions from the used technologies automatically (e.g., by processing meta-information provided by tools or components). Furthermore, it would be an interesting enhancement to implement algorithms that automatically detect the issues with which an assumption is concerned, as we have found it to be challenging for practitioners to make this alignment when doing the tailoring for the first time. However, both of those extension possibilities are beyond the scope of this work.

Tool Support. In contrast to the previous step and due to the aforementioned reason, the tool support (see Figure 40) is limited to the description of the aforementioned procedure as well as the to the provision of documentation possibilities.

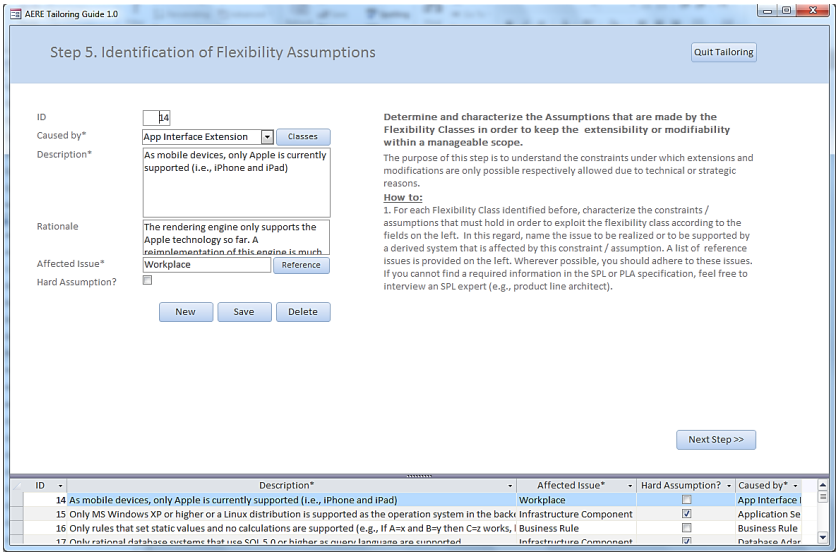


Figure 44. Screenshot of tailoring tool with example (step 5)

6.3.6 Characterization of Development Phases

Purpose. In this step, the development phases in AE projects according to the underlying development strategy and process are identified and characterized. The purpose of this step is to understand how customer-specific systems based on the given SPL are basically developed, configured, and integrated. Thus, the increments and milestones according to which customer-specific systems are built are determined and described. Based on this, development activities and corresponding information needs can then be identified in later steps. The characterization of development phases is therefore important for elaborating until which point

in time (milestone) during AE the different issues have to be discussed in order to gather all information that is needed to perform the development activities of the next phase. However, the elaboration of these activities and information needs is not part of this step, and will be cover in the next two steps.

Foundation. The foundation of this tailoring step is the class "Development Phase" within the underlying model introduced in chapter 2 (see Figure 45). According to this model, development phases are parts of a development process with a defined order. Development phases are determined by the underlying development strategy for AE. Milestones, where certain results are achieved, mark the end of a phase.

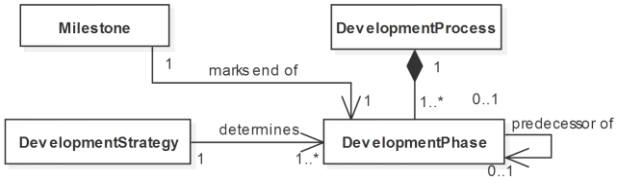


Figure 45. Foundation of tailoring step 6

Input. Development strategy.

Procedure. In order to characterize the development phases according to which customer-specific systems should be derived from the given SPL, the following procedure has to be applied:

Tailoring Step 6 – Analyze the development phases determined by the development strategy.

To do so, describe the development phases determined by the development strategy according to the following questions:

- *What is the name of the development phase?*
- *What is the purpose of the development phase?*
- *Which milestone is reached at the end of the development phase?*
- *Is the development phase carried out several times in an AE project (i.e., is it iterative)?*
- *Which development phase is the predecessor of the development phase?*

To extract this information, consider the development strategy according to which systems should be developed (i.e., configured, installed, extended, etc.) in an AE project. If you cannot find the required information here, feel free to interview an SPL expert.

Output. (List of) development phases.

Rationales. The reason for asking these questions is the need to understand what is done in different development phases and which separating milestones exist. The elaboration of this information is a prerequisite to finding out which requirements must be available at which point in time during an AE project.

Tool Support. As this tailoring step is very informal, the tool support (see Figure 46) is again limited to the description of the aforementioned procedure as well as the to the provision of documentation possibilities in this step.

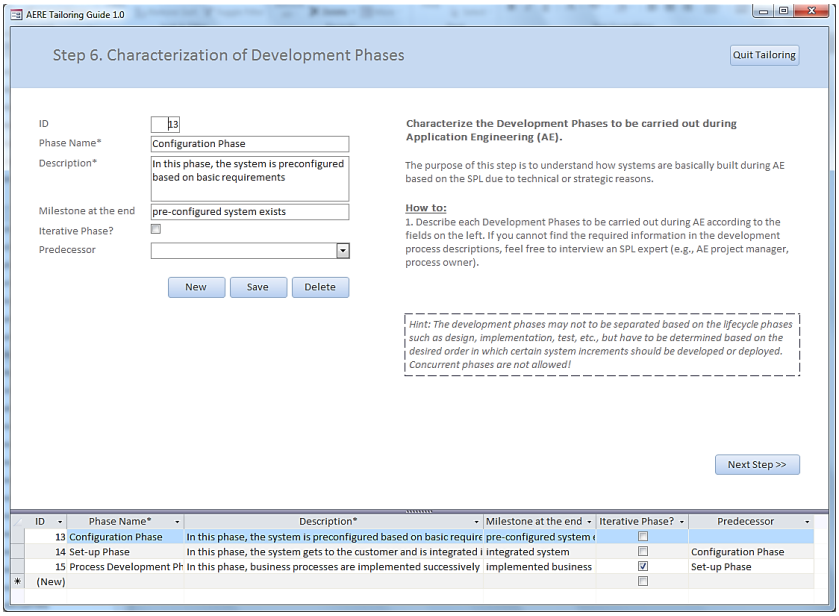


Figure 46. Screenshot of tailoring tool with example (step 6)

6.3.7 Identification of Development Activities

Purpose. In this step, the development activities to be performed during AE are identified and described. The purpose of this step is to understand how a customer-specific system is to be developed concretely based on the given SPL. Thus, all activities that are needed to instantiate the predefined variabilities, or that are needed to extend, respectively modify, the system when exploiting the supported flexibility classes are determined. This step is a prerequisite for being able to elicit the information needs that have to be satisfied through ARE processes in order to allow effective and efficient system development. In particular, without knowing the concrete responsibilities of the development roles, it is hard

to elaborate the corresponding information needs systematically. Thus, we also apply the idea of information-oriented RE [SLS+09] for the elaboration of information needs in this context. However, the elicitation of these needs is not part of this tailoring step.

Foundation. The foundation of this tailoring step is the class "Development Activity" within the underlying model introduced in chapter 2 (see Figure 47). According to this model, each development activity is part of a development phase performed by exactly one role and arranged in a defined order. Each development activity is either an inclusion activity, instantiation activity, extension activity, redevelopment activity, or a miscellaneous activity depending on the architectural element respectively architectural element type it is concerned with. The development activities are closely aligned with the product line architecture and therefore determined (to a large degree) by the result of the previous tailoring steps.

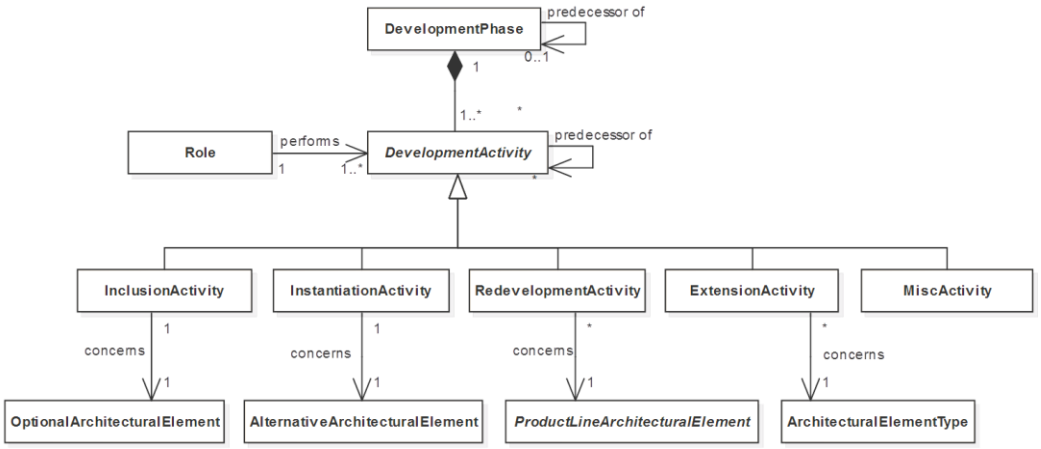


Figure 47. Foundation of tailoring step 7

Input. (List of) development phases, (list of) supported flexibility classes, (list of) architectural element types, (list of) architectural elements.

Procedure. In order to identify the concrete development activities that have to be performed in order to instantiate and extend a customer-specific system based on the given SPL, the following procedure has to be applied:

Tailoring Step 7 – Analyze the development activities that are part of the development phases.

1. *For each optional architectural element identified before, determine the inclusion activity that is concerned with it.*

2. *For each alternative architectural element identified before, determine the*

instantiation activity that is concerned with it.

3. *For each product line architectural element identified before with which a modification class is concerned, determine the redevelopment activity that is concerned with it.*

4. *For each architectural element type identified before with which an extension class is concerned, determine the extension class.*

5. *Describe all elaborated development activities according to the following questions:*

   - *What is the name of the development activity?*
   - *What is the purpose of the development activity?*
   - *Who (which role) is responsible for doing the development activity?*
   - *To which development phase does the development activity belong?*
   - *With which product line architectural element or which architectural element type is the flexibility concerned?*
   - *Is the development activity optional (i.e., does the activity need to be done in every case)?*

6. *For each development phase, identify and describe further (miscellaneous) activities that exist in this phase according to the questions listed above.*

To extract this information, consider the extracted architectural elements, the architectural element types, the flexibility classes, and the development strategy or process. If you cannot find the required information here, feel free to interview an SPL expert.

Output. (List of) development activities.

Rationales. The reason for asking these questions is the need to understand how the product line architecture and its supported flexibility classes influence the derivation of customer-specific systems during AE. Thus, the questions aim at providing information about the roles and their tasks within the development process. This is a prerequisite to elaborating corresponding information needs in the next step. However, as many steps that are determined by the aforementioned algorithm are often not explicitly documented in software organizations, the involvement of development experts is recommended here.

Tool Support. In contrast to most of the previous steps, in this tailoring step a higher degree of automation support is achieved (see Figure 48). Before displaying the instructions and forms for this tailoring step, the tool automatically generates a proposal for all possible development ac-

tivities based on the architectural element types, the product line architectural elements, and the flexibility classes identified before. Thus, instructions 1-4 of this tailoring step are performed automatically.

For each optional architectural element, the tool therefore generates a development activity using the following template:

Name: *Include <optional architectural element name>*

Purpose: *to add the <optional architectural element name>, if required.*

For each alternative architectural element, the tool generates a development activity as follows:

Name: *Instantiate <alternative architectural element name>*

Purpose: *to instantiate the <alternative architectural element name> (i.e., to configure the best fitting realization).*

For each product line architectural element with which a modification class is concerned, the tool then generates a development activity according to the following template:

Name: *Develop new realization of <product line architectural element name>*

Purpose: *to develop a new realization of the <product line architectural element name> if the existing ones are not sufficient to meet customer needs.*

In contrast to the other development activities, these development activities are set to "optional" by default because these activities only need to be carried out when the customer requirements cannot be fulfilled with existing variants. Thus, the information needs resulting from these activities also do not need to be satisfied in each case. This leads to a dynamic adaptation of the ARE process at runtime.

Finally, for each architectural element type with which an extension class is concerned, the tool therefore generates a development activity using the following template:

Name: *Develop new <architectural element type name>s*

Purpose: *to add new <architectural element type> elements and corresponding realizations, if required.*

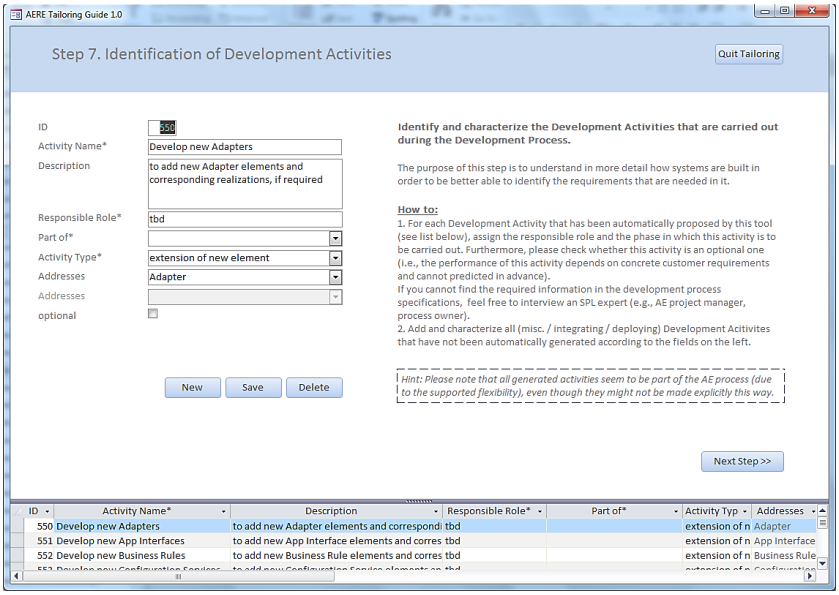


Figure 48. Screenshot of tailoring tool with example (step 7)

Thus, when applying this tailoring step by using the tool support, method tailors only need to enter the responsible roles and the development phases to which the activities belong. Of course, the automatically generated list of development activities should first be checked to see whether it reflects the actual development process correctly. At this point, it has to be kept in mind that in most development processes the activities are not modeled in such a fine-grained way, even though they all exist.

6.3.8 Elaboration of Decisions and Information Needs

Purpose. In this step, the concrete decisions to be made during the development of a customer-specific system and the corresponding information needs are elaborated. The purpose of this step is to understand the information that is required by the responsible roles regarding certain issues in order to perform their development activities. Thus, for each development activity identified before, it is determined which concrete decisions are made during this activity when producing the corresponding outcome. Then, for each decision, the information that must exist for this decision to be made deterministically is elicited from the decision-making role. This is a prerequisite to determining the set of relevant issues that have to be actually covered during an ARE process later on. In particular, all issues with which no information need is concerned can be left out. The same holds true for details (e.g., attributes) of an issue. However, the determination of the issues to be discussed and their

concrete details is not part of this step. This step just elaborates the information needs without consolidating them yet.

Foundation. The foundation of this tailoring step are the classes "Decision" and "Information Need" within the underlying model introduced in chapter 2 (see Figure 49). According to this model, one or more decisions are made during a development activity (and indirectly during the corresponding development phase) by the responsible role (decision maker). Decisions are either how-decisions, whether-decisions, or which-decisions, depending on the architectural element or architectural element type with which they are concerned. All decisions have in common that they cause information needs that must be satisfied in order to make these decisions in a deterministic way. Similar to assumptions, information needs are therefore concerned with (relevant) issues. This means that information about these issues is required for making decisions during AE.

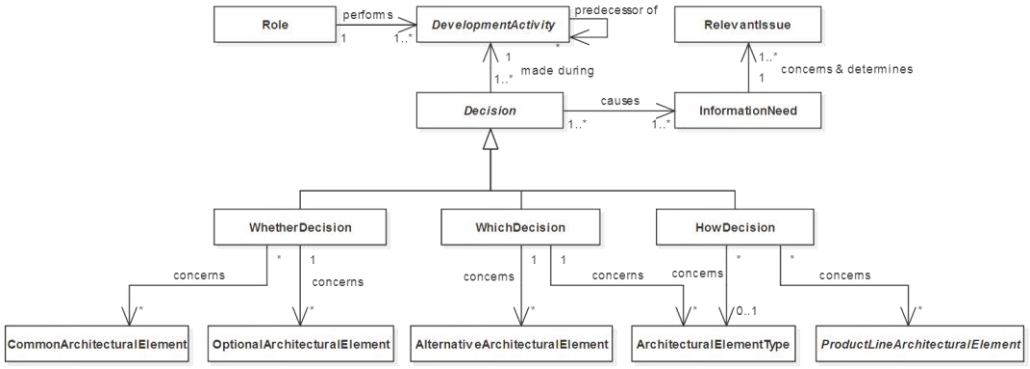


Figure 49. Foundation of tailoring step 8

Input. (List of) development activities, reference issue model.

Procedure. In order to elaborate the decisions to be made during the development of customer-specific systems and the corresponding information needs that must be satisfied by ARE, the following procedure has to be applied:

Tailoring Step 8 – Analyze the decisions that are made during the development activities as well as the corresponding information needs.

1. *For each inclusion activity identified before, determine the corresponding whether-decision concerning the affected optional architectural element.*

2. *For each instantiation activity identified before, determine the corresponding which-decision concerning the affected alternative architectural element.*

3. *For each re-development activity identified before, determine the corresponding how-decision concerning the affected product line architectural element.*

4. *For each re-development activity identified before that deals with the re-development of a common architectural element, determine the corresponding whether-decision concerning this common architectural element.*

5. *For each extension activity identified before, determine the corresponding how-decision and the which-decision concerning the affected architectural element type.*

6. *For each miscellaneous activity identified before, determine the corresponding decisions to be made in this activity.*

7. *Describe all elaborated decisions according to the following questions:*

   - *Which question is to be answered by the decision?*

   - *Which role is making the decision?*

8. *For each elaborated decision, determine the information needs that must be satisfied in order to be able to make the decision deterministically.*

9. *Describe all elaborated information needs according to the following questions:*

   - *What is the concrete information need (question to be answered)?*

   - *With which issue is the information need concerned?*

   - *What is to be known about the issue (details, attributes, …)?*

To perform this step, interview the roles that are responsible for the extracted development activities. When assigning the information needs to a certain issue, also consider the issue model introduced in chapter 5. If you cannot find the required information there, feel free to interview an SPL expert.

Output. (List of) information needs.

Rationales. The reason for asking these questions is the need to understand which information is needed in the development process in order to derive a customer-specific system from the given SPL. Without this information, an ARE elicitation instruction cannot be tailored, as it is not clear about which issues a discussion with the stakeholder is needed. However, as it is typically not appropriate to just interview the development roles which information they need, the decisions to be made by them must be identified first. The underlying assumption is that development roles can list their concrete information needs better and especially more completely when they imagine a concrete development situa-

tion and the decisions they have to make there. Thus, the elaboration of information needs is strongly related to an analysis of the affecting decisions during development.

Tool Support. As decision-making is a highly human-based process, we currently see no opportunity to provide a higher degree of formalism as described above. Nevertheless, this tailoring step is well supported by the tailoring tool (see Figure 50). Hence, before displaying the instructions and forms for this tailoring step, the tool automatically generates a proposal for all possible decisions based on the aforementioned development activities. In particular, instructions 1-5 (and for most parts also instruction 7) are completely automated based on the results of previous tailoring steps.

Thus, for each inclusion activity, the tool generates a whether-decision using the following template:

Question: *Do I have to include the <name of optional architectural element related to the inclusion activity> or not?*

For each instantiation activity, the tool generates a which-decision using the following template:

Question: *Which existing realization of the <name of variable architectural element related to the instantiation activity> should I take?*

Furthermore, for each redevelopment activity, the tool generates a how-decision using the following template:

Question: *How do I have to develop the customer-specific realization of the <name of product line architectural element related to the re-development activity>?*

If a re-development activity is concerned with a common architectural element, the tool additionally generates a whether-decision as follows:

Question: *Do I have to replace the common <name of common architectural element related to the re-development activity> with a customer-specific realization?*

Finally, for each extension activity, the tool generates both a which-decision and a how-decision using the following templates:

Question: *Which additional, customer-specific <name of architectural element type related to the extension activity>s do I have to develop?*

Question: *How do I have to develop the customer-specific <name of architectural element type related to the extension activity>s?*

In this regard, the tool also automatically sets the roles that are responsible for decisions, using the roles that are responsible for the development activities from which the decisions are derived.

Thus, when applying this tailoring step by using the tool support, method tailors can focus on the elaboration of information needs. In this regard, the generated list of decisions, which only needs to be extended manually for the decisions to be made in the miscellaneous activities, provides an instrument for guiding the information need elicitation systematically.

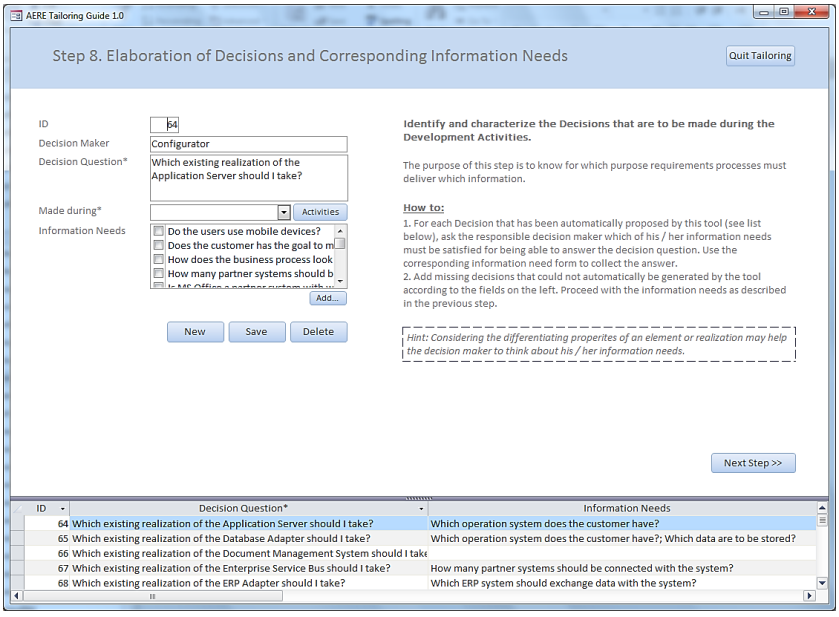


Figure 50. Screenshot of tailoring tool with example (step 8)

6.3.9 Determination of Relevant Issues

Purpose. In this step, the issues that have to be discussed during ARE in order to satisfy all information needs of the development roles are determined. The purpose of this step is to understand for which issues requirements have to be elicited during ARE, until which milestone, and with the consideration of which assumptions and existing SPL capabilities. Thus, this step consolidates all the relevant SPL information extracted in the previous tailoring steps, and extends this information with additional knowledge either from best practices or further SPL assets. For instance, the stakeholder roles to be involved when discussing a certain issue, or suitable elicitation and specification techniques are extended. Furthermore, information about reusable requirements concerning the

relevant issues is added (if existing) in order to make these assets directly accessible during the ARE process.

Foundation. The foundation of this tailoring step is the class "Relevant Issue" within the underlying model introduced in chapter 2 (see Figure 51). According to this model, relevant issues are determined via the information needs in the AE development process and must be discussed before the start of the first development phase in which these information needs exist. Relevant issues are to be discussed with certain stakeholders, and to be described with certain notations. The SPL specification may already contain reusable requirements concerning a relevant issue (explicitly anticipated requirements). However, in order to address non-explicitly anticipated (relevant) requirements as well the aforementioned assumptions that are concerned with the relevant issues must hold in order to satisfy these requirements with the calculated costs.

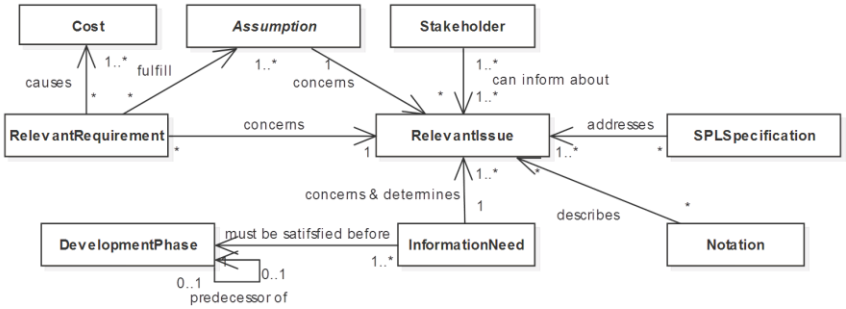


Figure 51. Foundation of tailoring step 9

Input. (List of) information needs, reference issue model, specification.

Procedure. In order to determine the relevant issues to be discussed during ARE as well as additional information concerned with them, the following procedure has to be applied:

Tailoring Step 9 – Analyze the issues with which the information needs are concerned.

1. *Determine the issues with which the information needs identified before are concerned. Remove duplicates, if necessary.*

2. *Describe all elaborated issues according to the following questions:*

   - *What exactly does the issue express?*
   - *Which details need to be known about this issue?*
     To answer this question, consider the details required in the identified information needs concerned with the issue.

- *What are the costs for realizing specific requirements concerning the issue?*
  To answer this question, consider the flexibility classes whose assumptions are concerned with the issue. Take the costs of the flexibility class with the highest costs.

- *Until the beginning of which development phase must the issue be discussed?*
  To answer this question, consider the development activities in which decisions are made and whose information needs are concerned with the issue. Take the development phase of the earliest development activity that leads to an information need concerned with the issue.

- *Which degree of freedom is provided by the SPL for requirements concerning the issue (only commonalities, only predefined variants, only specific requirements, predefined variants and specific requirements, commonalities and specific requirements, commonalities and predefined variants, commonalities and predefined variants and specific requirements)?*
  To answer this question, consider the SPL specification and analyze whether there are reusable requirements concerning the issues described in it. If the answer is yes, analyze whether these requirements are common or variable within the SPL and select a suitable degree of freedom.

- *Do requirements concerning the issue need to be documented in a system specification during ARE?*
  To answer this question, just check whether there are information needs concerned with the issue. If the answer is yes: mark "to be documented".

- *Which stakeholders can provide information about requirements concerning the issue?*

- *Which technique / notation should be used to additionally clarify the requirements concerning the issue?*

- *What is the conceptual type of the issue (normal class, singleton, abstraction)?*

3. *Recursively determine the issues that are needed to elicit requirements concerning the aforementioned issues. To do so, include the requiring, influencing, or containing issues of the already identified issue in the issue model, as well as the issues of which the already identified issues are a specialization. Answer the aforementioned questions for each identified issue.*

To perform this step, consider especially the extracted information needs as well as the issue model introduced in chapter 5 and the SPL specification. If you cannot find the required information there, feel free to interview an SPL or RE expert.

Output. (List of) relevant issues.

Rationales. The reason for asking these questions is the need to consolidate and concentrate all elaborated information about the SPL. As issues are the drivers for the ARE process and are the things to be developed with the stakeholders, all important SPL information has to be aligned with them.

Tool Support. As this tailoring step is quite tedious, but highly relies on already extracted information, it is automated to a very high degree (see Figure 52). Hence, before displaying the instructions and forms for this tailoring step, the tool automatically generates a proposal for all relevant issues based on the results of the previous tailoring steps and the issue model introduced in chapter 5. To make this happen, the tool first gathers all issues mentioned during the previous steps, and then enhances the corresponding issue descriptions with information provided by the issue model or previous steps. Thus, all instructions of this tailoring step are – as far as the issues are already part of the issue model – automatically executed.

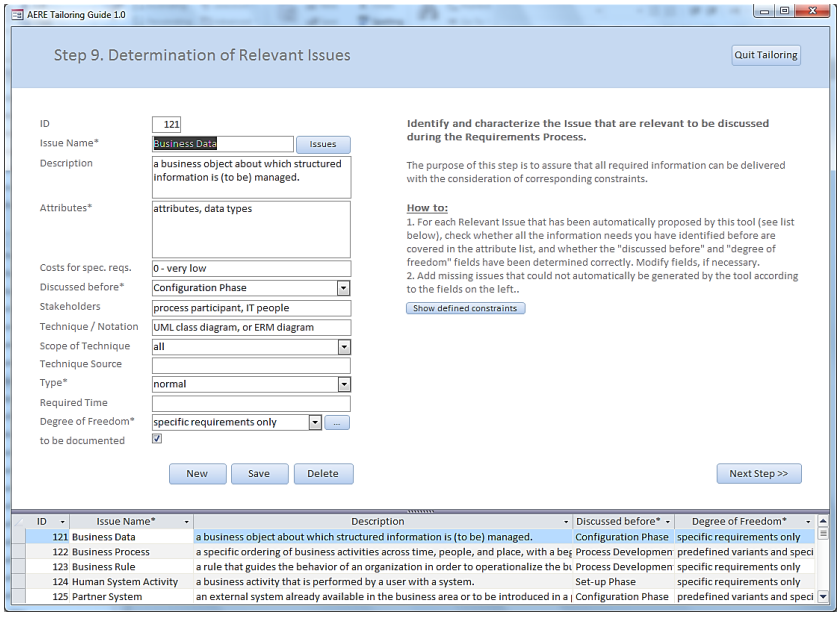


Figure 52. Screenshot of tailoring tool with example (step 9)

Thus, when applying this tailoring step by using the tool support, method tailors can focus on the elaboration of issues that are not covered in the issue model so far (which should be an exceptional case). Furthermore, only the degree of freedom has to be adapted manually, as an automatic analysis of the SPL specification is not part of this thesis. Finally, the method tailors should check the techniques and stakeholders to be

involved, as these entries are only recommendations copied from the issue model, and should add additional information, e.g., references to certain technique descriptions or time estimations for elicitation sessions.

6.3.10 Determination of Conceptual Relationships

Purpose. In this step, conceptual relationships that exist between the relevant issues are identified. The purpose of this step is to understand how the issues to be discussed during ARE depend on each other in order to define a meaningful elicitation sequence. Thus, this step deals with clarifying whether there are require, contain, influence, or specialize relationships between the aforementioned issues. In this regard, conflicting relationships that may lead to deadlock situations during ARE are identified and resolved, if necessary. To make this work efficient, best practices should be applied in this step.

Foundation. The foundation of this tailoring step is the class "Relationship" within the underlying model introduced in chapter 2 (see Figure 53). According to this model, each relevant issue may have a relationship with another relevant issue that is either a contain, influence, require, or specialize relationship. The relationships, however, do not depend on the given SPL but are rather inherent in the real world.

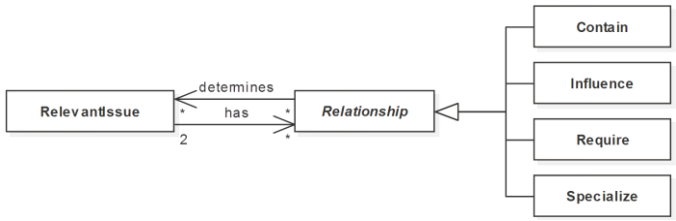


Figure 53. Foundation of tailoring step 10

Input. (List of) relevant issues, reference issue model.

Procedure. In order to determine the conceptual relationships among the relevant issues, which is a prerequisite to defining a meaningful elicitation sequence, the following procedure has to be applied:

Tailoring Step 10 – Analyze the relationships that exist between relevant issues.

1. *Identify the relationships that exist between each pair of issues identified before.*

2. *Describe all elaborated relationships according to the following questions:*

- *What is the multiplicity of the first issue within the relationship?*
- *What is the name of the relationship?*
- *What is the multiplicity of the second issue within the relationship?*
- *What is the type of the relationship (contain, require, specialize, influence)?*

3. *Check whether there are conflicts between the relationships and the order of the development phases to which the issues are assigned.*
   To do so, look for relationships in which the containing, influencing, requiring, or specialized issue is assigned to a later development phase than the contained, influenced, required, or specializing issue in this relationship.

4. *Resolve all identified conflicts.*
   To do so, either delete the relationship or reassign an issue in a conflicting relationship to the same development phase as its related issue.

To perform this step, consider the issue model introduced in chapter 5 as well as the set of relevant issues determined before. If you cannot find the required information here, feel free to interview an SPL or RE expert.

Output. (List of) conceptual relationships.

Rationales. The reason for asking these questions is the need to understand how the issues to be discussed during ARE are related. Without this information, neither a meaningful elicitation sequence nor clear elicitation instructions can be defined. Particularly as we aim at eliciting requirements in the context of their related requirements (see chapter 4), knowing these conceptual dependencies is a prerequisite.

Tool Support. As a pairwise check of issues to identify potential relationships is tedious work, we aim at reusing RE best practices formalized in the issue model here (see chapter 5). The tailoring tool therefore automates the determination of conceptual relationships to a very high degree (see Figure 54). In particular, before displaying the instructions and forms for this tailoring step, the tool automatically copies all relationships that exist between the issues identified before from the issue model. Furthermore, the tool automatically detects the aforementioned conflicts and displays them on screen.

Thus, when applying this tailoring step by using the tool support, method tailors can focus on the elaboration of relationships between issues that are not covered in the issue model so far (which should be an exceptional case). Otherwise, only the resolution of conflicts has to be done manually, as this requires clear trade-off decisions that cannot be made by the tool.

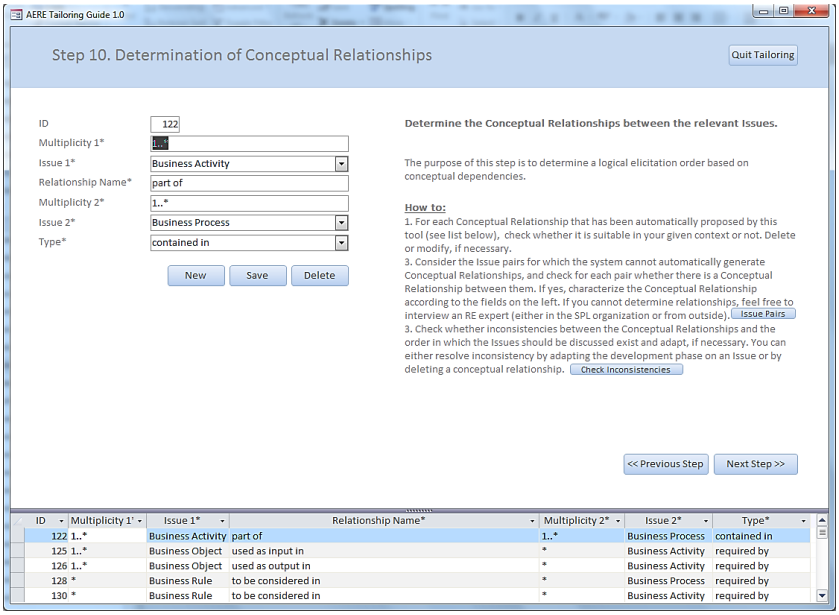


Figure 54. Screenshot of tailoring tool with example (step 10)

6.3.11 Definition of ARE Elicitation Instructions

Purpose. In this step, ARE elicitation instructions are created based on the identified and relevant issues, their relationships, the assumptions concerned with them, as well as the development phases for which they are important. The purpose of this step is to reflect important product and process knowledge about the given SPL and RE best practice in precise elicitation instructions. Thus, this steps deals with the incorporation of the extracted information into the ARE instruction template introduced in chapter 4.

Foundation. The foundation of this tailoring step is the class "Elicitation Instruction" within the underlying model introduced in chapter 2 (see Figure 55). According to this model, an elicitation instruction is composed of milestone sections that contain issue sections, which are composed of instruction blocks containing phrases.

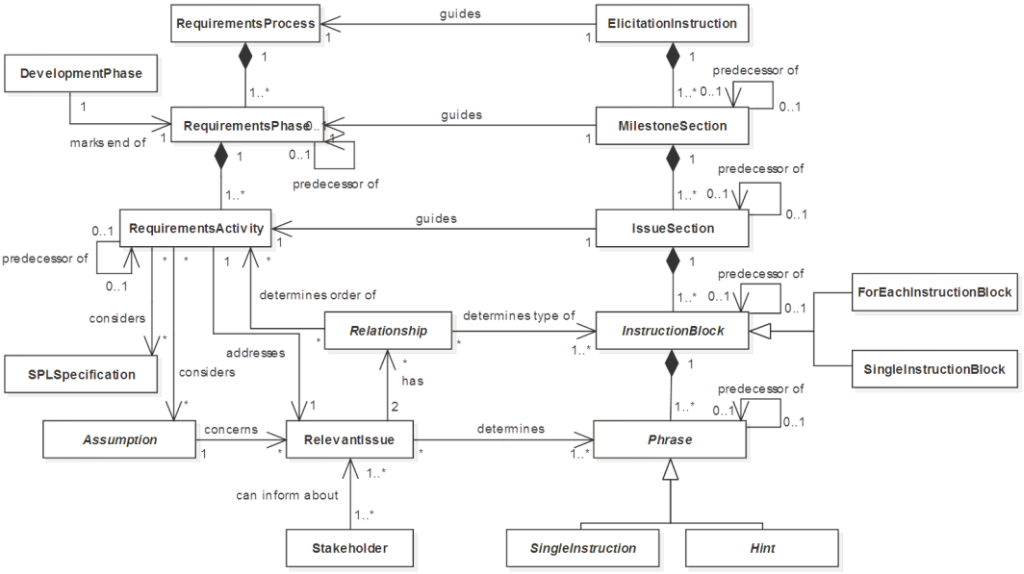


Figure 55. Foundation of tailoring step 11

Input. ARE instruction template, SPL characterization, (list of) assumptions, (list of) relevant issues, (list of) conceptual relationships.

Procedure. In order to generate a precise elicitation instruction based on all the information extracted and consolidated before, the following procedure has to be applied:

Tailoring Step 11 – Create an elicitation instruction document that guides the requirements process.

1. *Create a document in which the elicitation instructions should be described.*
2. *Create an overview description about the SPL and the systems to be derived from it. To do so, use the information extracted in the first tailoring step.*
3. *Exclude all issues that should not be addressed in the elicitation instructions. To do so, exclude the issues that are a) singletons and not to be documented, b) abstract without having a relationship (except for being specialized), c) not to be documented, but being specialized without having other relationships.*
4. *Identify all issues that are optional. To do so, identify all issues that are only caused by an information need that only depends on a development activity that is optional.*
5. *Bring the development phases identified before into a sequential order by considering the predecessor relationships.*

6. *For each development phase identified before:*

    a. *Create a milestone section and include a description of the purpose of this phase.*

    b. *Filter those issues that are to be discussed before this development phase starts.*

    c. *For each filtered issue that has no relationship to another issue, create an issue section.*

    d. *For each filtered issue that is not required by, not contained in, not influenced by, and not a specialization of another issue, create an issue section. If there is none: Create an issue section at least for the issues that are influenced by an already discussed issue, but that have no further required / contained / influenced / specialization relationships.*

    e. *For each filtered issue that is required by, contained in, influenced by, or a specialization of an issue already included in the elicitation instruction, and that is neither required by, contained in, influenced by, nor a specialization of an issue that has not been included yet, create an issue section. If there is more than one, create the issue sections in the following order:*

        i. *Issue sections for issues that specialize an already included issue*

        ii. *Issue sections for issues that are contained in an already included issue*

        iii. *Issue sections for issues that are required by an already included issue*

        iv. *Issue sections for issues that are influenced by an already included issue.*

        If there is more than one issue in each sub-order, create the corresponding issues section in the order in which the specialized / containing / requiring / influencing issue has appeared.

    f. *Go back to (5e) until all issues related to the development phase have been incorporated into the elicitation instruction document in terms of issue sections.*

7. *For each issue section incorporated in the elicitation instruction document:*

    a. *Include a description of the corresponding issue.*

    b. *Include an involvement hint that names the stakeholders to be in-*

volved when discussing requirements concerning the issue.

c. *If there are already common requirements concerning the issue in the SPL, include a commonality hint that informs about this fact.*

d. *If the issue is not abstract and is influenced by other issues, include an influence hint that informs about other requirements to consider when defining requirements concerning the issue.*

e. *Identify related issues that contain or require the issue.*

f. *For each containing or requirement issue, include an identifying and collecting instruction in order to elicit requirements concerning the issue based on other requirements.*

g. *Include an identifying and collecting instruction in order to elicit further requirements concerning the issue.*

h. *If an issue has a contain-relationship to itself, include a decomposition instruction.*

i. *If an issue has a require-relationship to itself, include an identifying and collection instruction.*

j. *If a technique / notation is defined for the issue, include a visualization hint that informs about how to clarify the requirements concerning the issue in a graphical manner.*

k. *If assumptions are concerned with the issue, include an assumption hint that informs about which assumptions must hold in order to not contravene the SPL.*

l. *If there are already predefined variable requirements for the issue in the SPL, include a selection hint that informs about this fact.*

m. *If requirements concerning the issue should not be documented, include a documentation hint that informs about this fact.*

n. *If the issue is optional, include the conditions under which the requirements concerning the issue have to be documented. To do so, copy the conditions from the optional development activities that have caused the optionality of the issue.*

o. *If the issue is to be documented and not abstract, include a describing instruction in order to assure that corresponding requirements are specified.*

p. *If the issue is to be documented and abstract, include a classifying instruction in order to assure that corresponding requirements are processed correctly.*

q. *If the reuse of requirements concerning the issue is mandatory, include a selecting instruction that assures this reuse.*

To perform this step, consider the tailoring results of the previous steps as well as the ARE instruction template introduced in chapter 4. If you cannot find the required information here, feel free to interview an SPL or RE expert.

Output. ARE instructions document.

Rationales. In this final tailoring step, no further information has to be extracted from the SPL and no decisions have to be made. Rather, all the information gathered so far in the previous tailoring steps is algorithmically processed and translated into ARE elicitation instructions using the text blocks and template introduced in chapter 4. Hence, the reason why the aforementioned procedure is as presented, is based on the rules defined in this template.

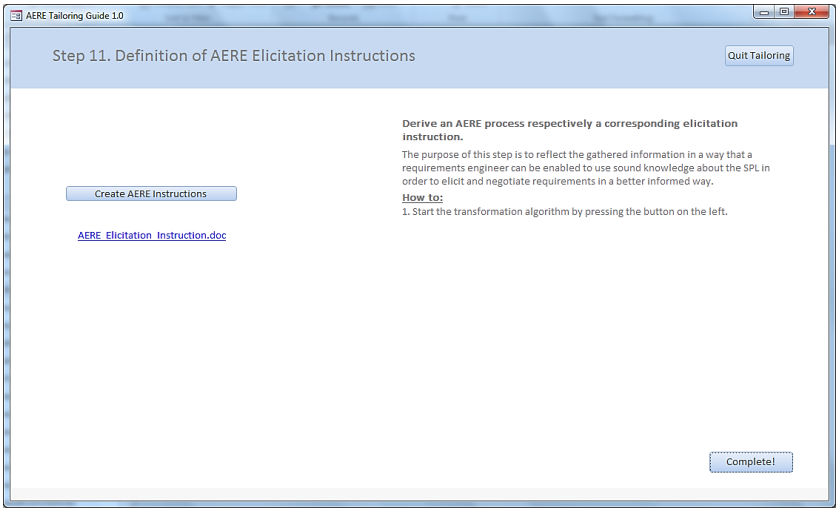


Figure 56. Screenshot of tailoring tool with example (step 11)

Tool Support. While the tool support in the other tailoring steps was limited to some sub-steps, this final step is automated completely (see Figure 56). Thus, when applying this tailoring step by using the tool support, method tailors only need to start the transformation algorithm. An ARE instructions document is then generated within a few seconds. The source code for the document generation is shown in Appendix C.

6.4 Summary

In order to make AE requirements engineers aware of the capabilities and constraints of a given SPL, corresponding process and product

knowledge must be extracted and incorporated into the ARE process, respectively supporting instructions.

In this chapter, a systematic approach for such an incorporation of SPL knowledge into an ARE process has therefore been introduced. This tailoring approach prescribes eleven steps that should be performed sequentially in order to extract SPL knowledge in a systematic and repeatable manner. In this regard, the approach explains how knowledge about the product line architecture, the development strategy for AE, as well as RE best practices has to be combined in order to provide AE requirements engineers with precise and helpful elicitation instructions. For this purpose, the tailoring approach makes use of the conceptual ARE model introduced in chapter 2 and explains how the elements of this formal model have to be processed.

However, even though we systematically derived the tailoring steps from this model and even though we were able to provide very precise and algorithmic guidance for these steps, the degree of automation is still limited. The reason for this is not a missing formalization, but the fact that a fully automated extraction of SPL knowledge would require a semantic description of all SPL artifacts such as the product line architecture or the development strategy. Both in practice and in academia, this precondition is not fulfilled and will probably not be for a long time. Thus, most knowledge about a given SPL can only be extracted from informal documents or SPL experts in a human-based way. Nevertheless, we consider the proposed tailoring approach as a valuable computer science contribution, as it formalizes this procedure in a way that allows an algorithmic (even though not completely automated) performance. In particular, all tailoring tasks that rely only on information gathered in a previous tailoring step are automated in the current version of the tailoring tool. Thus, the more tailoring progresses, the higher the degree of automation in the remaining tailoring steps is.

With regard to the state of the art in product line engineering, it is important to note that the tailoring approach does not aim at replacing any established practices there. Rather, this tailoring approach is intended to enhance the externalization of knowledge about an SPL after this SPL has been built. Thus, the tailoring approach should be integrated at a very late phase in DE/FE, i.e., when many design decisions and implementations have been done. In particular, the tailoring approach requires a stable product line architecture, a clear development strategy and technology decisions as input. Thus, doing the tailoring during an early phase, such as during scoping or domain analysis, is not possible.

The tailoring approach therefore aims at externalizing the knowledge about the capabilities and constraints of an SPL beyond the explicitly anticipated scope defined during the early scoping and domain analysis phase. While this is probably not needed in configurable SPLs, it is an indispensable means for benefiting from an SPL approach in domains that

require a significant degree of flexibility. As this is mainly the case in IS, the RE best practices used during tailoring have to be defined for this context. In the previous chapters, we have therefore introduced the RE best practices to be considered there.

7 Evaluation

"It is meaningless to say: We do our best.
We must be able to do what is necessary."
Winston Churchill

This chapter describes the evaluation of the thesis approach. Thus, the purpose of this chapter is to investigate the benefits and limitations when using the approach in order to enable practitioners and scientists to better estimate the potential improvements when using the approach in their context.

7.1 Research Approach

As research always involves gaining a deep understanding of the effects of a solution [Bas93], the approach for the final evaluation was closely aligned with the overall research process described in section 1.4. Thus, based on the elaborated problem statement, goals describing the intended benefits from a practical and a scientific point of view were defined in the first step.

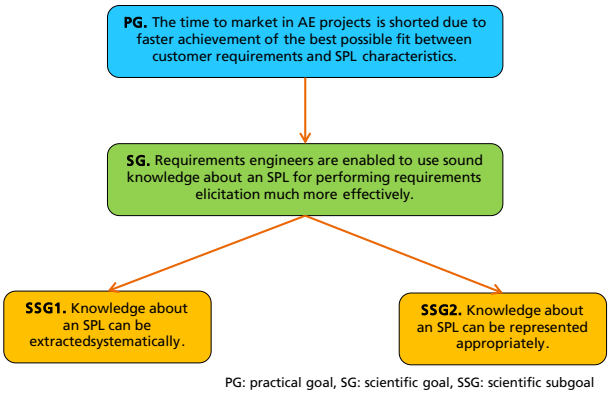


Figure 57. Goal tree of the thesis contributions

As depicted in Figure 57, the practical and overall aim of this thesis was to shorten the time to market in AE projects by achieving the best possi-

ble fit between customer requirements and SPL characteristics (PG) faster. Based on this, the derived scientific goal dealt with enabling requirements engineers to use sound knowledge about an SPL for performing requirements elicitation much more effectively (SG). As two scientific sub-goals in this regard, SPL knowledge was to be extracted systematically from a given SPL (SSG1), and to be represented appropriately to requirements engineers during AE (SSG2).

For each of these four goals, hypotheses were defined that expressed the quantifiable benefits that this thesis intended to achieve (step 1). These hypotheses are (as shown in chapter 1 already):

Hypothesis 1 – Efficiency of Application Engineering (addresses PG)

H_1. *An AE process using ARE instructions defined on the basis of the thesis approach has an at least 15% shorter time to market than an AE process using state of the art instructions.*

Hypothesis 2 – Effectiveness of Elicitation (addresses SG)

H_2. *ARE instructions defined on the basis of the thesis approach enable requirements engineers to achieve an at least 15% higher realization fit during an elicitation session than when using other instructions.*

Hypothesis 3 – Effectiveness of Tailoring (addresses SSG1)

H_3. *An incorporation of SPL knowledge into ARE instructions is possible when using the thesis approach, i.e., at least 80% of method tailors are able to successfully create ARE instructions without major problems.*

Hypothesis 4 – Suitability of Representation (addresses SSG2)

H_4. ARE instructions defined on the basis of the thesis approach suitably represent SPL-related product and process knowledge.

For each hypothesis, it was then decided whether this hypothesis should be evaluated explicitly in a final study, evaluated implicitly in immediate feedback loops during method development, or not be evaluated at all during this thesis research (step 2). In this regard, we also determined which study type (controlled experiment vs. case study vs. survey) should be applied for investigating each hypothesis.

According to this analysis, the effectiveness of elicitation (H_2) and the suitability of representation (H_4) were planned to be evaluated in a joint controlled experiment with an integrated survey (see section 7.2). The reason was that these two hypotheses were the only ones that could be evaluated in a controlled setting. In particular, a single case study, at least for H_2, would not have been sufficient for providing convincing evidence, as there would have been too many threats to validity.

However, for evaluating the effectiveness of tailoring (H_3), we chose another approach. As we had checked the basic feasibility and (semi-) automation of the tailoring approach during method development already (see chapter 6), we were interest to see whether the tailoring approach is applicable by practitioners in industry rather than investigating certain effects in a controlled setting. Thus, the corresponding hypothesis was planned to be evaluated by a case study (see section 7.3).

The hypothesis regarding an improved AE efficiency (H_1) was not evaluated at all as part of the thesis research. On the one hand the external validity of such a controlled experiment would have been too to provide convincing conclusions. On the other hand, performing a case study would have required too much time, as real AE projects often take months or even years. We will therefore deal with this in the future.

Thus, a controlled experiment and a case study were carried out for the purpose of validating the thesis contributions (step 5). For both studies, sub-hypotheses and metrics were derived, and the material and setup were prepared carefully (step 3). By discussing and improving the study designs with experimentation experts and RE experts in several iterations, a proper evaluation was constructively assured (step 4). Thus, the studies are expected to have sufficient quality to challenge their results. The entire research approach for preparing, executing, and analyzing the studies is shown in Figure 58.

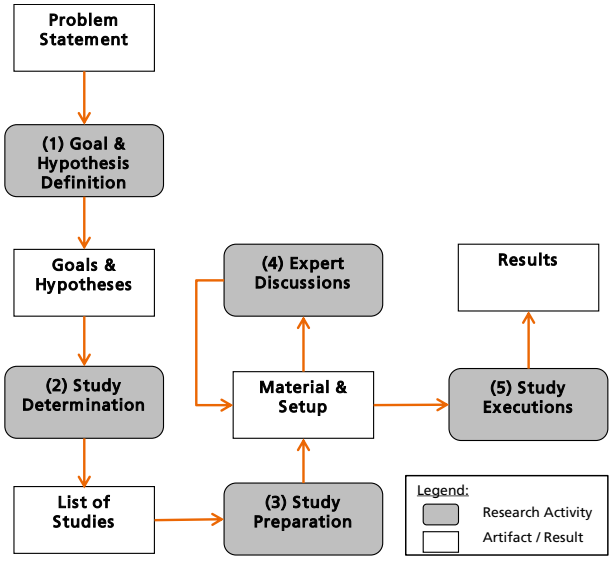


Figure 58. Research approach for empirical studies

7.2 Controlled Experiment

In this section, a controlled experiment for evaluating the effectiveness of elicitation (H_2) as well as the suitability of representation (H_4) is presented. Besides the goals and hypotheses of the study, its design and setup, its results and threats to validity, as well as its implications for research and practice are described.

7.2.1 Goals and Hypotheses

According to the GQM approach [BCR94], the main goal of this experiment was to analyze two elicitation approaches for the purpose of comparison with regard to elicitation effectiveness from the viewpoint of requirements engineers in the context of a controlled experiment with students. Thus, the goal was to know which approach allows performing more effective elicitation when using a given SPL. Below, we introduce the underlying research questions and one-side hypotheses of the experiment, as well as their related metrics (in brackets). However, we omit the corresponding null hypotheses, as they just state the opposite.

All hypotheses are based on the main research question $RQ_{2.M}$, which asks whether requirements elicitation in ARE is more effective when using ARE instructions according to this thesis than when using a traditional ARE approach because this was the underlying hypothesis for the entire experiment. As elicitation is basically the process of communicating with stakeholders to determine requirements [CA07], this research question was broken down into two sub-questions and related hypotheses.

$RQ_{2.1}$: *"Does our ARE approach enable requirements engineers to communicate more effectively with stakeholders than when using the traditional ARE approach?"*

As effective communication includes the exchange of relevant, complete, and correct information, we expect that requirements engineers using our ARE approach will:

- $H_{2.11}$ ask for more relevant information (#asked relevant questions)
- $H_{2.12}$ ask for less irrelevant information (#asked irrelevant questions)
- $H_{2.13}$ provide more correct information to stakeholders (# correctly answered stakeholder questions / # posed stakeholder questions)

- $H_{2.14}$ need less SPL expert involvement (# elicited requirements marked as "to be checked by SPL experts")

than requirements engineers using the instructions of the traditional ARE approach. Thus, these hypotheses deal with a bidirectional exchange of information, which is indispensable for eliciting and discussing requirements in a competent way. This ultimately allows stakeholders to feel better understood and more convinced, especially when requirements have to be negotiated.

$RQ_{2.2}$: *"Does our ARE approach enable requirements engineers to achieve better results (i.e., requirements) than when using the traditional ARE approach?"*

According to the taxonomy of elicited requirements described in section 2.6.3, we expect in this regard that requirements engineers using instructions according to this thesis will:

- $H_{2.21}$ elicit fewer unnecessary requirements (# accepted common requirements / # initially stated common requirements)

- $H_{2.22}$ elicit fewer problematic requirements (# accepted problematic requirements / # initially stated problematic requirements)

- $H_{2.23}$ achieve a higher satisfaction fit (# accepted realizable requirements / # initially stated requirements)

- $H_{2.24}$ achieve a higher realization fit (# accepted realizable requirements / # accepted requirements)

than requirements engineers using traditional ARE.

Figure 59 summarizes the research questions and related hypotheses. In this regard, we consider the research questions to be answered with "yes" if at least one hypothesis on the corresponding leaf level is confirmed and the opposite of all other related sub-hypotheses (which state that the traditional ARE is better in a certain metric) is not confirmed.

Furthermore, as we would also check the hypothesis regarding the suitability of representation (H_4) during this experiment, an additional research question was:

$RQ_{4.1}$: *"Is the representation of ARE instructions defined according to the thesis approach perceived as being suitable to support elicitation?"*

Here, we expect that requirements engineers using instructions according to this thesis will:

- $H_{4.11}$ perceive the instructions as easy to read (# participants who share this perception / # participants)
- $H_{4.12}$ perceive the instructions as easy to use (# participants who share this perception / # participants)
- $H_{4.13}$ feel supported well in finding important information quickly (# participants who share this perception / # participants)

In this regard, we consider the research question to be answered with "yes" if the values of the three metrics are higher than 75% each. The reason is that a statistical test is not possible here, as we are just interested in investigating a non-comparative statement.

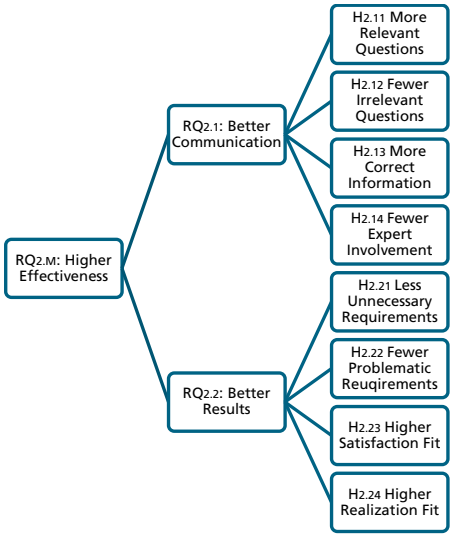


Figure 59. Questions and hypotheses in controlled experiment

7.2.2 Study Design and Setup

Based on the study goals, the hypotheses, and the related metrics, the controlled experiment was designed and prepared. Below, we describe the details of the experimental setup and how we constructively avoided as many threats to validity as possible.

7.2.2.1 Participants (Subjects)

The participants in the experiment were 26 (N_G=13) computer science / software engineering students (four female, 22 male) from the University of Kaiserslautern, Germany enrolled in the RE lecture in the winter term 2011 / 2012. Five of them were bachelor students, while the others were

enrolled in the master course. The participants were 23.9 years old on average, and participated voluntarily in the experiment. They were not informed about our hypotheses or study goal beforehand, nor were they informed about the experimental group to which they were assigned. Regarding their background, ten participants had gathered RE experience outside the RE lecture as well. However, only four participants had made more than three interviews for the purpose of elicitation already; the average was 0.8 interviews. Their self-assessed English competency on a scale of 1 (very good) to 5 (very bad) was good (1.8) on average, and only six participants mentioned that their knowledge of English was just medium (3). Regarding SPL engineering knowledge, 15 participants mentioned that they were aware of the basic concepts. However, no participant had had practical experience with SPLs before.

7.2.2.2 Experimental Design

In order to investigate the hypotheses H_2 and H_4 including their sub-hypotheses, the experiment was designed as a control group study (see Figure 60). The method group (MG) used a tailored ARE instructions document according to the thesis approach. In contrast, the control group (CG) used traditional ARE instructions according to a variability model (VM)-based ARE approach combined with a TORE [PK04], [ADE+09] guideline. This meant that for each explicitly anticipated variation point, the possible variants and their requirements were communicated to the stakeholder by using closed questions with predefined answer possibilities. In contrast, for non-explicitly anticipated, a "traditional" requirements elicitation without any consideration of SPL constraints took place from scratch. Thus, the control group was equipped with state of the art material, whereas the method group was equipped with our novel approach. In addition, the SPL specification in which the main SPL features and the product line architecture were described was also handed out to the participants of both groups.

During the experiment, both groups performed fictitious but controlled elicitation interviews in order to gather requirements. The participants' performance was measured using the aforementioned metrics in order to enable statistical comparison.

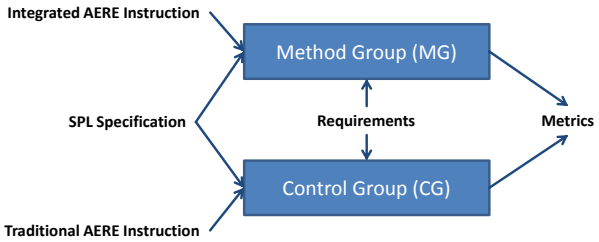


Figure 60. Overall setting of experiment

Each participant was randomly assigned to one group (see Table 4). As far as possible, we tried to assign the participants to the groups alternately based on their arrival to the experiment. This was the preferred strategy, as we did not know the participants before and considered their commitment to a certain time slot as random.

However, in order to assure equal distribution of RE-experienced and less RE-experienced participants in both groups (and to avoid corresponding threats to validity), their RE background was taken into consideration for finally making the assignment. Thus, the information about elicitation and interview experience provided in the pre-questionnaires was used for balancing the groups. A statistical test even showed that no significant difference regarding RE experience between the two groups could be confirmed. However, additional balancing of the participants according to their awareness of basic SPL concepts was not possible, as only six participants had both RE experience and SPL awareness at the time. In section 7.2.4, we discuss how this fact influenced the results.

	MG	CG
# Subjects	13	13
# Subjects in master course	11	10
Average age	23.3	24.5
Average # of interviews performed before	0.9	0.7
Average language competency	1.8	1.8
# more RE experienced subjects	6	4
# SPL-aware subjects	10	5

MG: method group, CG: control group

Table 4. Assignment of participants to groups

7.2.2.3 Material

For the execution of the experiment, as well as for data collection and analysis, different artifacts were used. Participants of both groups received a pre-questionnaire about their personal background, an SPL specification (ten pages long) of a fictitious SPL, study instructions, and a post-questionnaire. These were the same for both groups and did not allow drawing conclusions about the group to which the participants were assigned. The post-questionnaire contained "agreement" questions on a 5-point Likert scale and open questions to which free answers could be provided. The participants in the method group additionally got ARE instructions according to the thesis approach (seven pages long), while the participants in the control group received TORE-based elicitation instructions including a VM-based questionnaire (four pages long).

At this point, it has to be noted that the material provided to both groups artificially contained the same information about the underlying SPL, which was explicitly checked and approved by an independent expert. The material differed only in how this information was represented (i.e., directly integrated into the elicitation instructions vs. distributed in

in the SPL specification) and the strategy according to which the elicitation had to be done. In this context, "artificially" means that the material of the control group was supplemented with information typically not contained in such material. For instance, state of the art SPL specifications describe possible variants of a certain variation point as explicit enumerations or as a mathematical range, but contain typically no additional information on architectural constraints, for instance. Thus, the material of the control group was augmented by such information in order to make it more comparable.

The elicitation instructions of the control group were also made very precise in order to provide proper guidance to the (less experienced) participants in both groups. The reason for artificially adjusting the material was again to increase the groups' comparability in order to minimize corresponding threats to validity.

Besides the material for the participants, material was also needed for the fictitious stakeholder and the observer (see description of procedure above). Thus, based on a sample solution derived from the SPL specification, an observer checklist with 64 check criteria (e.g., *"Does the participant ask for user roles (yes / no)?"*, *"Does the participant give the right answers to stakeholder questions regarding additional costs (yes / no)?"*, etc.) was prepared. This checklist was aligned with a list of 27 prepared requirements and 12 interrupting questions (e.g., about additional costs) with which the fictitious stakeholder was prepared in order to standardize her behavior.

The material used in the experiment is included in Appendix E.

7.2.2.4 Experiment Procedure and Data Collection

In order to assure the success of the experiment, the entire design and setup material was carefully analyzed by both RE experts and empirical research experts at Fraunhofer IESE before the experiment was executed. From an empirical point of view, it was checked whether the setup was sound and whether important threats to validity were avoided constructively. From an RE point of view, it was checked whether the material of the control group was actually state of the art, not intentionally worse, and equal with regard to its information content. After incorporating the feedback received, a student assistant was coached in the list of prepared requirements and interruption questions, as she had to play the role of a stakeholder during the experiment. However, we took care not to inform her about the study design or about our hypotheses.

In order to check whether the material was understandable for the participants and the fictitious stakeholder and whether the scheduled timeframe was sufficient, we then performed a pilot run with two bachelor students. Thus, a pre-test with the method group's material

and one with the control group's material was carried out. The observations made during these tests were used to ultimately improve the material.

The actual experiment was conducted in November 2011 and was split into 26 single sessions of 90 minutes each. Thus, the participants performed the experiment individually, as it was neither meaningful nor organizationally possible to let multiple participants perform elicitation interviews in parallel. In order to avoid the threat of one participant informing other participants about the purpose, procedure, or material of the study, the participants had to sign a non-disclosure agreement that did not allow them to talk about the study with other people until the end of the entire experiment. Furthermore, we used a pre-questionnaire for getting data about their experience, and for assigning them to a suitable group.

After the group assignment, the participants received a package with the experiment material. Each participant then had about 25 minutes to become familiar with the material. In the second step, the participants were asked to answer questions about the content of the material in a questionnaire. The purpose of this step, which took around 10 minutes, was twofold: First, the participants were to intensively work with the material in order to note where to find certain types of information. Second, this task allowed us to objectively compare which of the used materials enabled requirements engineers to know more about a given SPL.

In the third step, the actual experiment took place. The participants were asked to perform an elicitation role-play in which the fictitious stakeholder, played by a student assistant, had to be interviewed in English using the provided material. English was chosen because 16 of the 26 students were not familiar with German, and because English was the language used in the entire RE lecture anyway. The participants interviewed the fictitious stakeholder by following the provided elicitation instructions and considering the SPL specification, if required. The fictitious stakeholder stated the corresponding requirements that had been prepared during study planning and that were aligned with an observer checklist (see Figure 61). In this regard, we took care that the student assistant who played the stakeholder was not aware of the group a participant was assigned to. Thus, we tried to minimize manipulation threats due to differences in her behavior.

Furthermore, the fictitious stakeholder posed interrupting questions to the participants; for instance, about the feasibility and costs of certain requirements. This was done in order to increase the representativeness of the study, as an interview in practice is never just a one-way communication. Thus, the purpose of these interruptions was to check whether the participants were able to provide sufficient information in order to make the elicitation more realistic and convincing. However, the prepared requirements and the interrupting questions depended on the be-

havior of the participants and were not posed anytime. For instance, interrupting questions regarding additional costs were only posed when a participant stated that there would be extra effort when implementing a certain requirement.

Figure 61. Impression from an experiment session

The interview questions asked by the participants, their reaction to the mentioned requirements, as well as the answers they gave to the stakeholder's questions were tracked by an observer (played by the thesis author) using the aligned observer checklist. The reason for using observations for data collection instead of the specified requirements was the fact that the elicitation performance and not specification performance was to be investigated. Furthermore, the likely heterogeneity in the quality of the specifications would have had such a high impact on the evaluation that no sound conclusion about the elicitation performance would have been possible when using this indirection. However, the threats to validity that resulted from this observation are discussed in section 7.2.4.

By using the observer checklist, the aforementioned metrics could be measured objectively. In this regard, the use of prepared requirements and a corresponding observer checklist aimed at making the results more comparable, which would otherwise have been a significant threat to validity. Furthermore, in order to avoid missing observations, the entire elicitation role-play was recorded using a voice recorder. The complete setting of this role-play, which took about 30 minutes in both groups, is shown in Figure 62.

In the last step of each study session, the participants were finally asked to fill out a post-questionnaire in which they assessed the helpfulness of the provided material for the interview.

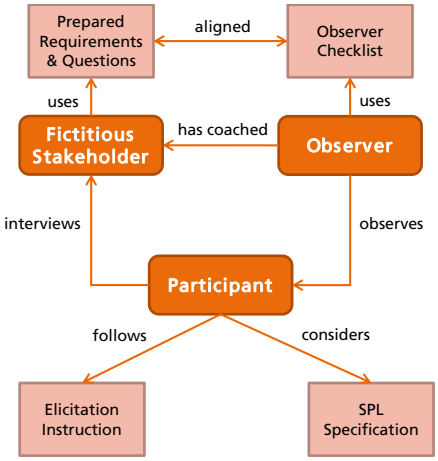


Figure 62. Detailed procedure and data collection

7.2.3 Analysis

Based on the measured data and the feedback gathered, a statistical analysis was performed The procedure and its results are described in this section.

7.2.3.1 Analysis Procedure

While data from the pre-questionnaire (e.g., age, interview experience, etc.) were entered directly into the statistical tool SPSS (version 18), the data gathered during observation (e.g., number of relevant questions asked, number of questions answered correctly, etc.) were recounted first in order to reduce the risk of measurement errors. Then, these data were also entered into SPSS. The subjective, quantitative ratings according to the Likert scale in the post-questionnaire were collected in MS Excel. The corresponding qualitative comments were listed in MS Word. In this regard, all data was processed in an anonymous and confidential way. For the quantitative observation data, descriptive statistics were calculated using the SPSS tool. Furthermore, the distributions were analyzed using the Kolmogorov-Smirnov test in SPSS. This was done in order to determine a suitable hypothesis test based on the different tests' assumptions regarding data distribution in an independent sample setting. For most hypotheses, we used the t-test for equality of means, except for those for which the Kolmogorov-Smirnov test had not confirmed equal distribution between the groups. As this was the case for $H_{2.12}$, $H_{2.13}$, and $H_{2.22}$, we used an independent samples median test instead. However, as the t-test is robust under violation of its assumptions [Zim87], we used this test also for variables that had non-parametric (but equal) distributions. An additional cross-check with the Mann-Whitney U

test confirmed that the application of the t-test was actually valid for calculating the same results. As all hypotheses were directional (e.g., "less," "higher", …), we used the one-tailed results with $\alpha=0.05$ in all hypothesis tests.

7.2.3.2 Objective Measurement Results

In this subsection, the observation results are described (see Table 5).

		Mean	SD	t	df	p	reject H_0
$H_{2.11}$: #asked relevant questions	MG	17.6	2.22	0.99	23	0.166	no
	CG	16.8	2.13				
$H_{2.12}$: #asked irrelevant questions	MG	0.77	0.93	-	24	0.002	yes
	CG	2.54	1.13				
$H_{2.13}$: # correctly answered stakeholder questions / # posed stakeholder questions	MG	0.77	0.20	-	24	0.001	yes
	CG	0.40	0.29				
$H_{2.14}$: # elicited requirements marked as "to be checked by SPL experts"	MG	1.01	1.12	-1.32	24	0.100	no
	CG	1.62	0.96				
$H_{2.21}$: # accepted common requirements / # initially stated common requirements	MG	0.62	0.36	1.42	24	0.084	no
	CG	0.81	0.33				
$H_{2.22}$: # accepted problematic requirements / # initially stated problematic requirements	MG	0.24	0.29	4.44	24	0.001	yes
	CG	0.70	0.25				
$H_{2.23}$: # accepted feasible requirements / # initially stated requirements	MG	0.64	0.06	1.52	24	0.071	no
	CG	0.59	0.08				
$H_{2.24}$: # accepted feasible requirements / # accepted requirements	MG	0.92	0.09	-	24	0.003	yes
	CG	0.83	0.07				

SD: standard deviation, t: t-value, df: degrees of freedom, p: probability

Table 5. Statistical results of experiment

Regarding the first hypothesis, which states that participants using ARE instructions according to this thesis are able to ask more relevant questions, we found that there seems to be no significant difference. While participants of the method group asked *17.6* relevant questions on average, the participants of the control group asked *16.8* relevant questions on average. With an effect size of $d=0.37$ and a significance of $p=0.166$, this hypothesis can therefore not be confirmed (i.e., the corresponding null hypothesis "no difference or fewer" cannot be rejected).

In contrast, the hypothesis that participants using the thesis ARE approach are able to ask fewer irrelevant questions could be confirmed. Here, a significant difference could be observed between the method group and the control group. With *d=1.71* and *p=0.002*, the corresponding null hypothesis "no difference or more" can be rejected. The test power was *~1.00* here.

In addition, the hypothesis that participants of the method group are able to provide more correct information about an SPL to stakeholders could be confirmed. While participants using our ARE instructions could answer *77%* of the posed stakeholder questions correctly, only *40%* of stakeholder questions were answered correctly by the control group participants. With *d=1.49* and *p=0.001*, the corresponding null hypothesis "no difference or fewer" can thus be rejected also. The test power was *0.98* in this case.

However, even though the method group participants were able to provide more correct information on their own, there is no significant difference to the required expert involvement. While participants using the thesis ARE approach marked *1.01* requirements as having to be checked by SPL experts, members of the control group asked for *1.62* expert checks. However, with α*=0.05*, this difference is not significant, and the effect size *d* is only *0.52*. Thus, the corresponding null hypothesis "no difference or more" cannot be rejected.

Hence, as the method group was significantly better with regard to two metrics and not worse in the other two metrics, we consider (according to Figure 59) ARE instructions according to this thesis as better means for communicating with stakeholders than the instructions of traditional ARE approaches.

Regarding the first hypothesis concerning "better results", which states that participants using our ARE instructions elicit fewer unnecessary requirements (i.e., requirements that are already implemented as commonalities within the SPL), we found that there seems to be no significant difference. Indeed, participants of the method group only accepted *62%* of the stated common requirements, while the members of the control group accepted *81%*. However, this difference is not significant with α*=0.05*. With *d=0.55* and *p=0.084*, this hypothesis can therefore not be confirmed (i.e., the corresponding null hypothesis "no difference or higher" cannot be rejected).

In contrast, the hypothesis that participants using the thesis ARE approach are able to elicit fewer problematic requirements could be confirmed. While the method group accepted *24%* of the stated problematic requirements on average, the control group accepted *70%* of the stated problematic requirements. With *d=1.73* and *p=0.001*, the corre-

sponding null hypothesis "no difference or more" can thus be rejected. The test power was ~*1.00* here.

With regard to the achievement of a higher satisfaction fit, there seems to be no difference between the groups. The satisfaction fit measures the degree to which a stakeholder's initially stated requirements are accepted and feasible. While *64%* of the initial stakeholder requirements were accepted in the method group, the participants of the control group accepted *59%* on average. Thus, with α=*0.05*, this difference is not significant, and the effect size is only *0.56*. The corresponding null hypothesis "no difference or lower" can therefore not be rejected.

Finally, with regard to the practical problem that motivated the thesis, the last hypothesis is maybe the most important one and could be confirmed with *p=0.003*, *d=1.12*, and a test power of *0.87*. In particular, participants using ARE instructions according to this thesis could achieve a *92%* realization fit, while members of the control group only achieved *83%* on average. This means that the degree of accepted requirements that are economically feasible with a given SPL is significantly higher when using the proposed approach than when using state of the art approaches. The corresponding null hypothesis "no difference or lower" can therefore be rejected.

Thus, as the method group is significantly better with regard to two metrics and, already from a descriptive point of view, not worse in the other metrics, we consider our ARE approach also more appropriate for achieving good results (i.e., requirements) than when using a traditional ARE approach. As far as the experiment results can tell, requirements elicitation in ARE is therefore more effective when using an ARE approach according to this thesis than when using a traditional ARE approach. In particular, with α=*0.10*, $H_{2.21}$ and $H_{2.23}$ would also be confirmed. The aggregated data and initial analysis results are included in Appendix F.

7.2.3.3 Subjective Assessment Results

The subjective assessment results using the post-questionnaire were analyzed in order to get an idea of how the participants of both groups perceived the used introductions. However, these findings were not statistically tested, as the underlying hypotheses were not comparative. Thus, we mainly investigated whether the ARE instructions according to the thesis approach are perceived as suitable, independent of whether the traditional instructions are too.

In Table 6, we show the assessment results according to certain criteria on a 5-point Likert scale ranging from "totally disagree (1)" to "totally agree (5)". The percentages indicate the number of participants who at least "rather agreed" to a statement. Thus, especially information about the reusable requirements, existing SPL constraints, and the progress of

elicitation are perceived to be very well presented in ARE instructions according to the thesis approach. The detailed participants' ratings can be found in Appendix F.

With regard to the three sub-hypotheses that aim at answering the research question regarding the suitability of representation, we found that about *85%* of the method group participants perceived the instructions according to this thesis as easy to read ($H_{4.11}$) and easy to use ($H_{4.12}$), which is why we consider these characteristics as fulfilled. However, only about *62%* felt appropriately supported in finding important information quickly ($H_{4.13}$). Thus, we consider the research question RQ_4 *"Is the representation of ARE instructions defined on the thesis approach perceived as suitable to support elicitation?"* as only partially answered with "yes", even though the instructions are not worse than the other ones.

Assessment Criterion	MG	CG
The elicitation instructions were helpful for performing the interview	100%	100%
The elicitation instructions provided me clear how-to guidance	85%	85%
The elicitation instructions informed me whether requirements of a certain type could be reused	92%	31%
The elicitation instructions informed me about the information which is relevant to be discussed	92%	92%
The elicitation instructions provided me with sound knowledge about the given constraints	92%	46%
The elicitation instructions were easy to read	85%	85%
The elicitation instructions were easy to handle	85%	62%
The elicitation instructions provided me with good indications to know when finished	85%	54%
The elicitation instructions supported me in finding important information quickly	62%	69%
The elicitation instructions allowed me to deviate from them, if necessary	54%	62%
I could answer all stakeholder's questions by using the information given in the instructions	54%	8%

The percentages indicate the number of participants who at least "rather agreed" to a statement.

Table 6. Subjective assessment results from experiment

Looking at the qualitative statements of the method group's participants, it was appreciated that a clear order of steps and precise instructions were given. Furthermore, the participants highlighted that additional information such as definitions of issues, additional costs, or constraints were provided. However, there were also critical comments that asked for better visual highlighting within the instruction text (e.g., important information such as constraints should be marked in bold, etc.). Furthermore, examples and better meta-explanations about the instruc-

tions document itself should be given. Finally, the order of issues should be better organized according to their level of abstraction.

7.2.3.4 Triangulation

During the interviews, the observer gathered several subjective impressions that were not tracked systematically, as these impressions had not been anticipated before. In order to get an empirical explanation for them, we performed a triangulation, where we established relationships between the objective observation data, the subjective assessment results, the qualitative comments of the participants, and the participants' personal backgrounds.

In the method group, only one participant objectively performed much worse than the other participants within this group. When considering the post-questionnaire of this participant, understanding problems may be a possible reason for this low performance. Interestingly, four other participants of the method group also mentioned handling problems or understanding problems in their post-questionnaires. However, when considering their objective performance, these participants did not perform significantly worse than the average within the group. According to their background, low English competency can be assumed as an explanation why they felt they had problems, even though they did not actually have any when only their objective data are considered.

However, when triangulating the results of the control group, we recognized another story. Here five participants mentioned handling problems or understanding problems in their post-questionnaire, with four participants actually performing worse than the average of their group. Interestingly, there were three other participants in the control group who also performed badly, but who did not mention any problems in their post-questionnaires. In order to identify what the actual problems might have been, we considered the background of all lower-performing participants. What most of them had in common was the fact that they had already performed a couple of interviews before. Thus, we assume that they had may have had different expectations regarding what an elicitation session should look like, which was even partially confirmed by their feedback statements in the post-questionnaire.

Hence, this triangulation confirmed the subjective impression of the observer that the participants in the control group who claimed to have interview experience did not precisely follow the instructions and tended to invent their own questions for the stakeholder, but also to give their own answers to the stakeholder's questions. Nevertheless, when all RE-experienced participants are removed from the sample, the confirmed hypotheses remain unchanged, at least from a mathematical point of view. Thus, it is not expected that the non-compliant participants in the control group affected the experimental results significantly.

7.2.4 Threats to Validity

7.2.4.1 Construct Validity

An important threat to construct validity is the mono-operation bias, as the study was performed based on one fictitious SPL only. Even though we assume the detected improvement tendency to be valid due to the significant differences, the actual values of the metrics will probably change when replicating the experiment in the context of another SPL. Furthermore, social threats to construct validity may also exist. Possible in this regard are especially evaluation apprehension and experimenter expectancies. Even though the participants were explicitly informed that the material and not their personal performance was being evaluated, most of them tried to look smart during the study. Thus, especially when they did not find sufficient information in the material fast, which was particularly the case in the control group, several participants tried to bring in their previous knowledge and experience, or invent their own questions and answers (e.g., about security, which was not a topic of interest here). Furthermore, evaluation apprehension can also be a reason why a couple of participants only gave information about the feasibility of requirements when they were explicitly asked to make statements about that. When replicating the experiment, the participants must therefore be instructed better to adhere to the material.

With regard to experimenter expectancies, the possible threat that the observer, who was the thesis author, influenced the results unintentionally (e.g., by a certain look, comment, or interruption) cannot be excluded. When replicating the experiment, we therefore recommend not having an observer in the experiment room again. Instead, the observer should sit behind a mirror glass or be replaced by video recording. However, for the fictitious stakeholder this threat does not apply, as she was not informed about the participants' group assignments and could, as she confirmed, not distinguish the group's elicitation style not even at the end of the study.

7.2.4.2 Internal Validity

As the experiment was organized as single sessions, each participant did the study during a different timeslot. Thus, each participant had another history before the experiment started, which might have had an impact on the results. For the same reason, there is a risk that the maturity of the participants was different depending on which time of the day they participated (e.g., early in the morning vs. late in the afternoon), even though the distribution over a day was equal between the groups. Furthermore, as the experiment was a highly human-based process, there is a likely risk that the fictitious stakeholder as well as the observer changed both their behavior and their attention during a day, and also

during the entire sequence of all sessions. In this regard, learning effects for the fictitious stakeholder and the observer can also be assumed. However, in the measured data, we cannot detect any difference between the participants at the beginning of the experiment and those at the end. In particular, as most participants were assigned alternately to the groups, all aforementioned threats are assumed to be balanced out.

With regard to material and instrumentation, no further threats were detected either. Five members of the method group and six members of the control group mentioned understanding or handling problems when using the instructions (see discussion on triangulation above). However, when statistical tests were run without these participants, the same hypotheses remain confirmed as when the full set of participants was used. In addition, $H_{2.21}$ (fewer unnecessary requirements) can also be confirmed with $\alpha=0.05$ when these participants are left out.

The risk that one participant had informed other participants who came to a later session was minimized through a non-disclosure agreement that all participants had to sign. In particular, when considering the experiment data, we did not find any evidence that later participants were better prepared or informed than the early ones.

Even though we had an equal number of RE-experienced and less experienced people in both groups, there were twice as many people with SPL knowledge in the method group than in the control group, which may be a potential threat. However, when we compared only the SPL-experienced people in both groups, or only the SPL-inexperienced people, we found that only hypothesis $H_{2.12}$ (fewer irrelevant questions) does not hold anymore, while the other hypotheses remain unchanged. Furthermore, hypothesis $H_{2.23}$ (higher satisfaction fit) can additionally be confirmed. Thus, we assume that the hypotheses $H_{2.13}$ (more correct information), $H_{2.22}$ (fewer problematic requirements), and $H_{2.24}$ (higher realization fit) are quite robust, while $H_{2.12}$ (fewer irrelevant questions) and $H_{2.23}$ (higher satisfaction fit) tend to be influenced by the participants' background.

7.2.4.3 Conclusion Validity

A basic threat to conclusion validity is the fact that the used metrics for assessing the effectiveness of different elicitation approaches is based on the same conceptual model on which the thesis approach is built. This holds especially true for the notion of relevant and irrelevant questions. In this context, it could not be evaluated whether success-critical requirements will be missing when corresponding questions are left out due to their classification as "irrelevant". Hence, the thesis' definition of "relevance" must be taken in mind when interpreting the results.

The reliability of measures can be considered as another threat to validity as the performance metrics were measured by only one human observer using manual recording tools (e.g., pen and paper), even though a voice recorder was used in addition. This was especially a problem when the fictitious stakeholder interrupted a participant too early, as this made it sometimes hard to record whether a right answer was given or whether a relevant question was posed. Thus, we recommend two actions when replicating the experiment: first, to apply more reliable instruments for observation, and second, to define clear rules for fictitious stakeholders regarding interrupting questions.

While the statistical power of the confirmed hypotheses was very high (up to *~1.00*) and violations of statistical test assumptions were avoided too, another possible threat to conclusion validity could be the heterogeneity of the participants. Especially as students typically differ significantly in their performance, the variance in each group would probably be lower if real requirements engineers were used. Thus, even though we had problems recruiting practitioners, we recommend involving as many as possible when replicating the experiment.

7.2.4.4 External Validity

Even though the elicitation interviews were made very realistic (e.g., by the interrupting questions, etc.), there are threats to external validity. The most important threat in this regard is probably the fact that all participants were students. Indeed, precise approaches in requirements or software engineering such as the instructions used are never intended to support or even replace expert engineers. However, average engineers who should be supported by a systematic approach are also no novices. Again, we therefore recommend involving a group of real requirements engineers in order to increase the external validity of the results.

Furthermore, elicitation sessions in practice are typically more complex and more interactive, involving a multitude of people, workshop techniques, etc. However, as such a setting cannot be replicated in a controlled environment, we recommend performing additional case studies.

A final remark with regard to external validity is the fact that the traditional ARE approach used by the control group is a combination that does not exist in this form yet (see related work in chapter 3). Thus, as the control group worked with an approach that is already more advanced than the approaches currently used in reality, we assume that the benefits when using our ARE approach may be even higher than shown by this experiment. Of course, this hypothesis still has to be evaluated.

7.2.5 Interpretation and Implications

The results of the controlled experiment have shown that requirements engineers can basically work more effectively in ARE when they use ARE instructions according to the approach introduced in this thesis. In this subsection, we would like to discuss the reasons and the implications that these improvements may have.

The first significant improvement is that participants of the method group asked *3.3* times fewer irrelevant questions than the participants of the control group. The reason for this improvement is the fact that the elicitation instructions according to the thesis approach were not just "best practice" but individually tailored to the specific information needs that had to be satisfied. Thus, questions regarding issues no one cares about were not included. However, future work is needed to investigate whether the requirements will be complete when just focusing on the "relevant" questions according to the definition introduced in this thesis, or whether critical requirements could be missing.

In this regard, it must also be noted that the time needed to perform the interviews was almost equal in both groups. While the method group needed *28.8* minutes, the control group needed *28.5* minutes on average. With an effect size of $d=0.05$ and $p=0.46$, the corresponding null hypothesis "no difference" cannot be rejected. Thus, even though the higher effectiveness achieved in the same amount of time indicates higher efficiency in the method group, no absolute time savings could be achieved when asking fewer irrelevant questions. Rather, the saved time seems to be spent on discussing other issues in more detail.

The second and third significant improvements are that participants of the method group elicited *2.92* times fewer problematic requirements and achieved a *1.11* times higher realization fit than the control group participants. The likely reason for these improvements is the fact that the existing constraints are explicitly mentioned in corresponding places in the method group's ARE instructions. In the control group, in contrast, feasibility constraints are only mentioned in the instructions for the explicitly anticipated requirements already, while all other constraints are just described "between the lines" in the SPL specification. Apparently, this direct representation of constraints in the elicitation instructions seems to be more suitable. As an implication for practice, it is therefore likely that better fitting requirements can be elicited when using the thesis approach. This realization fit improvement is then expected to also increase the overall AE efficiency. However, as already mentioned in section 7.1, this "efficiency of AE" hypothesis (H_1) still has to be evaluated.

The last significant improvement is finally that participants of the method group could correctly answer 1.93 times more stakeholder questions on their own than participants of the control group. The reason for this improvement is the fact that process and product knowledge about an SPL

is represented more appropriately in the ARE instructions according to the thesis approach than in the material according to the state of the art. The practical implication of this improvement is that requirements engineers can discuss and negotiate in a more competent way. This probably allows stakeholders to feel more convinced when their requirements are put into question.

Another important result of this experiment is that the participants of the method group did not perform worse in any metric than the participants of the control group. A likely explanation is the fact that our ARE approach enhances the state of the art and does not replace it. Thus, it is expected that requirements engineers using this approach can benefit significantly without having any relevant drawbacks.

However, the representation of the ARE instructions was not considered sufficiently suitable yet by the study participants. Thus, the qualitative feedback given by them has to be incorporated into the ARE instruction template in order to improve the subjective perception of the resulting instructions. Furthermore, it has to be kept in mind that the thesis approach is intended for rather flexible SPLs, i.e., SPLs in which the degree of explicitly anticipated requirements is limited. Thus, when highlighting the benefits of the thesis approach, the results cannot be simply transferred to rather configurable SPLs in which almost every requirement can only be satisfied by instantiating a predefined variability model (VM).

7.3 Case Study

While the previous section described a controlled experiment for evaluating the effectiveness of elicitation (H_2) and the suitability of representation (H_4) when using ARE instructions as proposed by this thesis, this section introduces a case study for evaluating the effectiveness of the underlying tailoring approach (H_3). Besides the goal of the study, its design and setup, its results and threats to validity, as well as its implications for research and practice are described.

7.3.1 Goals and Hypotheses

According to the GQM approach [BCR94], the overall goal of this case study was to analyze the thesis tailoring approach for the purpose of evaluation with regard to effectiveness from the viewpoint of practitioners in the context of a two-phase case study. Thus, the goal was to know whether the tailoring approach proposed by this thesis actually enables different engineers to successfully derive ARE instructions when following the tailoring approach proposed by this thesis. Below, we introduce the underlying research questions and hypotheses of the case study, as

well as their related metrics (in brackets). However, for reasons of brevity, we omit the corresponding null hypotheses here.

All hypotheses are based on the main research question $RQ_{3.M}$, which asks whether engineers are able to tailor ARE instructions based on a given SPL without major problems. As the ability to do something requires both the understanding of what to do and the actual capability to carry it out, we expect that most engineers using the tailoring approach will

- $H_{3.1}$ be able to successfully create an ARE instructions document (# participants who successfully finish the tailoring / # all participants)
- $H_{3.2}$ not need external help or explanations (# participants requiring help / # all participants).

In this regard, we expect that at least 80% of the participants are able to successfully create ARE instructions and that less than 20% of the participants need help for this.

7.3.2 Study Setup

Based on the study goals, the case study was designed and prepared. Below, we describe the details of the case study setup.

7.3.2.1 Participants (Subjects)

The participants in the case study, who participated voluntarily, were two software architects and two requirements engineers from Fraunhofer IESE, as well as a development expert from a medium-sized German software company dealing with the development of content recognition systems. All participants were informed about our study goal. Regarding their background, the person from the software company had more than ten years of experience in different positions at his organization (ranging from platform development to project management and RE in customer projects). The two architects from Fraunhofer had around two years of consulting and scientific experience, and the two requirements engineers five, respectively nine, years of experience in their area. No participant had knowledge about the tailoring approach before. For four of the five participants, it was even the first time that they had to deal with the adaptation of a requirements process at all. Regarding knowledge of SPL engineering, all participants mentioned that they were aware of the basic concepts, but not experts in this area.

7.3.2.2 Case Study Design and Procedure

The case study was conducted in two phases (see Figure 63). In the first phase, the software architects and requirements engineers from Fraunhofer IESE used the tool-supported tailoring approach of this thesis to derive an ARE instructions document based on a fictitious SPL in the domain of business process applications. In the second phase of the study, the same tool-supported tailored approach was used in a real software organization to derive an ARE instructions document based on the software platform on which customer-specific solutions are built there.

For the execution as well as for data collection and analysis, different artifacts were used. The Fraunhofer experts as well as the person from the software organization both received the tailoring tool for guidance and support throughout the tailoring process. Furthermore, both parties received a tracking sheet for measuring and assessing the required effort, the understandability, the applicability, the information sources used, and the help required in each tailoring step. For gathering qualitative feedback, the Fraunhofer experts additionally got a list of four open feedback questions, while the person from the software company was just asked to write down his personal impressions on a sheet.

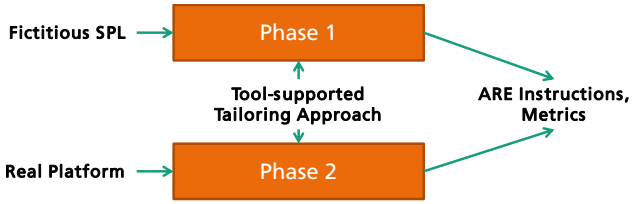


Figure 63. Overall setting of case study

The first phase of the case study with the Fraunhofer experts was conducted in summer 2011 and was split into four single sessions of 120 minutes each. Thus, the participants performed the tailoring individually, as it was not organizationally possible to let all of them perform the tailoring in parallel. In order to avoid the threat that one participant informed other participants about the material, the participants were kindly asked not to talk about the material with their colleagues until the first phase was over (a non-disclosure agreement was not used here).

In each session, the participants received the tailoring tool after we gave them some initial information about the purpose of the study. Each participant then used about 15 minutes to get familiar with the tool. During this time, the participants carefully read the explanations on the screens and clicked through the tool wizards. In order to provide some examples and keep the study within a manageable time slot, the first three tailoring steps (e.g., characterization of SPL, identification of architectural element types, and identification of architectural elements) had already

been performed by the thesis author before. Thus, these steps did not have to be processed by the participants again.

In the second step, the participants then started with the actual tailoring. Due to time reasons, there was no SPL specification handed out to the participants as tailoring input. Rather, the thesis author played the role of an SPL expert who was interviewed by the participants in order to extract the required SPL knowledge. In order to provide each participant with the same knowledge, the thesis author used a fictitious SPL specification as an information source. The given answers were entered into the tool by the participants. At the end of each tailoring step, the understandability, the applicability, the help required as well as the time required were tracked, respectively assessed, by the participants using the provided tracking sheet (see Appendix G).

After all tailoring steps had been completed, the participants were asked to answer the four open questions to allow gathering their qualitative feedback. Furthermore, they were asked to send us the database file of the tailoring tool as well as the generated ARE instructions. These results were then reviewed by the thesis author, who counted the elements created in each tailoring step and checked (based on a sample solution) whether there were wrong entries in the database or "strange" statements in the generated instructions.

As a starting point for the second phase of the case study, which was conducted between September 2011 and February 2012, a two-hour kick-off meeting was performed with two employees from the aforementioned software company. In this meeting, we presented the purpose and the expected benefits of the tailoring approach as well as the evaluation goal of our study. After then briefly explaining each tailoring step, we started to perform the tailoring with a few examples in order to make the company's people familiar with the tailoring tool and its underlying concepts. At the end of the meeting, the tool and a tracking sheet were handed out to one employee of the company, who committed himself to apply the tailoring approach within the next month. In parallel, five baseline questionnaires were handed out to different project leaders in the company in order to get data about the effectiveness and efficiency of RE in past customer projects (see Appendix I).

The filled-out database file of the tailoring tool, the tracking sheet, the feedback sheet, and the baseline questionnaires were sent back to us by the company a couple of months later. Again, these results were reviewed by the thesis author, who counted the elements created in each tailoring step and checked whether there were wrong (respectively strange) entries in the database or in the instructions.

7.3.3 Analysis

Based on the analysis of the tailoring's intermediate and final results as well as of the gathered feedback, an analysis was performed. The analysis procedure and its results are described in this section.

7.3.3.1 Analysis Procedure

In the first step, we used the database files of the participants' tailoring tools for generating the corresponding ARE instructions anew. We reviewed the resulting instructions carefully and looked for technical or linguistic problems, or for meaningless or strange elicitation instructions or hints. For each detected problem, we analyzed in which tailoring step a corresponding mistake was made. We then counted and classified these mistakes for each tailoring step, and tried to correct these mistakes wherever possible in order to assess their impact on further tailoring steps. However, whether the ARE instructions sufficiently reflected critical concepts of the underlying SPL was not checked, as this would have required actual application of these instructions in a project.

The gathered data about mistakes were then entered into MS Excel, and supplemented with the data provided regarding help required and time required in a certain tailoring step. Furthermore, quantitative ratings from the tracking sheets according to the Likert scale were collected in MS Excel. Finally, the corresponding qualitative comments were listed in MS Word. In this regard, all data was processed in a confidential way.

7.3.3.2 Objective Measurement Results

In this subsection, the measurement respectively review results of the entire case study (i.e., both phases) are shown and explained (see Table 7).

In this context, we consider a mistake in the second column of Table 7 as critical when it leads to meaningless, inconsistent, incomplete, or ambiguous elicitation instructions or hints. Linguistic or syntactical mistakes that do not result in the aforementioned problems are not considered to be critical. With regard to the last column, the numbers in brackets indicate the time required by the person from the software organization, while the numbers without brackets present the average of all participants. However, all findings were not suitable for the purpose of statistical hypothesis testing, as there were neither enough data points nor a clear baseline against which the data could be compared. Thus, only the raw analysis data are presented below.

In the ARE instructions created by the participants of the first phase (i.e., the Fraunhofer experts), only one critical mistake was recognized. In particular, two participants neglected to resolve existing conflicts between

conceptual relationships and development phase assignments of certain issues (see corresponding step in section 6.3.10). Even though this mistake prevented the tool from generating the ARE instructions due to a deadlock situation, it could be corrected within a few minutes and was rather a technical than a conceptual problem.

Tailoring Step	Participants requiring help from author	Participants making critical mistakes	Required effort in minutes per element
Characterization of Software Product Line (SPL)	0	1	N/A (30.00)
Identification of Architectural Element Types	0	0	N/A (12.00)
Identification of Architectural Elements	0	0	N/A (4.50)
Characterization of Supported Flexibility Classes	4	0	2.70 (8.57)
Identification of Flexibility Assumptions	3	1	5.80 (13.33)
Characterization of Development Phases	0	0	4.20 (15.00)
Identification of Development Activities	1	1	0.90 (1.76)
Elaboration of Decisions and Corresponding Information Needs	1	1	1.80 (4.09)
Determination of Relevant Issues	1	0	3.70 (17.1)
Determination of Conceptual Relationships	1	3	3.00 (13.33)
Definition of ARE Elicitation Instructions (completely automated)	N/A	N/A	N/A

Table 7. Results of case study

In the ARE instructions generated by the participant from the software company, more critical mistakes were recognized. The most important problem was an inconsistency in the conceptual relationships that prevented the tool from generating the ARE instructions due to a deadlock situation. However, in contrast to a missing resolution of conflicts as by the phase 1 participants, the reason here was that the participant did not adhere to the best practice relationships expressed in the issue model, but tried to delete or redefine own relationships. Another critical mistake was that a couple of identical constraints were entered into the tailoring tool; once as a hard constraint and once as a soft constraint, which did not make sense at all. A mistake with regard to the elaboration of information needs was that the participant forgot to specify under which conditions these information needs have to be satisfied. This was critical insofar as the high flexibility of the underlying platform affected the concrete information needs that may exist during runtime (e.g., some information needs depend on a previous decision concerning a variable architectural element). Finally, some information was not de-

scribed in a format that fit the text blocks in which the information was to be incorporated. This linguistic mistake led to ambiguous phrases in some cases.

Regarding our hypothesis $H_{3.1}$, which stated that 80% (four) of the participants are able to successfully derive ARE instructions based on a given SPL, we found that only 40% (two) of the case study participants were actually able to do this straightaway (i.e., without rework). In most cases, however, this was caused by a lack of conflict resolution in the "Determination of Conceptual Relationships" step, which resulted in deadlock situations during generation. Further steps in which critical mistakes were made could only be found in the case of one participant from the software organization. These steps were especially those in which the solution space (mainly the SPL constraints or information needs) had to be aligned with the problem space (i.e., the issues).

Regarding hypothesis $H_{3.2,}$ which claimed that less than 20% of the participants need help for tailoring, we found that this was also not possible at least in tailoring steps that dealt with flexibility classes and their assumptions. Here, more than half of the involved peopled (four) needed our support. The interpretation of the corresponding implications is discussed in section 7.3.5.

7.3.3.3 Subjective Assessment Results

The subjective assessment results (i.e., the assessment of understandability and applicability as well as the qualitative feedback) were analyzed in order to get an additional impression of how the participants of both groups perceived the tailoring approach. Again, these findings were only analyzed informally. Furthermore, we do not distinguish between the phase 1 and the phase 2 participants here.

In Table 8, the assessment results for each tailoring step on a 5-point Likert scale ranging from "totally disagree (1)" to "totally agree (5)" are shown. The percentages indicate the number of participants who at least "rather agreed" to a statement. An asterisk (*) marks the steps that were experienced as being "easy". The detailed ratings of each participant can be found in Appendix H.

While the tailoring steps dealing with development phases, development activities, and relevant issues were easy to handle by most participants, the identification of architectural elements, flexibility assumptions, information needs and conceptual relationships were perceived as rather challenging. This is interesting insofar as most participants stated that they indeed understood what they had to do, but experienced problems when actually performing these steps.

Looking at the qualitative statements of the participants, they agreed that the tailoring approach is basically easy to use, provides precise guidance on how to extract SPL knowledge, does not require much preknowledge in RE, and highly automates the representation of SPL knowledge into ARE instructions. However, the participants mentioned problems when matching solution and problem space, especially when aligning flexibility class assumptions or information needs with corresponding issues. In particular, while the architects mentioned that they were totally unfamiliar with the concept of issues, the involved requirements engineers stated that they were unfamiliar with the concept of architectural element types and architectural elements. Thus, explaining these concepts better was required to make the tailoring approach better applicable.

Tailoring Step	**I understand what I have to do**	**I can perform the step without problems**
Characterization of Software Product Line (SPL)*	100%	100%
Identification of Architectural Element Types	80%	0%
Identification of Architectural Elements	60%	0%
Characterization of Supported Flexibility Classes	60%	60%
Identification of Flexibility Assumptions	60%	40%
Characterization of Development Phases*	80%	80%
Identification of Development Activities*	80%	80%
Elaboration of Decisions and Corresponding Information Needs	80%	60%
Determination of Relevant Issues*	80%	80%
Determination of Conceptual Relationships	80%	60%
Definition of ARE Elicitation Instructions (completely automated)	N/A	N/A

The percentages indicate the number of participants who at least "rather agreed" to a statement.

Table 8. Subjective assessment results from case study

In addition, the person from the software company mentioned that some parts of the tailoring approach are hard to apply for software platforms in which there is no clear separation between configuration and additional development, e.g., when domain-specific languages are used. Furthermore, he mentioned that in order to come up with a high-quality ARE instructions document, the tailoring approach must be performed twice, i.e., once for initially extracting the required SPL knowledge and once for iterating the extraction results in order to reflect the SPL characteristics best. This was also a reason why this person required much

more time in each tailoring step than the rest of the participants. Furthermore, he assessed the usability of the tailoring tool as improvable.

The raw data of the analysis are included in Appendix H.

7.3.4 Threats to Validity

7.3.4.1 Construct Validity

An important threat to construct validity is the mono-operation bias, as the study was performed based on one fictitious SPL and one real product platform only. Even though we assume the observed effectiveness to be valid, replicating the study in other SPL organizations is needed in order to provide more evidence for this claim.

Furthermore, while the participants in the first phase followed the tailoring steps and interviewed a fictitious SPL expert as intended by the approach, the participant in the second phase acted in different roles. Besides acting as the method tailor, he also took the role of an SPL expert as well as working as a surrogate for further roles to be involved (e.g., in the information need step) due to the low availability of these people. When replicating the case study, the participating companies must be instructed to better distinguish the required roles in order to use the approach as intended.

Further threats to construct validity could be that the gathered subjective statements were not always honest because the participants were informed that the purpose of the study was to evaluate the results of a PhD thesis. In addition, there is a possible risk that the thesis author influenced the results unintentionally when playing an SPL expert. We therefore recommend not involving a person that already knows the tailoring method when replicating the study (no matter in which role).

7.3.4.2 Internal Validity

As the first phase of the study was performed in four single sessions, there is a risk that the attention of both the participants and the fictitious SPL expert varied depending on which time of the day the session took place (e.g., early in the morning vs. late in the afternoon).

In the second phase of the study, which was done at a software company, there was no control at all. In particular, as the tailoring was distributed over a period of months, there were probably many influencing factors that could have had effects on the results. Even if internal validity is basically low in each case study, we therefore recommend performing replicating studies within a defined time-slot, e.g., during a three-day

workshop, rather than distributing the tailoring steps over weeks in parallel to the daily business. In particular, the participants should work in an atmosphere that allows them to work in a concentrated manner and with the required accuracy.

7.3.4.3 Conclusion Validity

The reliability of measures can be considered as a threat to validity, as the success of the tailoring performance was assessed based on the thesis author's expert judgment, and the data regarding required time and help were tracked by the participants themselves. Thus, we recommend applying more reliable instruments for measurement to determine the success / effectiveness in replicating studies.

Another threat to conclusion validity were the small sample size and the corresponding impossibility to perform meaningful statistical analyses such as descriptive statistics or hypothesis tests.

7.3.4.4 External Validity

The external validity of the study is basically high due to the involvement of practitioners and an industry-size SPL. However, a threat to this validity might be the fact that the involved software company applies a platform-oriented SPL approach (see [DSB05]) rather than a full-fledged product line engineering approach as proposed in academic textbooks. If feasible, an organization using such state of the art SPL engineering methods should be involved when replicating this case study.

7.3.5 Interpretation and Implications

The results of the case study as well as the observations made during the development of the tailoring approach have shown that different people can basically incorporate SPL knowledge into ARE instructions when they use the tailoring approach introduced in this thesis. However, as less people than expected were able to successfully do this, this subsection discusses the case study findings and their implications.

On average, all participants confirmed the statement that they always knew what to do in the different tailoring steps, and they could perform most steps without problems. The steps in which the participants felt challenged were also those in which the critical mistakes were made. Interestingly, these steps (i.e., Identification of Flexibility Assumptions, Elaboration of Decisions and Corresponding Information Needs, and Determination of Conceptual Relationships) are the ones that form the key novelties of the thesis approach, as the reflection of assumptions or the

notion of information needs are the concepts that differentiate this thesis approach from existing ones.

Therefore, it is indispensable that these underlying core concepts as well as the corresponding tailoring steps are explained comprehensively before the tailoring approach can be used effectively in practice. This claim is supported by the observation that the participants in the first phase of the case study, who could ask the thesis author for help directly, performed better than the person from the software company who did the tailoring offline and asked for help only once. We therefore pose the hypothesis that the tailoring approach proposed by this thesis is basically applicable, but not self-explanatory. Future work should therefore deal with additional studies to evaluate the effectiveness of tailoring more thoroughly. Furthermore, training and coaching as well as an improved usability of the tool are seen as prerequisites before practitioners can effectively use the tailoring approach for creating individual ARE instructions based on a given SPL without help.

With regard to the effort needed by the software company to perform the tailoring in the case study, we consider this investment as highly justified. The processed software platform consists of about 40 high-level architectural elements of ten different types, supports 21 flexibility classes, and requires about 45 coarse-grained decisions to derive individual software solutions. The overall effort for deriving ARE instructions from this platform was 22.5 person-hours (~2.8 person-days). In past AE projects (see Appendix I), the average project size had been about 460 person days, of which about 60 person-days (~13.0%) had been spent on requirements activities. As the effort for requirements rework alone was about 15 person-days in these project on average, we consider the effort for tailoring as an investment that should pay off fast. However, the evaluation of the actual efficiency improvements in AE projects is, as already mentioned, beyond the scope of this thesis and still has to be done in order to assess this pay-off with the necessary evidence.

7.4 Summary

As research always involves gaining a deep understanding about the (intended) effects of a solution [Bas93], a sound evaluation of the thesis contributions is necessary. In this chapter, a controlled experiment for evaluating the effectiveness of elicitation when using ARE instructions according to this thesis as well as a case study for evaluating the effectiveness of the thesis tailoring approach have been presented.

The results of the controlled experiment have shown that requirements engineers can basically work more effectively in ARE when they use ARE instructions according to the approach introduced in this thesis. In particular, less problematic requirements can be elicited in each elicitation

session, leading to faster achievement of a certain realization fit. Furthermore, requirements engineers are able to inform stakeholders more convincingly about the capabilities and constraints of a given SPL, which also increases the effectiveness of the entire elicitation process.

However, even though objective effectiveness improvements can be achieved with this thesis approach, the representation of the ARE instructions was not considered sufficiently suitable yet by the study participants. Thus, their qualitative feedback has to be incorporated into the ARE instructions template in order to improve the subjective perceptions as well.

Hypotheses	Confirmed
H_1 Efficiency of Application Engineering	(not tested)
H_2 Effectiveness of Elicitation	yes
H_3 Effectiveness of Tailoring	partially
H_4 Suitability of Representation	partially

Table 9. Summary of evaluation results

Of course, as elicitation effectiveness improvements can only be achieved when these ARE instructions have been derived from the underlying SPL, the effective creation of ARE instructions is an indispensable aim. The results of our case study carried out in this regard have shown that different people are basically able to incorporate SPL knowledge into ARE instructions when they use the thesis tailoring approach. However, we found that some of the tailoring steps are not self-explanatory and may tempt people to make critical mistakes. Therefore, it is indispensable that the underlying concepts as well as the tailoring steps are explained and trained before the tailoring approach can be used successfully. Furthermore, the usability of the tool support should also be improved in order to make the tailoring easier for method tailors in practice.

However, as the incorporation of the evaluation findings into the thesis components will not be done as part of this thesis anymore, it should be addressed in future work. The next chapter therefore summarizes the achievements of this thesis and lists interesting ideas for possible or even required research activities.

In this regard, future work should particularly deal with the impact of the thesis approach on AE efficiency, as this is surely the most relevant information for practitioners. So far, only the effectiveness of elicitation has been evaluated. Hence, it is still interesting to see which impact this effectiveness may have on development efficiency in AE project.

Table 9 summarizes the findings of our studies.

8 Summary and Future Work

"You should not try to foresee the future,
but to enable it."
Antoine de Saint-Exupery

This chapter describes the contributions of this thesis and gives an outlook on future work. The purpose of this chapter is to explain the novelties that now exist and the research directions that could be followed. For this purpose, the chapter first summarizes the achievements of the thesis with regard to foundation, methodological approaches, engineering support and empirical evaluation, and then lists possible enhancements in these areas.

8.1 Contributions

8.1.1 Foundation

In this thesis, a conceptual ARE model (see chapter 2) and an issue model (see chapter 5) were developed in order to provide a solid basis for the development of other thesis components. For this purpose, we first defined and clarified important terms in the area of ARE. In particular, the notion of "relevant requirements" has been introduced in order to highlight that not all requirements are actually useful when developing new systems in a reuse-based way.

In order to guide ARE processes towards the elicitation of these relevant requirements, the conceptual ARE model clarifies how the relevance of requirements can be determined in the context of a certain SPL. Furthermore, the model explains how ARE processes are conceptually related to an SPL, and how SPL characteristics influence requirements elicitation steps. In particular, the interplay between a product line architecture, a development strategy, RE best practices and concrete ARE processes is formalized in this model, which allows aligning these concepts in product-oriented and process-oriented manner.

However, as the conceptual ARE model does not describe any content to be discussed in an ARE process, we also developed an issue model for IS that acts as a consolidation of RE best practices in this area. The purpose of this model is to provide basic knowledge about the typical topics for which requirements have to be elicited there. In particular, the issue model acts as an input for our tailoring approach (see chapter 6), as it describes the elements of the problem space with which constraints or information needs of an SPL organization can basically be concerned.

8.1.2 Methodological Approaches

The methodological approaches described in this thesis deal with the question of how SPL knowledge can be systematically extracted and represented to AE requirements engineers. For this purpose, we developed an ARE instructions template (see chapter 4) and a tailoring method (see chapter 6).

The ARE instructions template provides a generic structure as well as an elicitation strategy, and a set of predefined text blocks for representing both best practices and important knowledge about an SPL to requirements engineers in a suitable manner. The basic idea implemented in this template is to perform requirements elicitation in an algorithmic way. This means that each issue, with which requirements can be concerned, is processed in a repeatable order and manner. To make this happen, the template structures elicitation instructions documents in the same way as the intended ARE processes are structured. Thus, for each phase of an ARE process, a corresponding section must exist within an ARE instructions document. Furthermore, for each issue to be discussed during a phase of the ARE process, a corresponding sub-section has to be defined. In these sub-sections, concrete instruction statements and hints are then provided, which support requirements engineers in eliciting all requirements concerning one specific issue. Hence, these instructions and hints provide algorithmic guidance to the requirements engineers regarding the elicitation activities to be done and the information that must be considered.

The second methodological approach developed by this thesis is the tailoring method. This approach describes a clear sequence of activities to be carried out by method tailors during the DE/FE phase in order to derive ARE instructions from a given SPL. For this purpose, the tailoring approach prescribes eleven algorithmic steps that precisely explain how knowledge about the product line architecture, the intended development strategy, as well as RE best practices has to be extracted and combined in order to provide AE requirements engineers with precise and helpful elicitation instructions according to the aforementioned template. We consider this tailoring approach as a valuable and especially novel computer science contribution, as it formalizes this procedure in a way that allows an (semi-)automated performance.

8.1.3 Engineering Support

To support the aforementioned methodological approaches, a tailoring tool has been introduced in this thesis. The purpose of this tool is to assure that the thesis approach can be leveraged in practical settings, as otherwise it might be too tedious and time-consuming.

The tailoring tool was developed as a Visual Basic application based on MS Access 2010. It provides a wizard to guide method tailors in collecting information about a given SPL, and it (semi-) automates the tailoring, as it generates proposals for the results of different tailoring steps based on the results of previous steps wherever possible. Thus, when using the tool for tailoring, method tailors only need to enter information that the tool cannot calculate based on already entered information. Apparently, the further the tailoring proceeds, the less input needs to be provided by a human. The last step of tailoring, for instance, is even fully automated, i.e., the tool is able to create an ARE instructions document based only on the intermediate results achieved before. Thus, already in this single step, significant time and effort can be saved, as the performance of this step can be reduced from several hours to a couple of seconds. Furthermore, the correctness of the generated ARE instructions can be assured constructively as the tool applies a validated template (see chapter 4) and does not require the method tailors to select phrases manually. In general, the tool continuously applies plausibility checks, which increase the correctness and completeness of the processed information. Without this help, the application of the tailoring would likely be much more complicated, error-prone, and time-consuming.

8.1.4 Empirical Evaluation

In this thesis, empirical evaluations (see chapter 7) were used to show the usefulness of the methodological approaches with regard to the research questions introduced in chapter 1.

In a case study with two RE experts and two architecture experts from Fraunhofer IESE as well as a practitioner from a medium-sized software company, we tried to evaluate the effectiveness of the tailoring method in a first step. As our goal was to provide evidence that ARE instructions can be defined systematically, we let these people tailor such instructions based on a given (fictitious) SPL, respectively a real software platform on which the involved company develops its customer-specific solutions. The results of this case study have shown that an incorporation of SPL knowledge into ARE instructions is basically possible when using the tailoring as proposed in this thesis. In particular, all participants confirmed the statement that they knew what to do in the different tailoring steps, and that they could perform most steps without problems. Furthermore, we found that even for industry-size SPLs, an ARE tailoring is feasible within less than one person-week, which we consider justifiable effort.

However, we also found that it is indispensable that the underlying concepts as well as the tailoring steps are much better explained before the tailoring can be effectively used in practice. In this context, we received important feedback for improving the tailoring approach and especially the usability of the tool.

Besides the case study, we also performed a controlled experiment with 26 students in order to evaluate whether requirements engineers using an ARE instructions document according to this thesis are able to elicit requirements more effectively. As our goal was to investigate whether this is actually the case, we let a group of students perform interviews with "our" ARE instructions, while another group used state of the art ARE instructions for this purpose. The results of the controlled experiment showed that people can actually work more effectively in requirements elicitation when they use ARE instructions according to this thesis. In particular, we found that fewer irrelevant questions are posed, fewer problematic requirements are elicited, a higher realization fit is achieved, and more stakeholder questions are answered correctly and convincingly. Another important result of this experiment was that when using ARE instructions according to this thesis, no other quality characteristic of the elicitation approach decreased (e.g., no important requirements were forgotten, etc.). A possible explanation could be the fact that our ARE approach enhances the state of the art and does not replace it. Thus, requirements engineers using this approach are expected to benefit significantly without experiencing any relevant drawbacks. However, the representation of the ARE instructions was not considered to be optimal by the participants.

8.2 Open Issues and Future Work

8.2.1 Foundation

While the conceptual ARE model constitutes a valuable contribution towards clarifying the relationships between SPLs and ARE processes, it is not comprehensive in the sense that it covers all aspects that may exist in SPLs or in AE projects in general. This means that concepts that go beyond RE may probably not be described sufficiently in this model for being able to build other software engineering methods. An interesting part of future work could therefore be investigating how this model must be extended in order to provide a suitable foundation for other software engineering disciplines as well. For instance, quality assurance processes or detailed architecting methods for customer-specific systems could be derived with an extended version of this model. Furthermore, it is interesting to see whether this model remains stable when applied in the context of modern architectural styles or system types such as service-oriented, event-based, or emergent systems.

With regard to the proposed issue model, we are aware that there is no one-fits-all-solution, even though we carefully elaborated the contained issues by applying an iterative research approach. Adapting the issue model based on the actual information needs that exist in a development process is therefore an important step in the overall thesis approach. Nevertheless, future work should analyze whether the issues described in the literature (and, thus, also in our issue model) are valid anyway, and whether there is an empirically funded set of issues that is needed every time. Furthermore, future research could deal with the question of how requirements concerning certain issues should be represented best in order to make both the provision and consumption of requirements in AE as efficient as possible. Maybe the ongoing work of Gross [Gro10] will be able to provide suitable answers in this regard.

8.2.2 Methodological Approaches

Even though the representation of SPL knowledge based on the ARE instructions template has enabled participants of a controlled experiment to perform more effective elicitation, we have identified room for future work in this regard. Besides better representation of the information in ARE instructions documents, which was asked for by the study participants (see section 7.2.3.3), future work could also investigate which additional support would be helpful for requirements engineers during elicitation, analysis, prioritization, specification, and validation of requirements in an AE projects. Especially as requirements elicitation always deals with trade-off decisions, approaches that provide information beyond those of the current ARE instructions template would be welcome. Such work could facilitate the prioritization of requirements, or the immediate calculation of additional costs. In this regard, the role of tool support should also be discussed, either to improve the impact assessment for customer-specific requirements or to align requirements with existing solution assets directly. This is especially an important topic when service repositories are used that may, in contrast to traditional SPLs, change rapidly over time and do not allow adapting the rather static ARE instructions continuously.

With regard to the tailoring, which forms the core contribution of this thesis, we see future work mainly in the area of automation. Even though we were able to provide very precise and formal guidance for each tailoring step, the actual degree of automation is still limited. Thus, a significant part of knowledge about a given SPL can only be extracted from informal documents or SPL experts in a human-based way. Future work should therefore deal with the question of how the degree of automation can be increased further. This challenge is directly concerned with the discussion about future work regarding engineering support, and a technical rather than formalization problem.

From a conceptual point of view, there are also possibilities for future research. First of all, it should be investigated whether the tailoring also works when no SPL in the academic sense is used, but rather a service platform or any other reuse approach. This also holds true for platforms in which an explicit distinction of configuration and programming using a domain-specific language is hard to make (see case study in section 7.3). In particular, it should be analyzed how short-lived or open platforms (like large service repositories in the Cloud) will affect the feasibility of the tailoring approach. Research must be done on the question of what can remain stable in such a setting (and be reflected in an explicit ARE instructions document) and what has to be addressed with a complementary approach due to the high change frequency. Furthermore, the feasibility of our tailoring approach in development contexts in which there is no clear development strategy must be checked. Particularly since an increasing number of IS is nowadays built in an agile way, investigating the impact of this paradigm on the elaboration of development activities, decisions, and information needs is important.

With regard to the state of the art in SPL engineering, the integration of the tailoring approach in the DE/FE phase should finally be investigated more thoroughly. In particular, it is interesting to see which additional assets from DE/FE could also be taken as input for tailoring. Again, specific attention should be paid to the automatic processing of this input.

8.2.3 Engineering Support

As it was an important finding from our case study described in section 7.3, the usability and visualization of the tailoring tool has to be improved in order to make the tool easier to use. As a critical point in this regard, especially the descriptions of how to perform the tailoring steps with the tool should be enhanced, as otherwise, wide applicability in practice will not be possible without coaching.

In this regard, we consider better support for aligning the concepts of the solution space with the concepts of the problem space (issues) as indispensable. Interesting future work could include an investigation of how text-mining approaches could support the alignment of assumptions with corresponding issues automatically. A similar enhancement could also be valuable in the tailoring step that deals with the elaboration and alignment of information needs. Furthermore, it would also be interesting research to find out whether it is possible to automatically derive assumptions from the technologies or the architectural elements used. This could mean that meta-information about existing components are automatically analyzed in order to understand the constraints under which these components can provide their services. This is a challenge insofar as standardized, semantic descriptions of assets do not exist yet and probably will not exist for a long time.

Furthermore, we see future work in upfront and downstream activities. In particular, the extraction of architectural element types and architectural elements seems to be a tailoring step that could be automated to a high degree by processing the architecture or even the source code of an SPL. A prominent tool at Fraunhofer IESE that deals with architectural analysis is the SAVE tool [DKL09]. In future work, it would therefore be interesting to see how SAVE could enter extracted architectural element types and architectural elements into the tailoring tool automatically. In addition, it would be helpful to provide tool mechanisms that quantify the architectural impact (low, medium, high) of supported flexibility classes. This work could disburden software architects during tailoring from making this assessment, and also increase the degree of automated tailoring.

With regard to downstream activities, such as the actual elicitation to be done during AE, it would be nice to provide requirements engineers with more than just an ARE instructions document in MS Word. We therefore plan to investigate how the tailoring tool could generate an elicitation tool instead of a textual instructions document. Such a tool could guide requirements engineers in a wizard-based way, and give them the opportunity to collect the elicited requirements directly in a database. Furthermore, by using the stored information about conceptual relationships or assumptions, the tool could automatically inform the requirements engineers about syntactic incompleteness, inconsistencies, or violations of the given assumptions. Also, direct access to and alignment with reuse assets stored in a repository would be possible. We expect that this would further increase the effectiveness of elicitation.

8.2.4 Empirical Evaluation

With regard to the quality of our case study, its low validity (especially due to the small sample size and low control) can be considered as a weakness that motivates future work. We therefore recommend replicating our case study without its observed threats to validity, in order to get better insights into the strength and weaknesses of the tailoring in practice. Also interesting in this regard would be a comparison of the tailoring approach with ad-hoc tailoring done by SPL experts (as systematic candidate approaches do not exist yet). Such an experiment could provide evidence that the tailoring approach is not only effective, but also more effective and efficient than any other procedure.

With regard to the quality of the controlled experiment, we consider the observed threats to validity as acceptable for drawing the aforementioned conclusions. However, it would be interesting future work to see whether the experimental results remain stable when involving practitioners instead of students and when minimizing the observed threats such as the reliability of measures or the experimenter's expectancies. Thus, we plan to run an (improved) replicating study.

Finally, as our studies have not evaluated the overall impact of the thesis on AE efficiency, future work should particularly deal with this effect. So far, only the effectiveness of elicitation has been evaluated, but not the impact this effectiveness may have on development efficiency, which is ultimately the more relevant information for practitioners. So far, we can only show this improvement in an argumentative way. This means that we assume that the higher the realization fit achieved after elicitation, the higher the development efficiency (as fewer components have to be developed from scratch). Whether this claim holds true or whether there are too many other influencing factors has to be evaluated in a series of industrial case studies. As this was not achievable in the context of the thesis research, we plan to prepare and carry out such a concluding study in the future.

References

[ABB+02] Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Leitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J., Zettel, J.: Component-based Product Line Engineering with UML. Addison-Wesley, 2002

[ABQ+11] Alegria, J., Bastarrica, M., Quispe, A., Ochoa, S.: An MDE Approach to Software Tailoring. In: Proceedings of the SCSSP 2011. ACM, 2011

[Ada10] Adam, S.: Improving SPL-based Information System Development Through Tailored Requirements Processes. In: Proceedings of the Doctoral Symposium at the 18th IEEE International Conference on Requirements Engineering 2010. Sydney, 2010

[Ada12] Adam, S.: Providing SPL Knowledge to Requirements Engineers – A Template for Elicitation Instructions. In: Proceedings of the 18th REFSQ, LNCS 7195. Springer, 2012

[Ada11] Adam, S.: Towards Faster Application Engineering through Better Informed Elicitation – A Research Preview. In: Proceedings of the REFSQ 2011 Workshops, ICB-Research Report No. 44. Universität Duisburg-Essen, 2011

[Ada11b] Adam, S.: Higher Efficiency through Tailored Requirements Processes in Reuse-oriented Development. In: Proceedings of the REFSQ 2011 Workshops, ICB-Research Report No. 44. Universität Duisburg-Essen, 2011

[Ada11c] Adam, S.: Produktlinien-bewusste Anforderungserhebung durch maßgeschneiderte Erhebungsprozesse. Fachgruppentreffen RE der Gesellschaft für Informatik, 2011

[AD08] Adam, S., Doerr, J.: The Role of Service Abstraction and Service Variability and its Impact on Requirements Engineering for Service-oriented Systems. In: Proceedings of the 32nd IEEE International Conference on Computer Software and Applications. IEEE, 2008

[ADE09] Adam, S., Doerr, J., Eisenbarth, M.: Lessons learned from best practice-oriented process improvement in Requirements Engineering – A glance into current industrial RE application. In: Proceedings of 4th International Workshop on Requirement Engineering Education and Training. IEEE, 2009

[ADE+09] Adam, S., Doerr, J., Eisenbarth, M., Gross, A.: Using Task-oriented Requirements Engineering in Different Domains – Experiences with Application in Research and Industry. In: Proceedings of the IEEE International Requirements Engineering Conference. IEEE Computer Society, 2009

[ADE+10] Adam, S., Doerr, J., Ehresmann, M., Wenzel, P.: Incorporating SPL Knowledge into a Requirements Process for Information Systems – Architecture-driven Tailoring Approach. In: Proceedings of the 1st PLREQ Workshop at the REfSQ 2010. Essen, 2010

[AFC+05] Alves, C., Franch, X., Carvallo, J., Finkelstein, A.: Using Goals and Quality Models to Support the Matching Analysis During COTS Selection. In: Proceedings of ICCBSS, LNCS 3415. Springer, 2005

[AHH11] Arpinen, T., Hämäläinen, T., Hännikäinen, M.: Meta-Model and UML Profile for Requirements Management of Software and Embedded Systems. In: EURASIP Journal on Embedded Systems. 2011

[Ale05] Alexander, I.: A Taxonomy of Stakeholders – Human Roles in System Development. In: International Journal of Technology and Human Interaction, Vol. 1, 2005

[Alve03] Alves, C.: COTS-Based Requirements Engineering. In: Proceedings of Component-Based Software Quality, LNCS 2693. Springer, 2003

[ANA+10] Alves, V., Niu, N., Alves, C., Valenca, G.: Requirments Engineering for software product lines: A systematic literature review. In: Information and Software Technology. Elsevier, 2010

[ANT10] Adam, S., Naab, M., Trapp, M.: A Service-Oriented View on Business Processes and Supporting Applications. In: Proceedings of 11th International Workshop on BPMDS, LNBIP 50. Springer, 2010

[AQG12] Analytic Quality Glossary: Effectiveness. http://www.qualityresearchinternational.com/glossary/effectiveness.htm (last visited: 2012-02-17)

[ARC08] Adam, S., Riegel, N., Carbon, R.: . Requirements Engineering Approach for the Identification of Innovative Services. Fraunhofer IESE Report 037.08/E. Fraunhofer IESE, 2008

[ARG+12] Adam, S., Riegel, N., Gross, A., Uenalan, O., Darting, S.: A Conceptual Foundation of Requirements Engineering for Business Information Systems. In: Proceedings of BPMDS, LNBIP. Springer, 2012

[Ars+07] Arsanjani, A., Zhang, L., Ellis, M., Allam, A., Channabasavaiah: S3 – A Service-oriented Reference Architecture. In: IT Pro, May / June 2007. IEEE Computer Society, 2007

[AUY+07] Aoyama, K., Ugai, T., Yamada, S., Obata, A.: Extraction of Viewpoints for Eliciting Customer's Requirements based on Analysis of Specification Change Records. In: Proceedings of the 14th Asia-Pacific Software Engineering Conference. IEEE Computer Society, 2007

[Bas93] Basili, V.: The Experimental Paradigm in Software Engineering. In: Proceedings of Dagstuhl-Workshop, LNCS 706. Springer, 1993

[BBG+00] Baum, L., Becker, M., Geyer, L., Molter, G.: Mapping Requirements to Reusable Components using Design Spaces. In: Proceedings of IEEE International Requirements Engineering Conference. IEEE Computer Society, 2000

[BC12] Business Dictionary. http://www.businessdictionary.com, last visited: 2012-01-12

[BCK03] Bass, L., Clements, P., Kazman. R.: Software Architecture in Practice, Second Edition, SEI Series in Software Engineering, Addison-Wesley, 2003.

[BCR94] Basili, V.R., Caldiera, G., Rombach, H. D.: Goal Question Metric Paradigm, Encyclopedia of Software Engineering, Volume 1, pp. 528-532. John Wiley & Sons, 1994

[Ber10] Berry, D.: Advice for Finishing that Damn Ph.D. http://se.uwaterloo.ca/~dberry/FTP_SITE/lecture.slides/finishing.phd.talk.pdf. (visited: 2010-11-11)

[BGM+00] Bayer, J., Gacek, C., Muthig, D., Widen, T.: PuLSE-I: Deriving Instances from a Product Line Infrastructure. In: Proceedings 7th IEEE International Conference and Workshop on the Engineering of Computer-based Systems. IEEE Computer Society, 2000

[BHL+06] Bühne, S., Halmans, G., Lauenroth, K., Pohl, K.: Scenario-Based Application Requirements Engineering: In: Software Product Lines. Springer, 2006

[BI12] Bundesministerium des Inneren: Das V-Modell XT. http://www.v-modell-xt.de/ (last visited: 2012-01-27)

[CA07] Cheng, B., Atlee, J.: Research Directions in Requirements Engineering. In: Proceedings of Future of Software Engineering (FOSE). IEEE Computer Society, 2007

[Car08] Carbon, R.: Improving the Production Capability of Product Line Organizations by Architectural Design for Producibility. In: Proceedings of Doctoral Symposium @ SPLC 2008

[Car11] Carbon, R.: Architecture-Centric Software Producibility Analysis. PhD Theses in Experimental Software Engineering, Vol. 38. Fraunhofer Verlag, 2011

[CDS+05] Ceron, R., Duenas, J., Serrano, E., Capilla, R.: A meta-model for requirements engineering in system family context for software process improvement using CMMI. In: Profes 2005, LNCS 3547. Springer, 2005

[CFM+02] Casati, F., Fugini, M., Mirbel, I., Pernici, B.: WIRES: A Methodology for Developing Workflow Applications. In: Requirements Engineering, Vol. 7. Springer, 2002

[CN01] Clements, P., Northrop, L.: Software Product Lines: Patterns and Practice. Addison Wesley, 2001

[Coc00] Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, 2000

[Dav93] Davis, A.: Software Requirements – Objects, Functions & States. Prentice Hall PTR, 1993

[DKL09] Duszynsky, S., Knodel, J., Lindvall, M.: SAVE: Software Architecture Visualization and Evaluation. In: Proceedings of 13th European Conference on Software Maintenance and Reeengineering. IEEE Computer Society, 2009

[DPK04] Doerr, J., Paech, B., Koehler, M.: Requirements Engineering Process Improvement Based on an Information Model. In: Proceedings of the 12th International Requirements Engineering Conference. IEEE Computer Society, 2004

[DSB05] Deelstra, S., Sinnema, M, Bosch, J.: Product derivation in software product families: a case study. In: The Journal of Systems and Software, vol. 74. Elsevier, 2005

[DS07] Djebbi, O., Salinesi, C.: RED-PL, a Method for Deriving Product Requirements from a Product Line Requirements Model. In: Proceedings of CAiSE 2007. Springer, 2007

[FD12] Free Dictionary. http://www.thefreedictionary.com/, last visited: 2012-01-12

[FGM07] Fricker, S., Gorschek, T., Myllyperkiö, P.: Handshaking Between Software Projects and Stakeholders Using Implementation Proposals. In: Proceedings of REFSQ 2007, LNCS 4542. Springer, 2007

[FMS+10] Ferrari, R., Madhavji, N., Sudmann, Ol., Henke, C., Geilser, J.: Requirements Engineering Decisions in the Context of an Existing Architecture: A Case Study of a Prototypical Project. In: Proceedings of the 18th IEEE International Requirements Engineering Conference. IEEE Computer Society, 2010

[GD07] González, J.L., Diáz, J.S.: Business process-driven requirements engineering: a goal-based approach. Proceedings of the BPMDS Workshop, Trondheim, Norway, 2007

[GKB08] Goknil, A., Kurtev, I., Berg, K.: A Metamodeling Approach for Reasoning about Requirements. In: Model-Driven Architecture – Foundation and Application, LNCS 5095. Springer, 2008

[GP07] Guelfi, N., Perrouin, G.: A Flexible Requirements Analysis Approach for Software Product Lines. In: Proceedings of RefSQ 2007, LNCS 4542. Springer, 2007

[GPW06] Gordijn, J., Petit, M., Wieringa, R.: Understanding Business Strategies of Networked Value Constallations Using Goal- and Value Modeling. In: Proceedings of 14th IEEE International Requirements Engineering Conference. IEEE Computer Society, 2006

[Gro10] Gross, A.: Perspective-based Specification of Efficiently and Effectively Usable Requirements Documents. Doctoral Symposium RE'10. Sydney (2010)

[GRT00] Gzara, L, Rieu, D., Tollenaere, M.: Patterns Approach to Product Information System Engineering. In: Requirements Engineering, Vol. 5. Springer, 2000

[Gom04] Gomaa, H.: Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures. Addison-Wesley, 2004

[Gul04] Gulla, J.A.: Understanding Requirements in Enterprise System Projects. In: Proceedings of the 12th IEEE International Requirements Engineering Conference. IEEE Computer Society, 2004

[Hay03] Hay, D.: Requirements Analysis – From Business Views to Architecture. Prentice Hall PTR, 2003

[HP03] Halmans, G., Pohl, K.: Communicating the variability of a software-product family to customers. In: Software and System Modeling 2003/2. Springer, 2003

[HPS08] Halmans, G., Pohl, K., Sikora, E.: Documenting Application-Specific Adaptations in Software Product Line Engineering. In: Proceedings of CAiSE 2008. Springer, 2008

[IEEE98a] IEEE Computer Society: IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. IEEE, 1998

[IEEE98b] IEEE Computer Society: IEEE 1233-1998 - IEEE Guide for Developing System Requirements Specifications Description. IEEE, 1998

[IEEE98c] IEEE Computer Society: IEEE 1362-1998 - Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document Description. IEEE, 1998

[IEEE98d] IEEE Standard 1058-1998 for Software Project Management Plans

[JEF04] Jiang, L., Eberlein, A., Far, B.: A Methodology for Requirements Engineering Process Development. In: Proceedings of the 11th International Conference and Workshop on the Engineering of Computer-Based Systems. IEEE Computer Society, 2004

[JJM09] Jeusfeld, M, Jarke, M., Mylopoulos, J.: Metamodeling for Method tailoring. The MIT Press, 2009.

[JKL+06] John, I., Knodel, J., Lehner, T., Muthig, D.: A Practical Guide to Product Line Scoping. In: Proceedings of the 10th Software Product Line Conference. IEEE Computer Society, 2006

[Joh10] John, I.: Pattern-based Documentation Analysis for Software Product Lines. PhD Theses in Experimental Software Engineering, Vol. 30. Fraunhofer Verlag, 2010

[KAP+04] Kazhamiakin, R., Pistore, M., Roveri, M.: A Framework for Integrating Business Processes and Business Requirements. In: Proceedings of the 8th IEEE International Enterprise Distributed Object Computing Conference. IEEE Society, 2004

[Kit04] Kitchenham, B.: Procedures for Performing Systematic Reviews. In: Keele University Technical Report TR/SE-0401. Keele University, 2004

[Kne01] Knethen, A.: Change-Oriented Requirements Traceability. Support for Evolution of Embedded Systems. PhD Theses in Experimental Software Engineering, Vol. 9. Fraunhofer IRB, 2001

[Kru00] Kruchten, P.: The Rational Unified Process. Addison-Wesley, 2000

[KS98] Kontonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. Wiley & Sons, 1998

[Lam04] Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice. In: Proceedings of the 12th IEEE International Requirements Engineering Conference. IEEE, 2004

[Lam09] Lamsweerde, A.: *Requirements Engineering: From System Goals to UML Models to Software Specifications.* John Wiley & Sons, 2009

[Lau02] Lauesen, S.: Software Requirements – Styles and Techniques. Addison-Wesley, 2002

[LJB98] Lam, W., Jones, S., Britton, C.: Technology Transfer for Reuse: A Management Model and Process Improvement Framework. In: Proceedings of the IEEE International Requirements Engineering Conference. IEEE Computer Society, 1998

[LLC04] Laguna, M., Lopez, O., Crespo, Y.: Reuse, Standardization, and Transformation of Requirements. In: Proceedings of 8th International Conference on Software Reuse, LNCS 3107. Springer, 2004

[LW00] Leffingwell, D, Widrig, D.: Managing Software Requirements. Addison-Wesley, 2000

[MA02] Muthig, D., Atkinson, C.: Model-driven Product Line Architectures. In: Proceedings of the Software Product Line Conference, LNCS 2379. Springer, 2002

[Mut02] Muthig, D.: A Light-weight Approach Facilitating an Evolutionary Transition Towards Software Product Lines. PhD Theses in Experimental Software Engineering, Vol. 11. Fraunhofer IRB, 2002

[Naa09] Naab, M.: Improving the Flexibility of SOA-Based Information Systems by Adopting Practices from Product Line Engineering. Doctoral Symposium of SPLC, 2009

[NC07] Northrop, L., Clements, P.: A Framework for Software Product Line Practice, Version 5.0, Software Engineering Institute, 2007

[NE00] Nuseibeh, B., Easterbrook, S.: Requirements Engineering – A Roadmap. In: Proceedings of the IEEE International Conference on Software Engineering (ICSE). IEEE Computer Society, 2000

[NM10] Naab, M., Muthig, D.: Designing Flexible Architectures for Product Lines Dominated by Open Variability. IESE Report IESE-062.10/E. Fraunhofer IESE, 2010

[OMG11] Onject Management Grouo: Unified Modeling Language. http://www.uml.org (last-visited: 2011-11-30)

[OMG12] Object Management Group: Meta Object Facility. http://www.omg.org/mof/ (last visited: 2012-02-08)

[OMG12b] Object Management Group: BPMN Specification. http://www.bpmn.org (last visited: 2012-02-10)

[ORR+09] O'Leary, P., Rabiser, R., Richardson, I., Thiel, S.: Important Issues and Key Activities in Product Derivation: Experience from Two Independent Research Projects. In: Proceedings of the Software Product Line Conference 2009. SEI, 2009

[PK04] Paech, B., Kohler, K.: Task-driven Requirements in Object-oriented Development. In: Perspectives on Software Engineering. Kluwer Academic Publishers, 2004

[PKG+08] Perrouin, G., Klein, J., Guelfi, N., Jezequel, J.: Reconciling Automation and Flexibility in Product Derivation. In: Proceedings of 12th Software Product Line Conference. IEEE Computer Society, 2008

[Poh07] Pohl, K.: Requirements Engineering – Grundlagen, Prinzipien, Techniken. dpunkt.verlag, 2007

[PTG+06] Peffers, K., Tuunanan T., Gengler, C., Rossi, M., Hui, W., Virtanen, V., Bragge, J.: The Design Science Research Process: A Model for Producing and Presenting Information System Research. In: Proceedings of DESRIST 2006. CGU, 2006

[RAG11] Riegel, N., Adam, S., Gross, A.: Adressing Requirements Engineering Challenges in the Context of Emergent Systems. In: Proceedings of Workshop on Requirements Engineering for Systems, Services, and Systems-of-Systems. IEEE, 2011

[RB01] Reason P., Bradbury H. (eds.): Handbook of Action Research. Sage Publications, 2001

[RGD07] Rabiser, R., Grünbacher, P., Dhungana, D.: Supporting Product Derivation by Adapting and Augmenting Variability Models. In: Proceedings of the 11th International Software Product Line Conference. IEEE Computer Society, 2007

[RGD09] Rabiser, R., Grünbacher, P., Dhungana, D.: Requirements for product derivation support: Results from a systematic literature review and an expert survey. In: Information and Sofware Technology. Elsevier, 2009

[RD07] Rabiser, R., Dhungana, D.: Integrated Support for Product Configuration and Requirements Engineering in Product Derivation. In: Proceedings of 33rd Conference on Software Engineering and Advanced Applications. IEEE Computer Society, 2007

[RR99] Robertson, S., Robertson, J.: Mastering the Requirements Process. Addison-Wesley, 1999

[RSB98] Rolland, C., Souveyet, C., Ben Achour, C.: Guiding Goal Modeling using Scenarios. In: IEEE Transactions on Software Engineering. IEEE Computer Society, 1998

[Rup07] Rupp, C.: Requirements Engineering und Management – Professionelle, iterative Anforderungsanalyse für die Praxis. Hanser, 2007

[RW05] Rozanski, N., Woods, E.: Software System Architecture – Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley, 2005

[Sch01] Scheer, A.-W.: ARIS – Modellierungsmethoden, Metamodelle, Anwendungen. 4. Auflage. Springer, 2001

[Sch03] Schmid, K.: Planning Software Reuse - A Disciplined Scoping Approach for Software Product Lines. PhD Theses in Experimental Software Engineering, Vol. 12. Fraunhofer IRB, 2003

[Sch95] Scheer, A.-W.: Wirtschaftsinformatik – Referenzmodelle für industrielle Geschäftsprozesse. Springer, 1995

[SDH06] Sinnema, M., Deelstra, S., Hoekstra, P.: The COVAMOF Derivation Process. In: International Conference on Software Reuse. Springer, 2006

[SEI08] Software Engineering Institute CarnegieMellon: Software Product Lines. http://www.sei.cmu.edu/productlines/ (visited: 2008-10-10)

[SGD+01] Soffer, P., Golany, B., Dori, D., Wand, Y.: Modelling Off-the-Shelf Information System requirements: An Ontological Approach. In: Requirements Engineering, Vol. 6. Springer, 2001

[Som04] Sommerville, I.: Software Engineering – Seventh Edition. Addison-Wesley, 2004

[Som05] Sommerville, I.: Integrated Requirements Engineering: A Tutorial. In: IEEE Software, January / February 2005. IEEE Computer Society, 2005

[SLS+09] Sommerville, I., Lock, R., Storer, T., Dobson, J.: Deriving Information Requirements from Responsibility Models. In: Proceeedings of CAiSE, LNCS 5565. Springer, 2009

[SS97] Sommerville, I., Sawyer, P.: Requirements Engineering – A good practice guide. John Wiley, 1997

[ST05] Sindhgatta, R., Thonse, S.: Functional and Non-functional Requirements Specification for Enterprise Applications. In: Proceedings of PROFES 2005, LNCS 3547. Springer, 2005

[SW04] Soffer, P., Wand, Y.: Goal-Driven Analysis of Process Model Validity. In: Advanced Information Systems Engineering, Bd. 3084. Springer, 2004

[SW11] Scheinholtz, L., Wilmont, I.: Interview Patterns for Requirements Elicitation. In: Proceedings of REFSQ 2011, LNCS 6606. Springer, 2011

[Swe04] IEEE's Software Engineering Body of Knowledge – 2004 Version. http://www.swebok.org, IEEE Computer Society, 2004

[UG12] Usability.gov: Card Sorting. http://www.usability.gov/methods/ design_site/cardsort.html (last visited: 2012-01-29)

[VMT07] Vicente-Chicote, C., Moros, B., Toval, A.: REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting. IN: Journal of Object Technology, Vol. 6, No. 9. ETH Zurich, 2007

[Wie05] Wiegers, K.: Software Requirements – Second Edition (German Version). Microsoft Press, 2005

[WJR+07] Wegmann, A., Julia, P., Regev, G., Perroud, O., Rychkova, I.: Early Requirements and Business-IT Alignment with SEAM for Business. In: 15th IEEE International Requirements Engineering Conference. IEEE Computer Society, 2007

[WW92] Wand, Y., Weber, R.: An ontological model of an information system. In: IEEE Transactions on Software Engineering. IEEE Computer Society, 1992

[ZoGe03] Zowghi, D., Gervasi, V.: On the interplay between consistency, completeness and correctness in requirements evolution. In: Journal of Information and Software Technology, Volume 45. Elsevir, 2003

[Zim87] Zimmermann, D.: Comparative Power of Student T Test and Mann-Whitney U Test for Unequal Sample Sizes and Variances. In: Journal of Experimental Education. Heldref, 1987

Appendix

Appendix A: Review Protocols

Background	The survey is needed to find related work in the area of requirements elicitation instructions.
Research Question	Which work exists that aims at providing effective guidance for requirements elicitation?
Search Strategy	Search String: - (("requirements elicitation" OR "requirements gathering" OR "requirements collection" OR "elicitation interview" OR "requirements interview" OR "requirements negotiation" OR "requirements identification" OR "requirements determination") AND (instruction OR guideline OR checklist OR guidance OR procedure OR strategy)) Primary Resources: - IEEE Xplore - ACM - Elsevier - Already know literature Search Criteria: - only software engineering or related fields such as business management - no standards, but all other publication channels (e.g., conference proceedings, dissertation, books, journals, etc.) - search terms only in meta-data (title, abstract, etc.), at least for primary search - not older than 15 years (at least for primary search) Search Approach: 1. Search in the resources 2. Exclude non-fitting papers 3. Search for further papers in the reference list of the fitting paper 4. Proceed with step 2
Study Selection Criteria and Process	Exclusion Criteria: - does not deal with requirements engineering (but just uses this term somewhere) - is focused on the elicitation of only one specific type of requirements (e.g., security requirements) - does not present guidelines but just mentions them - describes an experience report / case study and no method
Study Quality Assessment	1. Does the guideline provide precise instructions (how-to descriptions)? 2. Does the guideline propose a clear elicitation order? 3. Is the content of the guideline customizable (i.e., can be adapted based on information needs)? 4. Is the guideline modularized and allows a separation of concerns? 5. Does the guideline reflects RE best practices (e.g., which stakeholders are to be involved)? 6. Does the guideline provide information about capabilities, needs, and constraints of downstream development activities? 7. Is the guideline introduced in the context of a reuse-based development approach (with reuse)? 8. Does the guideline fit into state-of-the-art RE? 9. Is the guideline empirically validated? 10. Is the guideline applicable in the IS domain?

Background	The survey is needed to find related work in the area of application engineering requirements engineering.
Research Question	Which work exists that aims at providing an effective requirements engineering for the application engineering phase?
Search Strategy	Search String: - ("requirements engineering" OR "elicitation") AND ("application engineering" OR derivation OR instantiation OR "development with reuse" OR customization OR configuration) AND ("product line" OR "product family" OR reuse OR platform) Resources: - IEEE Xplore - ACM - Elsevier - Already know literature Search Criteria: - only software engineering or related fields such as business management - no standards, but all other publication channels (e.g., conference proceedings, dissertation, books, journals, etc.) - search terms only in meta-data (title, abstract, etc.), at least for primary search - not older than 15 years (at least for primary search) Search Approach: 1. Search in the resources 2. Exclude non-fitting papers 3. Search for further papers in the reference list of the fitting paper 4. Proceed with step 2
Study Selection Criteria and Process	Exclusion Criteria: - does not deal with requirements engineering (but just uses this term somewhere) - does not describe how to apply RE in the "development with reuse" context - describes an isolated technique and not a complete approach
Study Quality Assessment	1. Does the approach support a problem-oriented elicitation? 2. Does the approach explain elicitation and negotiation activities? 3. Does the approach explain how to deal with customer-specific requirements beyond the predefined variants? 4. Does the approach explain how to align customer requirements with reuse capabilities? 5. Is the approach customizable based on the given reuse asset base? 6. Is the approach empirically validated? 7. Does the approach provide precise guidance? 8. Is the approach applicable in the IS domain?

Background	The survey is needed to find related work in the area of requirements process tailoring.
Research Question	Which work exists that aims at providing an effective tailoring or reengineering of requirements engineering processes?
Search Strategy	Search String: - ((("requirements engineering process" OR "requirements process" OR "elicitation process") AND (tailoring OR customization OR adaptation OR improvement OR reengineering OR definition)) OR (requirements AND ("product line" OR reuse) AND (reengineering OR extraction OR incorporation))) Resources: - IEEE Xplore - ACM - Elsevier - Already know literature Search Criteria: - only software engineering or related fields such as business management - no standards, but all other publication channels (e.g., conference proceedings, dissertation, books, journals, etc.) - search terms only in meta-data (title, abstract, etc.), at least for primary search - not older than 15 years (at least primary paper) Search Approach: 1. Search in the resources 2. Exclude non-fitting papers 3. Search for further papers in the reference list of the fitting paper 4. Proceed with step 2
Study Selection Criteria and Process	Exclusion Criteria: - does not deal with requirements engineering processes definition or requirements reengineering (but just uses this term somewhere)
Study Quality Assessment	1. Is the approach applicable in the IS domain? 2. Does the approach provide precise guidance? 3. Is the approach empirically validated? 4. Does the approach address the extraction and reflection of information needs? 5. Does the approach address the extraction and reflection of constraints? 6. Does the approach consider the development strategy? 7. Does the approach fit into state-of-the-art RE and consider best practice? 8. Does the approach leads to a requirements process?

Appendix B: Requirements on ARE Instructions

	Requirement	Expert Confirmation
General Nature	**R.N.1.** An instruction should allow requirements engineers to deviate from it in case of need.	Strong
	R.N.2. An instruction should provide clear how-to guidance (i.e., it should clearly mention a sequence of steps to be carried out during elicitation).	Normal
	R.N.3. An instruction should explain how to proceed with the elicited requirements (e.g., visualizing, describing, classifying, decomposing, refactoring, referencing, …)	Normal
	R.N.4. An instruction should be specific, i.e., customized for a certain development or project context.	Normal
	R.N.5. An instruction should enable less or average experienced requirements engineers to elicit quite a good set of requirements.	Strong
Structure	**R.S.1.** An instruction should make clear in which order certain elicitation steps should be performed best.	Normal
	R.S.2. An instruction should be modularized.	Normal
	R.S.3. An instruction should make clear at which point in time breaks between elicitation sessions could be done best.	Normal
	R.S.4. An instruction should provide good indications to know when the elicitation is finished.	Strong
Content	**R.C.1.** An instruction should mention the issues which are relevant to be discussed with the stakeholders.	Strong
	R.C.2. An instruction should make clear until which point in time certain issues have to be discussed.	Normal
	R.C.3. An instruction should name the typical stakeholders needed in a certain elicitation step.	Strong
	R.C.4. An instruction should inform about the concrete information to be elicited with regard to a certain issue (e.g., for which details do I really need to ask).	Strong
	R.C.5. An instruction should inform against which criteria the elicited requirements are to be checked.	Normal
	R.C.6. An instruction should make clear about which issues a discussion with the stakeholders is unnecessary (e.g.because no one in the subsequent development process will care about them).	Strong
	R.C.7. An instruction should inform which requirements are implemented by default anyway (e.g., common realization decisions / features).	Normal
	R.C.8. An instruction should inform whether requirements concerning a certain issue are restricted by architectural constraints.	Normal
	R.C.9. An instruction should make clear which properties / assumptions a requirement concerning a certain issue must fulfill in order to be implementable.	Normal
	R.C.10. An instruction should inform about those technical issues from the solution space that might influence the feasibility of requirements.	Strong
	R.C.11. An instruction should represent architectural / technical constraints in a non-technical language.	Normal
	R.C.12. An instruction should inform the requirements engineers about conceptual dependencies between issues (e.g., a use case is related to the user role performing the use case).	Strong

Appendix C: ARE Instructions Generation Algorithm (VB Code)

```
Option Compare Database

Dim objWord
Dim objDoc
Dim range
Dim strFile As String


' Procedure for building ARE Instruction Document
Private Sub-Command0_Click()

' Variables

    On Error Resume Next

    ' Create path to the Instruction document
    strFile = CurrentProject.Path & "\ARE_Elicitation_Instruction.doc"

   ' Get existing instance of Word if it exists.
   Set objWord = GetObject(, "Word.Application")
   If Err <> 0 Then
      ' If GetObject fails, then use CreateObject instead.
      Set objWord = CreateObject("Word.Application")
   End If

   ' Make Word Visible
   objWord.Visible = True

   ' Brief Waiting Time
   Wscript.Sleep 500

   ' Add a new document.
   objWord.Documents.Add

   ' Add introduction text to document
   objWord.ActiveDocument.Content.InsertAfter ("ARE Elicitation Instruction")
   objWord.ActiveDocument.Paragraphs.Last.Style = objWord.ActiveDocument.Styles("Heading
1")
   objWord.ActiveDocument.Paragraphs.Last.Format.SpaceAfter = 5
   objWord.ActiveDocument.Content.InsertParagraphAfter

   ' Set variables for database query
    Dim dbs As Database
    Dim rst As DAO.Recordset
    Dim strSQL As String
    Dim i As Integer
    On Error Resume Next

    ' Make Database Query for getting general information about SPL
    Set dbs = CurrentDb
    strSQL = "SELECT TOP 1 * FROM [SPL Characterization]"
    Set rst = dbs.OpenRecordset(strSQL)

    ' Iterate over results
    If Not (rst.EOF And rst.BOF) Then
        rst.MoveFirst
        objWord.ActiveDocument.Content.InsertAfter ("This document includes precise process
instruction on how to elicit, negotiate and specify requirements concerning a system de-
rived from the " & rst![SPL name] & " Product Line. ")
        objWord.ActiveDocument.Content.InsertAfter ("For this purpose, the document pro-
vides a sequence of elicitation steps to be carried out in the described order. The overall
aim of this document is to provide requirements engineers with answers to the following
questions:")
```

```
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("1) With regard to which issues are
requirements needed to make development decisions?")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("2) Which detailed information must be
elicited (and specified) about each of these issues?")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("3) What can / should / must be reused
when specifying requirements? And what must be elicited from scratch?")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("4) Which constraints are given by the
product line architecture?")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("5) Which stakeholders should be in-
volved in the different elicitation steps?")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("6) Which notations could support the
elicitation?")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("7) What is a recommended order in
which the important issues should be discussed for saving rework?")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("8) How are requirements related and
how can these relationships be used for achieving completeness rather constructively?")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("To answer all these questions, the
following instructions have been tailored individually based on the constraints and needs
of the underlying software product line (SPL) and the development organization. ")
        objWord.ActiveDocument.Content.InsertAfter ("The overall purpose of system derived
from this SPL is " & rst![purpose] & " ")
        objWord.ActiveDocument.Content.InsertAfter ("in the domain of " & rst![business
domain] & ". ")
        objWord.ActiveDocument.Content.InsertAfter ("While typical customers are " &
rst![Customers] & ", typical users include " & rst![Users] & ". ")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("The main benefits, systems from this
SPL provide, are that " & rst![Benefits] & ". ")
        objWord.ActiveDocument.Content.InsertAfter ("To make this happen, the following key
features are important: " & rst![Key Features] & ". ")
        objWord.ActiveDocument.Content.InsertAfter ("However, the following limitations
have to be considered: " & rst![Constraints and limitations] & ". ")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("Systems derived from this SPL are
integrated into their usage environment as follows: " & rst![environmental integration] &
". ")
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("The SPL Specification that provides
you with further and elicitation-relevant information can be found in " &
rst![Specification source] & ". ")
    End If

    Set rst = Nothing

   ' Identify the issues to be included
   dbs.Execute ("UPDATE Issues SET [to be included] = true")
   dbs.Execute ("UPDATE Issues SET [to be included] = false WHERE " & _
               "([type]=3 AND [to be documented]=false) OR " & _
               "(type=2 AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] WHERE
[issue1]=[Issues].[id] AND [type]<>3)) OR " & _
              "([to be documented]=false AND EXISTS (SELECT * FROM [Conceptual Relation-
ships] WHERE [issue1]=[Issues].[id] AND [type]=3) AND NOT EXISTS (SELECT * FROM [Conceptual
Relationships] WHERE [issue1]=[Issues].[id] AND [type]<>3))")

   ' Identify the optional issues
   dbs.Execute ("UPDATE Issues SET optional=false")
   dbs.Execute ("UPDATE Issues SET [optional]=true WHERE EXISTS (SELECT DISTINCT * " & _
                "FROM (SELECT DA.[activity name], DA.[condition], DA.[optional], [affected
issue] FROM (SELECT * FROM [Development Activities] AS DA, Decisions AS D WHERE
DA.ID=D.[made during] And DA.optional=true)  AS temp INNER JOIN [Information Needs] AS I1
ON I1.[ID] = temp.[Information needs].Value) " & _
                "WHERE [issue name]=I1.[affected issue])")
```

```
   dbs.Execute ("UPDATE Issues SET [optional]=false WHERE EXISTS (SELECT DISTINCT * " & _
                "FROM (SELECT DA.[activity name], DA.[condition], DA.[optional], [affected
issue] FROM (SELECT * FROM [Development Activities] AS DA, Decisions AS D WHERE
DA.ID=D.[made during] And DA.optional=false)  AS temp INNER JOIN [Information Needs] AS I1
ON I1.[ID] = temp.[Information needs].Value) " & _
                "WHERE [issue name]=I1.[affected issue])")


   ' Start the actual instruction generation
   dbs.Execute ("DELETE FROM [Processed Issues]")
   dbs.Close
   Call GenerateMilestoneSections

   ' Save the document
   objWord.ActiveDocument.SaveAs FileName:=strFile

   ' Exit Word.
   objWord.Quit

   ' Clear object memory
   Set objWord = Nothing

End Sub


' Generate the milestone sections
Sub-GenerateMilestoneSections()

    ' Set variables
    Dim dbs As Database
    Dim rst As DAO.Recordset
    Dim strSQL As String
    Dim i As Integer
    On Error Resume Next

    Set dbs = CurrentDb

    ' Make Database Query
    strSQL = "SELECT * FROM [Development Phases] WHERE [predecessor] IS NULL OR [predeces-
sor]<1"
    Set rst = dbs.OpenRecordset(strSQL)

    ' Iterate over results
    If Not (rst.EOF And rst.BOF) Then

        Do
            rst.MoveFirst
            i = i + 1
            objWord.ActiveDocument.Sections.Add
            objWord.ActiveDocument.Content.InsertAfter ("Requirements to be elicited for "
& rst![phase name])
            objWord.ActiveDocument.Paragraphs.Last.Style = ob-
jWord.ActiveDocument.Styles("Heading 1")
            objWord.ActiveDocument.Content.InsertParagraphAfter
            objWord.ActiveDocument.Content.InsertAfter (rst![purpose] & ".")
            objWord.ActiveDocument.Content.InsertParagraphAfter
            objWord.ActiveDocument.Content.InsertAfter ("In order to do the development
work in this phase, requirements concerning all the issues mentioned below are needed by
the engineers for making corresponding design or development decisions. ")
            objWord.ActiveDocument.Content.InsertAfter ("Hence, it is indispensible that
these requirements are elicited before this development phase can start. To make this hap-
pen, we strongly recommend performing the steps in the mentioned order as there are depend-
encies between the addressed issues that may impact the elicitation. ")
            objWord.ActiveDocument.Content.InsertAfter ("In this regard, we describe wheth-
er and in which form artifacts have to be produced for the corresponding requirements. ")
            objWord.ActiveDocument.Content.InsertParagraphAfter
            objWord.ActiveDocument.Content.InsertAfter ("However, before you start, plan
which steps you would like to perform during one common elicitation session (interview /
workshop). For this purpose, consider the stakeholders to be involved in these steps as
well as the expected effort and the dependencies between the issues. ")
```

```
            ' Checks whether phase is iterative
            If (rst![iterative phase] = True) Then
                objWord.ActiveDocument.Content.InsertParagraphAfter
                objWord.ActiveDocument.Content.InsertAfter ("Important hint: As the phase
is iterative, this section may be repeated several times.")
                Call Format("Important hint: As the phase is iterative, this section may be
repeated several times.", 0, False, False, True)
                Call Format("Important hint:", 0, True, True, False)
                objWord.ActiveDocument.Paragraphs.Last.Format.SpaceBefore = 5
            End If

            GenerateIssueSections (rst![ID])

            Set rst = dbs.OpenRecordset("SELECT * FROM [Development Phases] WHERE [prede-
cessor]=" & rst![ID])

        Loop Until (rst.EOF Or rst.BOF)

    End If

    dbs.Close
    Set rst = Nothing

End Sub


' Generate the issue sections
Sub-GenerateIssueSections(phaseId As Integer)

' Set variables
    Dim dbs As Database
    Dim rst As DAO.Recordset
    Dim strSQL As String
    Dim i As Integer
    Dim help As Integer
    i = 0
    help = 0

    On Error Resume Next

    ' Make database query
    Set dbs = CurrentDb

    ' 1) Discuss all issues in a random order that do not have any relationship to another
issue.
    strSQL = "SELECT * FROM [Issues] i WHERE [to be discussed before start of]=" & phaseId
& " AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] WHERE issue1=i.id OR is-
sue2=i.id)"
    Set rst = dbs.OpenRecordset(strSQL)

    If Not (rst.EOF And rst.BOF) Then
        rst.MoveFirst
        Do

            If (rst![to be included] = True) Then

                i = i + 1

                ' Start generation
                Call GenerateIssueSectionInternals(rst, i)

            End If

            ' Insert processed issue
            dbs.Execute ("INSERT INTO [Processed Issues] ([processed issue]) VALUES (" &
rst![ID] & ")")

            ' Iterate
            rst.MoveNext

        Loop Until rst.EOF
```

```
    End If

    Set rst = Nothing

    ' Discuss all issues in a random order that are not required by, not contained in, not
influenced by, and not a specialization of another issue.
    strSQL = "SELECT * FROM [Issues] i WHERE [to be discussed before start of]=" & phaseId
& " AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] WHERE issue1=i.id) AND NOT
EXISTS (SELECT * FROM [Processed Issues] WHERE [processed issue]=i.id)"
    Set rst = dbs.OpenRecordset(strSQL)

    If Not (rst.EOF And rst.BOF) Then
        rst.MoveFirst
        Do

            If (rst![to be included] = True) Then

                i = i + 1

                ' Start generation
                Call GenerateIssueSectionInternals(rst, i)

            End If

            ' Insert processed issue
            dbs.Execute ("INSERT INTO [Processed Issues] ([processed issue]) VALUES (" &
rst![ID] & ")")

            ' Iterate
            rst.MoveNext

        Loop Until rst.EOF

    ' If there is none, discuss at least the issues in a random order that are influenced
by an already discussed issue, but that have no further required / contained / influenced /
specialization-relationships.
    Else
        Set rst = Nothing
        strSQL = "SELECT * FROM [Issues] i WHERE [to be discussed before start of]=" &
phaseId & " AND EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id
AND cr.type=4 AND EXISTS (SELECT * FROM [Processed Issues] WHERE [processed is-
sue]=cr.issue2)) AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE
cr.issue1=i.id AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed is-
sue]=cr.issue2))"
        Set rst = dbs.OpenRecordset(strSQL)

        If Not (rst.EOF And rst.BOF) Then
            rst.MoveFirst
            Do

                If (rst![to be included] = True) Then

                    i = i + 1

                    ' Start generation
                    Call GenerateIssueSectionInternals(rst, i)

                End If

                ' Insert processed issue
                dbs.Execute ("INSERT INTO [Processed Issues] ([processed issue]) VALUES ("
& rst![ID] & ")")

                ' Iterate
                rst.MoveNext

            Loop Until rst.EOF
        End If

    End If
```

```
    Set rst = Nothing

    ' Discuss all issues that are required by, contained in, influenced by, or a specialization of an already discussed issue, and that are neither required by, contained in, influenced by, nor a specialization of an issue that has not been discussed yet.

    ' If there is more than one, discuss them in the following order.
    ' If there is more than one in each sub-order, discuss them in the order in which the specialized / containing / requiring / influencing issue has appeared.
    ' Adapt the order continuously and repeat this procedure until all issues related to a certain milestone have been discussed.


    ' Iterates over all remaining issues of this phase
    Do
        Set rst = Nothing

        help = help + 1

        ' 1) issues that specialize an already discussed one,
        Do
            strSQL = "SELECT * FROM [Issues] i WHERE [to be discussed before start of]=" & phaseId & " AND EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id AND cr.type=3 AND EXISTS (SELECT * FROM [Processed Issues] WHERE [processed issue]=cr.issue2)) AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed issue]=i.id) AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] cr2 WHERE issue2=i.id AND EXISTS(SELECT * FROM [Processed Issues] WHERE [processed issue]=cr2.issue1)) AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id AND cr.issue2<>i.id AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed issue]=cr.issue2))"
            Set rst = dbs.OpenRecordset(strSQL)

            If Not (rst.EOF And rst.BOF) Then
                rst.MoveFirst

                If (rst![to be included] = True) Then

                    i = i + 1

                    ' Start generation
                    Call GenerateIssueSectionInternals(rst, i)

                End If

                ' Insert processed issue
                dbs.Execute ("INSERT INTO [Processed Issues] ([processed issue]) VALUES (" & rst![ID] & ")")

            End If

        Loop Until (rst.EOF Or rst.BOF)

        Set rst = Nothing

        ' 2) issues that are contained in an already discussed one
        Do
            strSQL = "SELECT * FROM [Issues] i WHERE [to be discussed before start of]=" & phaseId & " AND EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id AND cr.type=2 AND EXISTS (SELECT * FROM [Processed Issues] WHERE [processed issue]=cr.issue2)) AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed issue]=i.id) AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] cr2 WHERE issue2=i.id AND EXISTS(SELECT * FROM [Processed Issues] WHERE [processed issue]=cr2.issue1)) AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id AND cr.issue2<>i.id AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed issue]=cr.issue2))"
            Set rst = dbs.OpenRecordset(strSQL)

            If Not (rst.EOF And rst.BOF) Then
                rst.MoveFirst

               If (rst![to be included] = True) Then
```

```
                    i = i + 1

                    ' Start generation
                    Call GenerateIssueSectionInternals(rst, i)

                End If

                ' Insert processed issue
                dbs.Execute ("INSERT INTO [Processed Issues] ([processed issue]) VALUES ("
& rst![ID] & ")")

            End If

        Loop Until (rst.EOF Or rst.BOF)

        Set rst = Nothing

        ' 3) issues that are required by an already discussed one
        Do
            strSQL = "SELECT * FROM [Issues] i WHERE [to be discussed before start of]=" &
phaseId & " AND EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id
AND cr.type=1 AND EXISTS (SELECT * FROM [Processed Issues] WHERE [processed is-
sue]=cr.issue2)) AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed is-
sue]=i.id) AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] cr2 WHERE issue2=i.id
AND EXISTS(SELECT * FROM [Processed Issues] WHERE [processed issue]=cr2.issue1)) AND NOT
EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id AND
cr.issue2<>i.id AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed is-
sue]=cr.issue2))"
            Set rst = dbs.OpenRecordset(strSQL)

            If Not (rst.EOF And rst.BOF) Then
                rst.MoveFirst

                If (rst![to be included] = True) Then

                    i = i + 1

                    ' Start generation
                    Call GenerateIssueSectionInternals(rst, i)

                End If

                ' Insert processed issue
                dbs.Execute ("INSERT INTO [Processed Issues] ([processed issue]) VALUES ("
& rst![ID] & ")")

            End If

        Loop Until (rst.EOF Or rst.BOF)

        Set rst = Nothing

        ' 4) issues that are influenced by an already discussed one.
        Do
            strSQL = "SELECT * FROM [Issues] i WHERE [to be discussed before start of]=" &
phaseId & " AND EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id
AND cr.type=4 AND EXISTS (SELECT * FROM [Processed Issues] WHERE [processed is-
sue]=cr.issue2)) AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed is-
sue]=i.id) AND NOT EXISTS (SELECT * FROM [Conceptual Relationships] cr2 WHERE issue2=i.id
AND EXISTS(SELECT * FROM [Processed Issues] WHERE [processed issue]=cr2.issue1)) AND NOT
EXISTS (SELECT * FROM [Conceptual Relationships] cr WHERE cr.issue1=i.id AND
cr.issue2<>i.id AND NOT EXISTS (SELECT * FROM [Processed Issues] WHERE [processed is-
sue]=cr.issue2))"
            Set rst = dbs.OpenRecordset(strSQL)

            If Not (rst.EOF And rst.BOF) Then
                rst.MoveFirst

                If (rst![to be included] = True) Then
```

```
                    i = i + 1

                    ' Start generation
                    Call GenerateIssueSectionInternals(rst, i)

                End If

                ' Insert processed issue
                dbs.Execute ("INSERT INTO [Processed Issues] ([processed issue]) VALUES ("
& rst![ID] & ")")

            End If

        Loop Until (rst.EOF Or rst.BOF)

        Set rst = Nothing

        ' Check whether there is still something to be included
        strSQL = "SELECT * FROM [issues] i WHERE i.[to be discussed before start of]=" &
phaseId & " AND [to be included] = true AND NOT EXISTS (SELECT * FROM [Processed Issues]
WHERE i.id = [processed issue])"
        Set rst = dbs.OpenRecordset(strSQL)

        ' Timeout
        If (help > 99) Then
                objWord.ActiveDocument.Content.InsertParagraphAfter
                objWord.ActiveDocument.Content.InsertAfter ("GENERATION CANCELLED DUE TO
UNRESOLVABLE INCONSISTENCY IN ISSUE MODEL")
                objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
        End If

    Loop Until (rst.EOF Or rst.BOF Or help > 99)

End Sub


' Generate Issue Section Internals
Sub-GenerateIssueSectionInternals(rst As DAO.Recordset, i As Integer)

    ' Define variables
    Dim dbs As Database
    Dim subRst As DAO.Recordset
    Dim subSQL As String
    Dim name As String
    Dim description As String
    Dim stakeholders As String
    Dim attributes As String
    Dim ID As Integer
    Dim influencingIssues As String
    Dim notation As String
    Dim notationScope As Integer
    Dim notationSource As String
    Dim degreeOf As Integer
    Dim toBeDocumented As Boolean
    Dim forEach As Boolean
    Dim costs As String
    Dim issueType As Integer
    Dim time As String
    Dim opt As Boolean
    Dim conditions As String
    Dim alreadyDescribed As Boolean


    On Error Resume Next

    ' Read and set variables
    Set dbs = CurrentDb
    name = rst![issue name]
    description = rst![description]
    stakeholders = rst![stakeholders]
    attributes = rst![attributes]
```

```
    issueType = rst![type]
    ID = rst![ID]
    notation = rst![notation]
    notationScope = rst![notation scope]
    degreeOf = rst![degree of freedom]
    toBeDocumented = rst![to be documented]
    influencingIssues = ""
    costs = rst![costs for specific requirements]
    forEach = False
    notationSource = rst![notation source]
    time = rst![required time]
    opt = rst![optional]
    alreadyDescribed = False

    ' Create header
    ' objWord.ActiveDocument.Sections.Add
    objWord.ActiveDocument.Content.InsertParagraphAfter
    If (issueType <> 3) Then objWord.ActiveDocument.Content.InsertAfter (i & ". Elicitation
of " & name & "s")
    If (issueType = 3) Then objWord.ActiveDocument.Content.InsertAfter (i & ". Elicitation
of " & name)
    objWord.ActiveDocument.Paragraphs.Last.Style = objWord.ActiveDocument.Styles("Heading
2")
    objWord.ActiveDocument.Paragraphs.Last.Format.SpaceBefore = 40
    objWord.ActiveDocument.Paragraphs.Last.Format.SpaceAfter = 10

    ' Insert description of issue
    description = "A " & name & " is " & LCase(description)
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Definition: " & description)
    Call Format("Definition: " & description, 0, False, False, True)
    Call Format("Definition:", 0, False, True, False)

    ' Insert involvement hints
    subSQL = "SELECT * FROM [Conceptual Relationships] cr, [Issues] i WHERE cr.[issue1] = "
& ID & " AND cr.[type] = 3 AND cr.[issue2]=i.[id] and i.[type]=i2"
    Set subRst = dbs.OpenRecordset(subSQL)

    ' No singleton
    If (issueType <> 3) Then
        If ((Not (subRst.EOF And subRst.BOF) And toBeDocumented) Or (subRst.EOF And
subRst.BOF)) Then
            Call InvolvementHint1(name, stakeholders, time)
        End If
    End If

    ' singleton
    If (issueType = 3) Then
        Call InvolvementHint2(name, stakeholders, time)
    End If

    ' Issue has common requirements
    If (degreeOf = 3 Or degreeOf = 4 Or degreeOf = 6) Then
        Call CommonalityHint(name)
    End If

    Set subRst = Nothing

    ' Insert influence hint when influencing issues exist and issue is NOT abstract
    If (issueType <> 2) Then
        subSQL = "SELECT i.[issue name] FROM [Issues] i, [Conceptual Relationships] cr
WHERE cr.[issue1] = " & rst![ID] & " AND cr.[issue2] = i.[id] AND cr.[type]=4"
        Set subRst = dbs.OpenRecordset(subSQL)
        If Not (subRst.EOF And subRst.BOF) Then
            subRst.MoveFirst
            Do
                influencingIssues = influencingIssues & " " & subRst![issue name] & "s,"
                subRst.MoveNext
            Loop Until subRst.EOF
            Call InfluenceHint1(name, influencingIssues)
        End If
```

```
        Set subRst = Nothing
    End If

    ' Identifying requirements via consideration of related issues
    subSQL = "SELECT i.[id], i.[issue name], i.[type], cr.[relationship name],
cr.[multiplicity2] FROM [Issues] i, [Conceptual Relationships] cr, [Processed Issues] pr
WHERE cr.[issue1] = " & ID & " AND pr.[processed issue]=i.[id] AND cr.[issue2] = i.[id] AND
cr.[issue1]<>cr.[issue2] AND (cr.[type]=1 OR cr.[type]=2) ORDER BY pr.[id]"
    Set subRst = dbs.OpenRecordset(subSQL)

    ' If requiring or containing issues exist
    If Not (subRst.EOF And subRst.BOF) Then
        subRst.MoveFirst
        Do

            ' If related issue is no singleton
            If (subRst![type] <> 3) Then
                objWord.ActiveDocument.Content.InsertParagraphAfter
                objWord.ActiveDocument.Content.InsertAfter ("For each " & subRst![issue
name] & " identified before:")
                objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
                forEach = True
            End If

            ' If issue class of interest is no singleton
            If (issueType <> 3) Then
                Call IdentifyingInstruction1(name, subRst![relationship name],
subRst![issue name], subRst![multiplicity2], subRst![type], forEach)
                Call CollectingInstruction1(name, subRst![issue name], subRst![type], fo-
rEach)
            Else
                Call DescribingInstruction2(name, subRst![relationship name], subRst![issue
name], attributes, forEach)
                alreadyDescribed = True
            End If

            forEach = False

            subRst.MoveNext
        Loop Until subRst.EOF
    End If

    Set subRst = Nothing

    ' Covering additional requirements (only applicable when: not singleton and not a spe-
cialization and multiplicity2="*")
    subSQL = "SELECT * FROM [Conceptual Relationships] WHERE issue1 =" & ID & " AND type<>3
AND type<>4 AND multiplicity2='*'"
    Set subRst = dbs.OpenRecordset(subSQL)

    If (issueType <> 3 And Not (subRst.EOF Or subRst.BOF)) Then
        Call IdentifyingInstruction2(name, True)
        Call CollectingInstruction2(name)
    End If

    ' Covering additional requirements (also applicable when issue has no relationship but
is not singleton")
    subSQL = "SELECT * FROM [Issues] WHERE [id]= " & ID & " AND type<>3 AND NOT EXISTS
(SELECT * FROM [Conceptual Relationships] WHERE issue1=" & ID & " and type<>4)"
    Set subRst = dbs.OpenRecordset(subSQL)

    If (Not (subRst.EOF Or subRst.BOF)) Then
        Call IdentifyingInstruction2(name, False)
        Call CollectingInstruction2(name)
    End If


    Set subRst = Nothing

    ' Decomposion
```

```
    subSQL = "SELECT [issue1], [relationship name] FROM [Conceptual Relationships] WHERE
[issue1] = " & ID & " AND [issue2] = " & ID & " AND [type]=2"
    Set subRst = dbs.OpenRecordset(subSQL)

    If Not (subRst.EOF And subRst.BOF) Then
        subRst.MoveFirst
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("For each " & name & " identified so
far:")
        DecomposingInstruction (name)
    End If

    Set subRst = Nothing

    ' Self Require
    subSQL = "SELECT [issue1], [relationship name], [multiplicity2] FROM [Conceptual Rela-
tionships] WHERE [issue1] = " & ID & " AND [issue2] = " & ID & " AND [type]=1"
    Set subRst = dbs.OpenRecordset(subSQL)

    If Not (subRst.EOF And subRst.BOF) Then
        subRst.MoveFirst
        objWord.ActiveDocument.Content.InsertParagraphAfter
        objWord.ActiveDocument.Content.InsertAfter ("For each " & name & " identified so
far:")
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
        forEach = True
        Call IdentifyingInstruction1(name, subRst![relationship name], name,
subRst![multiplicity2], issueType, forEach)
        Call CollectingInstruction1(name, name, issueType, forEach)
        forEach = False
    End If

    Set subRst = Nothing

    ' Notation instruction for visualizing interplay
    If (notationScope = 2 And notation <> "") Then
        Call VisualizingInstruction2(name, notation, notationSource)
    End If


    ' Assumption Hints for Soft Assumptions
    Dim j As Integer
    Dim normalConstraint As Boolean
    normalConstraint = False

    subSQL = "SELECT * FROM [Assumptions] WHERE [affected issue] = '" & name & "' AND [hard
assumption]=false"
    Set subRst = dbs.OpenRecordset(subSQL)
    If Not (subRst.EOF And subRst.BOF) Then

        j = 0
        normalConstraint = True

        Call AssumptionHint1(name, j, subRst)

    End If
    Set subRst = Nothing


    ' Assumption Hints for Hard Assumptions
    subSQL = "SELECT * FROM [Assumptions] WHERE [affected issue] = '" & name & "' AND [hard
assumption]=true"
    Set subRst = dbs.OpenRecordset(subSQL)

    If Not (subRst.EOF And subRst.BOF) Then

        j = 0

        Call AssumptionHint2(name, normalConstraint, j, subRst)

    End If
```

```
    Set subRst = Nothing


    ' Selection Hint
    If (degreeOf = 1 Or degreeOf = 3 Or degreeOf = 5 Or degreeOf = 6) Then
        Call SelectionHint(name)
    End If



    ' Insert influence hint when influencing issues exist and issue IS abstract
    If (issueType = 2) Then
        subSQL = "SELECT i.[issue name] FROM [Issues] i, [Conceptual Relationships] cr
WHERE cr.[issue1] = " & ID & " AND cr.[issue2] = i.[id] AND cr.[type]=4"
        Set subRst = dbs.OpenRecordset(subSQL)
        If Not (subRst.EOF And subRst.BOF) Then
            subRst.MoveFirst
            Do
                influencingIssues = influencingIssues & " " & subRst![issue name] & "s,"
                subRst.MoveNext
            Loop Until subRst.EOF
            Call InfluenceHint2(name, influencingIssues)
        End If
        Set subRst = Nothing
    End If

    ' Documentation Hint without condition
    If (toBeDocumented = False) Then
        Call DocumentationHint1(name)
    End If

    ' Documentation Hint with condition
    If (toBeDocumented = True And opt = True) Then

        subSQL = "SELECT DISTINCT [condition] " & _
                 "FROM Issues, (SELECT DA.[activity name], DA.[condition], DA.[optional],
[affected issue] FROM (SELECT * FROM [Development Activities] AS DA, Decisions AS D WHERE
DA.ID=D.[made during])  AS temp INNER JOIN [Information Needs] AS I1 ON I1.[ID] =
temp.[Information needs].Value) " & _
                 "WHERE [issue name]=I1.[affected issue] and [Issues].[optional]=true AND
[Issues].[ID]=" & ID

        Set subRst = dbs.OpenRecordset(subSQL)
        conditions = ""

        If Not (subRst.EOF And subRst.BOF) Then
            subRst.MoveFirst

            Do
                conditions = conditions & subRst![condition]
                subRst.MoveNext

                If (Not subRst.EOF) Then
                    conditions = conditions & ", or "
                End If

            Loop Until subRst.EOF
            Call DocumentationHint2(name, conditions)
        End If

    End If


    ' Actual Description (if toBeDocumented or abstract)
    If (toBeDocumented Or issueType = 2) Then

        ' If related issue is no singleton
        If (issueType <> 3) Then
            objWord.ActiveDocument.Content.InsertParagraphAfter
            objWord.ActiveDocument.Content.InsertAfter ("For each " & name & " identified
so far:")
```

```
            objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
            forEach = True
        End If

        ' if issue is abstract
        If (issueType = 2) Then

            ' Identify specializing issues and integrate classification hint
            influencingIssues = ""
            subSQL = "SELECT i.[issue name] FROM [Issues] i, [Conceptual Relationships] cr
WHERE cr.[issue2] = " & ID & " AND cr.[issue1] = i.[id] AND cr.[type]=3"
            Set subRst = dbs.OpenRecordset(subSQL)
            If Not (subRst.EOF And subRst.BOF) Then
                subRst.MoveFirst
                Do
                    influencingIssues = influencingIssues & " " & subRst![issue name] & ","
                    subRst.MoveNext
                Loop Until subRst.EOF
                Call ClassifyingInstruction(name, influencingIssues, forEach)
            End If
            Set subRst = Nothing

        ' reuse in every case required
        ElseIf (degreeOf = 1 Or degreeOf = 3) Then
            Call SelectingInstruction1(name, forEach)

        ' reuse (only if possible) required
        ElseIf (degreeOf = 5 Or degreeOf = 6) Then
            Call SelectingInstruction2(name, attributes, forEach)
            Call FlexibilityHint(name, costs, forEach)

        ' else
        ElseIf (alreadyDescribed = False) Then
            Call DescribingInstruction1(name, attributes, forEach)

        End If

        ' Notation instruction for visualizing each requirement
        If (notationScope = 1 And notation <> "") Then
            Call VisualizingInstruction1(name, notation, notationSource, forEach)
        End If

    End If


End Sub




' Instructions & Hints

' Identifying Instruction 1
Sub-IdentifyingInstruction1(issue As String, relationship As String, referencingIssue As
String, cardinality As String, ty As Integer, forEach As Boolean)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    If (ty <> 3) Then
        objWord.ActiveDocument.Content.InsertAfter ("Ask the stakeholders the following
question: Which " & issue & "s are " & relationship & " this " & referencingIssue & "?")
    Else
        objWord.ActiveDocument.Content.InsertAfter ("Ask the stakeholders the following
question: Which " & issue & "s are " & relationship & " the " & referencingIssue & "?")
    End If

    If (forEach) Then
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40
    Else
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
    End If
```

```
End Sub


' Identifying Instruction 2
Sub-IdentifyingInstruction2(issue As String, okay As Boolean)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    If (okay = True) Then
        objWord.ActiveDocument.Content.InsertAfter ("Ask the stakeholders the following
question: Are further " & issue & "s required? If yes, which ones?")
    Else:
        objWord.ActiveDocument.Content.InsertAfter ("Ask the stakeholders the following
question: Which " & issue & "s are required?")
    End If
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Collecting Instruction 1
Sub-CollectingInstruction1(issue As String, referencingIssue As String, ty As Integer,
forEach As Boolean)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Collect the identified " & issue & "s in a
corresponding bullet list (if not yet done)")

    If (ty <> 3) Then
        objWord.ActiveDocument.Content.InsertAfter (" and add a link to the related " &
referencingIssue & ".")
    Else
        objWord.ActiveDocument.Content.InsertAfter (".")
    End If

    If (forEach) Then
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40
    Else
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
    End If

End Sub


' Collecting Instruction 2
Sub-CollectingInstruction2(issue As String)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Collect the identified " & issue & "s in a
corresponding bullet list (if not yet done).")
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Describing Instruction 1
Sub-DescribingInstruction1(issue As String, attributes As String, forEach As Boolean)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Ask the stakeholders the following ques-
tion: Could you please describe the " & issue & " especially with regard to " & attributes
& "?")

    If (forEach) Then
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40
    Else
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
    End If

End Sub
```

```
' Describing Instruction 2
Sub-DescribingInstruction2(issue As String, relationship As String, referencingIssue As
String, attributes As String, forEach As Boolean)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Ask the stakeholders the following ques-
tion: Could you please describe the " & issue & " " & relationship & " this " & refer-
encingIssue & " especially with regard to " & attributes & "?")

    If (forEach) Then
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40
    Else
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
    End If

End Sub


' Classifying Instruction
Sub-ClassifyingInstruction(issue As String, subIssues As String, forEach As Boolean)

    subIssues = Mid(subIssues, 1, Len(subIssues) - 1)
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Discuss with the stakeholders if this " &
issue & " is one of the following specialized issues and categorize it accordingly:" &
subIssues & ".")

    If (forEach) Then
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40
    Else
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
    End If

End Sub


' Visualizing Instruction 1
Sub-VisualizingInstruction1(issue As String, notation As String, source As String, forEach
As Boolean)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Use a " & notation & " to clarify the
details of this " & issue & ". ")
    If (source <> "") Then
        objWord.ActiveDocument.Content.InsertAfter ("Additional information about this
technique can be found in " & source)
    End If

    If (forEach) Then
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40
    Else
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
    End If

End Sub


' Visualizing Instruction 2
Sub-VisualizingInstruction2(issue As String, notation As String, source As String)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Use a " & notation & " to clarify the
interplay between all " & issue & "s. ")
    If (source <> "") Then
        objWord.ActiveDocument.Content.InsertAfter ("Additional information about this
technique can be found in " & source)
    End If

    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
```

```
End Sub


' Docomposing Instruction
Sub-DecomposingInstruction(issue As String)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Decompose the hierarchy of this " & issue
& " until no further decomposition is possible. Collect the identified " & issue & "s in a
corresponding bullet list (if not yet done) and add a link to the parent " & issue & ".")
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40

End Sub


' Selecting Instruction 1
Sub-SelectingInstruction1(issue As String, forEach As Boolean)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Let the stakeholders select the best fit-
ting " & issue & " from the SPL specification and map it accordingly. Reject all elicited "
& issue & "s that cannot be mapped.")

    If (forEach) Then
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40
    Else
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
    End If

End Sub


' Selecting Instruction 2
Sub-SelectingInstruction2(issue As String, attributes As String, forEach As Boolean)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Motivate the stakeholders to select the
best fitting " & issue & " from the SPL specification and map it accordingly. If the re-
quired " & issue & " is not covered sufficiently in the SPL specification yet, describe
this " & issue & " especially with regard to " & attributes & " from scratch.")

    If (forEach) Then
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 40
    Else
        objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0
    End If

End Sub


' Involvment Hint 1
Sub-InvolvementHint1(issue As String, stakeholders As String, time As String)

    If (stakeholders = "") Then stakeholders = "stakeholder"
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Invite and involve a (group of) " & stake-
holders & "s to an elicitation session in order to discuss requirements concerning " &
issue & "s. ")
    If (time <> "") Then
        objWord.ActiveDocument.Content.InsertAfter ("The required time for this step is
expected to take around " & time & ".")
    End If
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Involvment Hint 2
Sub-InvolvementHint2(issue As String, stakeholders As String, time As String)

    If (stakeholders = "") Then stakeholders = "stakeholder"
```

```
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Invite and involve a (group of) " & stake-
holders & "s to an elicitation session in order to discuss requirements concerning the " &
issue & ". ")
    If (time <> "") Then
        objWord.ActiveDocument.Content.InsertAfter ("The required time for this step is
expected to take around " & time & ".")
    End If
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Influence Hint 1
Sub-InfluenceHint1(issue As String, influencingIssues As String)

    influencingIssues = Mid(influencingIssues, 1, Len(influencingIssues) - 1)
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Important hint: Consider especially the
already elicited requirements concerning the following issues when determining the " &
issue & "s:" & influencingIssues & ".")
    Call Format("Important hint:", 0, True, True, True)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Influence Hint 2
Sub-InfluenceHint2(issue As String, influencingIssues As String)

    influencingIssues = Mid(influencingIssues, 1, Len(influencingIssues) - 1)
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Important hint: Consider especially the
already elicited requirements concerning the following issues when classifying the " &
issue & "s:" & influencingIssues & ".")
    Call Format("Important hint:", 0, True, True, True)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Commonality Hint
Sub-CommonalityHint(issue As String)

    Dim text As String
    text = "Be aware that a set of " & issue & "s is already implemented by default and
need not to be elicited again. Consider the list of these " & issue & "s in the SPL speci-
fication and inform the stakeholders about, so that you can break discussions immediately
as soon as they start asking for the collection of these common requirements. Additional
requirements are of course allowed."
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Important hint: " & text)
    Call Format("Important hint:", 0, True, True, True)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Assumption Hint 1
Sub-AssumptionHint1(issue As String, j As Integer, subRst As DAO.Recordset)

    Dim text As String
    text = "Important hint: Be aware that there are constraints defined for " & issue & "s.
Hence, the " & issue & "s, stakeholders may ask for, are restricted as follows."

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter (text)
    Call Format("Important hint:", 0, True, True, True)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

    subRst.MoveFirst
```

```
    Do
        j = j + 1
        Call AssumptionText(subRst![description], subRst![rationale], j)
        subRst.MoveNext
    Loop Until subRst.EOF

    text = "If the stakeholders require something that contravenes these constraints, in-
form them about possible (high) extract costs and that an expert check must be done before
you can accept this requirement."

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter (text)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Assumption Hint 2
Sub-AssumptionHint2(issue As String, normalConstraint As Boolean, j As Integer, subRst As
DAO.Recordset)

    Dim text As String

    If (normalConstraint) Then
        text = "Important hint: Be aware that in addition, there are also hard constraints
defined for " & issue & "s! Hence, the " & issue & "s, stakeholders may ask for, are fur-
thermore restricted as follows."
    Else
        text = "Important hint: Be aware that there are constraints defined for " & issue &
"s that are hard! Hence, the " & issue & "s, stakeholders may ask for, are restricted as
follows."
    End If

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter (text)
    Call Format("Important hint:", 0, True, True, True)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

    subRst.MoveFirst
    Do
        j = j + 1
        Call AssumptionText(subRst![description], subRst![rationale], j)
        subRst.MoveNext
    Loop Until subRst.EOF

    text = "If the stakeholders require something that contravenes these constraints, in-
form them that this is technically not possible."

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter (text)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub

' Assumption itself
Sub-AssumptionText(constraint As String, rationale As String, i As Integer)

    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter (i & ". " & constraint & ". ")
    If (rationale <> "") Then
        objWord.ActiveDocument.Content.InsertAfter ("The reason for this constraint is: " &
rationale & ".")
    End If

    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 20

End Sub

' Selection Hint
Sub-SelectionHint(issue As String)
```

```
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter ("Consider the set of predefined " & issue &
"s in the SPL specification.")
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Flexibility Hint
Sub-FlexibilityHint(issue As String, costs As String, forEach As Boolean)

    Dim text As String
    text = " In in this case, please inform the stakeholders about " & costs & " extra
costs also (even if the given constraints are hold)."
    objWord.ActiveDocument.Content.InsertAfter (text)

End Sub


' Documentation Hint 1
Sub-DocumentationHint1(issue As String)

    Dim text As String
    text = "Important hint: It is not necessary to elicit or describe details about " &
issue & "s. A pure enumeration or collection (e.g., in a bullet list) is sufficient."
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter (text)
    Call Format("Important hint:", 0, True, True, True)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub


' Documentation Hint 2
Sub-DocumentationHint2(issue As String, conditions As String)

    Dim text As String

    If (conditions = "") Then
        conditions = "(no condition mentioned)"
    End If

    text = "Important hint: It is only necessary to elicit and describe details about " &
issue & "s, if the customer " & conditions & ". Otherwise, a pure enumeration or collection
(e.g., in a bullet list) is sufficient."
    objWord.ActiveDocument.Content.InsertParagraphAfter
    objWord.ActiveDocument.Content.InsertAfter (text)
    Call Format("Important hint:", 0, True, True, True)
    objWord.ActiveDocument.Paragraphs.Last.Format.LeftIndent = 0

End Sub



' Formating:
Sub-Format(text As String, size As Integer, bold As Boolean, underline As Boolean, italic
As Boolean)

    ' Format header
    Set myRange = objWord.ActiveDocument.Content
    myRange.Find.Execute FindText:=text, Forward:=False
    If myRange.Find.Found = True Then

        If (bold) Then myRange.bold = True
        If (underline) Then myRange.Font.underline = True
        If (italic) Then myRange.Font.italic = True
        If size <> 0 Then myRange.Font.size = size

    End If

End Sub
```

Appendix D: Issue Section Generation Algorithm (Pseudo Code)

```
IssueSection(issueName)
{
        // Insert Involvement Hints
        InvolvingInstruction(issueName, stakeholderGroup);

        // Insert Commonality Hint
        If (issue has common requirements concerned with it)
                CommonalityHint(issueName);

        // Insert Influence Hint for non-abstract Issues
        If (issue is not abstract and issue is influenced by another issue)
                InfluenceHint1(issueName, influencingIssues);

        // Process the related Issues
        If (issue is required or contained by / in another issue)
        {
                If (requiring or containing issues is no singleton)
                        open (forEachInstructionBlock(requiring or containing issue));
                if (issue is no singleton)
                {
                        IdentifyingInstruction1(issueName, relatedIssue, relationship);
                        CollectingInstruction1(issueName, relatedIssue);
                }
                Else
                        DescribingInstruction2(issueName, relatedIssue, relationship);

                If (requiring or containing issues is no singleton)
                        close (forEachInstructionBlock(requiring or containing issue));
        }

        If (issue can have further instances beyond those related to other issues)
        {
                IdentifiyingInstruction2(issueName);
                CollectingInstruction2(issueName);
        }

        If (issue has a self-contain relationship)
                DecomposingInstruction(issueName);

        If (issue has a self-require relationship)
        {
                open (forEachInstructionBlock(issue));
                IdentifyingInstruction1(issueName, issueName, relationship);
                CollectingInstruction1(issueName, issueName);
                close (forEachInstructionBlock(issueName));
        }

        // Visualizing the Interplay of Issues
        If (issue has notation and notationScope is all)
                VisualizationInstruction2(issueName, notation);

        // Insert Assumption Hint
        If (soft assumptions are defined on issue)
                AssumptionHint1(issueName, assumptions);
        If (hard assumptions are defined on issue)
                AssumptionHint2(issueName, assumptions);

        // Insert Selection Hint
        If (issue has predefined requirements concerned with it)
                SelectionHint(issueName);

        // Insert Influence Hints for abstract Issues
```

```
        If (issue is abstract and issue is influenced by another issue)
                InfluenceHint2(issueName, influencingIssues);

        // Insert Documentation Hint
        If (issue is not to be documented)
                DocumentationHint(issueName);

        // Describe the details of the elicited Issues
        If (issue is to be documented or issue is abstract)
        {
                If (issue is no singleton)
                        open forEachInstructionBlock(issueName);
                if (issue is abstract)
                        ClassifyingInstruction(issueName, specializingIssues);
                else if (predefined requirements concerning issue must be reused)
                        SelectingInstruction1(issueName);
                else if (predefined requirements concerning issue should be reuse wherever
                possible)
                {
                        SelectingInstruction2(issueName, attributes);
                        FlexibilityHint(issueName, costs);
                }
                Else
                        DescribingInstruction1(issueName, attributes);
                  If (issue is no singleton)
                        close forEachInstructionBlock(issueName);

                // Visualize issue details
                If (issue has notation and notationScope is each)
                        VisualizationInstruction1(issueName, notation);
        }
}
```

Appendix E: Experiment Material

Fachbereich Informatik: Software Engineering: Processes and Measurement Research Group
Dozent: Prof. Dr. Dr. h.c. Dieter Rombach, Dr. Jörg Dörr
Wintersemester 2011/12, Vorlesung 89-3155

Non-Disclosure Agreement & Informed Consent

Last Name (Student):

First Name:

Address:

Date of Birth:

Enrollment Number:

The student is taking part in the lecture 'Requirements Engineering' of AG SE in cooperation with the Fraunhofer Institute for Experimental Software Engineering during the winter semester 2011/2012. During the lecture the student will participate in an experiment at Fraunhofer IESE.

To ensure the success of the experiment the student will keep confidential all information gained during the experiment and shall not disclose any information about aim, procedure and materials especially to the other students joining the lecture.

The student also accepts that the experiment is recorded for the purpose of anonymous analysis. All gather data will be kept in confidence. The performance or the results of the experiment will not influence the grading of the student in the 'Requirements Engineering' lecture.

Kaiserslautern, ……..

…….
(Student)

Background Questionnaire

Participant Code: _____

Group:	O M	O C	(filled out by advisor!)
Age:	_____	O male	O female
What is subject of your study?	_____		
In which semester?	_____	O bachelor	O master O diploma
What did you studied before?	_____		

Is the RE lecture the only source for your requirements engineering knowledge?

O yes O no

If no: Where did you make further experience with requirements engineering?

_____

How many interviews have you already conducted to elicit requirements?

_____ interviews for requirements elicitation

How many interview have you already conducted in other contexts?

_____ interviews for other purposes

If you have conducted interview before: Have you used checklists / instructions for these interviews?

O yes O partially O no

For how many years have you already worked in the software industry (also as student)?

_____ years

Do you know the basics of software product lines?

O yes O no

How do you rate your English competence in understanding text and speak?

O high O rather high O medium O rather low O low

Thank you!

Study Instruction

First of all, I would like to **thank you** for participating in this study.

The overall purpose of this study is to investigate the usefulness of the requirements elicitation instruction (see material A) handed out to you. For this purpose, you are asked to perform the tasks described below.

In the main task of the study (task 3), you have to play the role of a requirements engineer that interviews a stakeholder in order to elicit requirements by using this instruction.

Basically, an elicitation instruction is a document that provides you with concrete steps you have to do and questions you have to ask during an elicitation session. Furthermore, additional information and constraints you have to consider are also included in it.

Task 1: Read the elicitation instruction (material A) **carefully and completely in order to understand all elicitation steps** as well as the overall structure of the instruction. This is a prerequisite that you are able to use this instruction appropriately in your interview later. After that, take also a look at the SPL specification (material B) and browse through it in order to know where you can quickly find certain information, if required. However, it is neither needed to read nor to memorize this document completely. Thus, we assume that you can do this entire task in approximately 15 minutes.

Task 2: Fill out the material questionnaire (material C). We assume that you can do this task in approximately 10 minutes.

Well, now you are perfectly prepared for the interview.

Task 3: Go to the study adviser and inform him that you are ready to start the interview. Take all the material as well as a pen and blank sheets with you. Interview the stakeholder by using the elicitation instruction (material A). You should **follow the instruction wherever possible**, but you are free or even requested to improvise. In particular, the stakeholder will confront you with situations you will not find as-is in the instruction. Be prepared and **use the information given in the material** for negotiating; however, please avoid reading the entire material during interview. For each requirement mentioned, **inform the stakeholder whether you can accept this requirement or whether you have to reject it, respectively whether you have to ask an expert first**. We assume that you can do this task in approximately 30 minutes.

Task 4: Fill out the post questionnaire (material D) and give all the material you received (A, B, C, D) back to the advisor. We assume that you can do this task in approximately 10 minutes.

Again, we thank you very much for your participation. **Please remember that you are not allowed to talk with other people about this study**, especially not with the other study participants!

D

POST QUESTIONNAIRE Participant Code: _____

With this questionnaire, we would like to investigate how helpful the **elicitation instruction** has been for supporting you in the interview.

Your general opinion:

What do you like in the elicitation instruction?

What is the most valuable information provided to you by the elicitation instruction?

What should be improved in the elicitation instruction?

What did you miss in the elicitation instruction?

How do you agree or disagree with regard to the following statements:

	totally disagree (1)	rather disagree (2)	neither agree nor disagree (3)	rather agree (4)	totally agree (5)
The elicitation instruction was helpful for performing the interview					
The elicitation instruction enabled me to achieve good results / to elicit quite a good set of requirements					
I would not have achieved the same results / requirements without using the elicitation instruction					
I would have required more time for achieving the same results / requirements when not using the elicitation instruction					
The elicitation instruction allowed me to deviate from it, if necessary					
The elicitation instruction could always be used as a concrete process to follow					
The elicitation instruction provided me clear how-to guidance (i.e., it clearly mentions a sequence of steps to be carried out during an elicitation process)					
The elicitation instruction was easy to read					
The elicitation instructions was easy to handle					
The elicitation instruction reflected the style in which I would also perform interviews without an elicitation instruction					

The elicitation instruction provided me with sound knowledge about the given constraints and capabilities I have to consider					
The elicitation instruction made clear in which order certain elicitation steps should be performed best					
The elicitation instruction provided me with good indications to know when the elicitation is finished					
The elicitation instruction informed me about the information (e.g., business processes, data, use cases, interfaces to external systems, etc.) which are relevant to be discussed with the stakeholders					
The elicitation instruction enabled me to get a good understanding of the software product line on which the new system should be built					
The elicitation instructions supported me in finding important information quickly					
The elicitation instruction informs me whether requirements of a certain type do already exist and could be reused instead of elicited from scratch (e.g., reusable use case variants)					
I could answer all stakeholder's questions by only using the information given in the elicitation instruction					

Thank you!

C

Material Questionnaire

Participant Code: _____

The purpose of this questionnaire is to check whether the given material provides you with certain information or not.

Please try to answer the questions based on the information given in the elicitation instruction wherever possible. Only if you cannot find the required information in it, take a look into the SPL specification too. If you can also not find the required information there, you are free to answer the questions based on what you have learned in the RE lecture or elsewhere.

To each question, you may only make **ONE** selection.

Information Needs	1. Is it necessary to elicit the (business) activities that are to be supported? ☐ yes ☐ yes, but without details ☐ no ☐ I have no idea 2. Is it necessary to elicit system use cases? ☐ yes ☐ yes, but without details ☐ no ☐ I have no idea 3. Which detailed information about the user groups are you asked to elicit? ☐ interest ☐ education ☐ responsibilities ☐ I have no idea and would have to ask an expert 4. Which system functions would you have to elicit and specify in detail, if they were required by the stakeholders? ☐ all required system functions have to be elicited and specified ☐ data mapping, XML export ☐ PDF export, email notification ☐ no required system function needs to be elicited and specified ☐ I have no idea and would have to ask an expert

Constraints & Capabilities	1. Is it easy to realize the requirement that the system shall exchange data with a self-developed partner system (ERP)? ☐ yes ☐ no ☐ I have no idea 2. Is it possible to realize the design requirement that the user interface shall be implemented with the Adobe Flex/Air technology? ☐ yes ☐ no ☐ I have no idea 3. What would be your answer to a stakeholder who requests a data exchange with MS Office? ☐ Fine, let's define this requirement in more detail (e.g., office version) ☐ That's possible but will be very expensive to realize ☐ Such an export mechanism already exists ☐ I have no idea and would have to ask an expert 4. Which constraints are defined on business rules to be implemented in the system? ☐ Only static "if then" rules work, but no calculations ☐ Only rules with fix values are possible but no rules that read values from a database ☐ All business rules can be implemented (i.e., there are no constraints) ☐ I have no idea and would have to ask an expert
Rationales	5. Why is it not acceptable that the business processes to be executed shall be modeled in UML? ☐ An additional transformation engine would have to be developed ☐ Two main components of the system would have to be replaced ☐ UML is not sufficient to express all important process details ☐ I have no idea and would have to ask an expert 6. Which extra costs are typically needed for supporting a specific, but commercial ERP system? ☐ low extra costs ☐ medium extra costs ☐ high extra costs ☐ I have no idea and would have to ask an expert
	Thank you!

A

AERE Elicitation Instruction

This document includes precise process instruction on how to elicit, negotiate and specify requirements concerning a system derived from the ARÜS BPM+ plus Product Line. For this purpose, the document provides a sequence of elicitation steps to be carried out in the described order.

The overall purpose of system derived from ARÜS BPM+ is to support the definition, development, execution and monitoring of business processes. While typical customers are enterprises, typical users include process participants, administrators, analysts, developers, and controllers.

In order to do the development work, requirements concerning all the issues mentioned below are needed by the engineers for making corresponding design or development decisions. Hence, it is indispensable that these requirements are elicited before the development can start. To make this happen, we strongly recommend performing the steps in the mentioned order because there are dependencies between the addressed issues that may impact the elicitation.

1. Elicitation of Partner Systems & Interfaces

Definition: A Partner System is an external system already available in the customer's IT landscape or to be introduced in a parallel project.

Ask the stakeholders the following question: Which Partner Systems are to be connected with the system?

Collect the identified Partner Systems in a corresponding bullet list.

Important hint: Be aware that there are constraints defined for Partner Systems. Hence, the Partner Systems, stakeholders may ask for, are restricted as follows.

1. Only commercial ERP systems and no proprietary, self-developed ERP systems can be connected with the system. The reason for this constraint is: Due to liability reasons, data are only allowed to be exchanged with certified systems.

If the stakeholders require something that contravenes these constraints, inform them about possible (high) extract costs and that an expert check must be done before you can accept this requirement.

For each Partner System identified so far:

Motivate the stakeholders to select a best fitting Partner System from the following list and map it accordingly.

- SAP
- MS Office

If the required Partner System is not covered sufficiently in this list yet, describe this Partner System especially with regard to name, purpose, system class (e.g., ERP, office, etc.),

interface description from scratch. Inform the stakeholders about medium extra costs (even if the given constraints are hold) in this case.

2. Elicitation of Technical Infrastructure Components

Definition: A Technical Infrastructure is the external information technology (server operation system, database systems, user management system, etc.) whose services are used by the system under development to run.

Ask the stakeholders the following question: Which Technical Infrastructure Components (operation system, database systems, user management system) are given at the customer side?

Collect the identified Technical Infrastructure Components in a corresponding bullet list.

Important hint: Be aware that there are constraints defined for Technical Infrastructure Components that are hard! Hence, the Technical Infrastructure Components, stakeholders may ask for, are restricted as follows.

1. Only MS Windows XP or higher or a Linux distribution is supported as the operation system in the backend. The reason for this constraint is: The application server used within the system is only available for these operation systems.

2. Only rational database systems that use the SQL language in the 2008 version or a newer version as query language are supported. The reason for this constraint is: The system makes use of the build-in mechanisms of SQL and therefore requires a corresponding support through the database system.

If the stakeholders require something that contravenes these constraints, inform them that this is technically not possible.

For each Technical Infrastructure Component identified so far:

Motivate the stakeholders to select a best fitting Technical Infrastructure Component from the following list and map it accordingly.

Databases
- Oracle 10g
- IBM DB2
- MS SQL

User Management
- Active Directory

If the required Technical Infrastructure Component is not covered sufficiently in this list yet, describe this Technical Infrastructure Component especially with regard to name, description, component type (e.g., database, operation system, etc.) from scratch. Inform the stakeholders about medium extra costs (even if the given constraints are hold) in this case.

Elicitation of Business Processes

Definition: A Business Process is a specific ordering of business activities across time, people, and place, with a beginning, an end, and clearly identified inputs and outputs in order to react on a business interaction.

Ask the stakeholders the following question: Which Business Processes are to be implemented?

Collect the identified Business Process in a corresponding bullet list.

Important hint: Be aware that there are constraints defined for Business Processes. Hence, the Business Processes, stakeholders may ask for, are restricted as follows.

1. Only business processes that starts with ONE clear start event can be supported, i.e., processes that may start with alternative events are not executable. The reason for this constraint is: The process algorithm implemented in the common process engine needs a clear starting point.
2. Only business processes that are expressible with the BPMN 2.0 notation can be supported. The reason for this constraint is: Both, the process designer component and the process engine are only able to deal with BPMN. Replacing or reimplementing these core components would be much too expensive.

If the stakeholders require something that contravenes these constraints, inform them about possible high extract costs and that an expert check must be done before you can accept this requirement.

For each Business Process identified so far:

> Ask the stakeholders the following question: Could you please describe the Business Process especially with regard to name, purpose, precondition, trigger, importance, frequency, quality constraints.

4. Elicitation of Business Activities

Definition: A Business Activity is a step of a business process that results in a valuable, stable state.

For each Business Process identified before:

> Ask the stakeholders the following question: Which Business Activities are part of this Business Process?

Collect the identified Business Activities in a corresponding bullet list.

Important hint: It is not necessary to elicit or describe details about Business Activities. A pure enumeration or collection (e.g., in a bullet list) is sufficient.

Discuss with the stakeholders if this Business Activity is one of the following specialized issues and categorize it accordingly: Human System Activity, i.e., a business activity that is performed by a user with the system.

5. Elicitation of Business Data

Definition: A Business Data is a business object about which structured information is (to be) managed.

For each Business Activity identified before:

Ask the stakeholders the following question: Which Business Data are used as output and input in this Business Activity?

Collect the identified Business Data in a corresponding bullet list.

For each Business Data identified so far:

Ask the stakeholders the following question: Could you please describe the Business Data especially with regard to name, attributes, data types?

6. Elicitation of Business Rules

Definition: A Business Rule is a rule that guides the behavior of an organization in order to operationalize the business strategy.

For each Business Process identified before:

Ask the stakeholders the following question: Which Business Rules are to be considered in this Business Process?

Collect the identified Business Rules in a corresponding bullet list.

Collect the identified Business Rules in a corresponding bullet list (if not yet done).

Important hint: Be aware that there are constraints defined for Business Rules. Hence, the Business Rules, stakeholders may ask for, are restricted as follows.

1. Only rules that set static values and no calculations are supported (e.g., If A=x and B=y then C=z works, but if A=x then C=x*y does not work). The reason for this constraint is: The functionality of the underlying rule engine is still limited. However, extending the rule engine is much too expensive.

If the stakeholders require something that contravenes these constraints, inform them about possible (high) extract costs and that an expert check must be done before you can accept this requirement.

For each Business Rule identified so far:

> Ask the stakeholders the following question: Could you please describe the Business Rule especially with regard to name, description, type?

7. Elicitation of System Use Cases

Definition: A System Use Case is an interaction sequence between a user and a system (including possible alternative or exceptional flows) in order to perform a human system activity.

Important hint: Be aware that a set of System Use Cases is already implemented by default and need not to be elicited again. Consider the list of these System Use Cases below and inform the stakeholders about, so that you can break discussions immediately as soon as they start asking for the collection of these common requirements. Additional requirements are of course allowed.

- Model Business Process. Enables the creation of a business process model using BPMN.
- Implement Business Process. Enables the implementation of an executable business process.
- Deploy Business Process. Enables the execution of an implemented business process.
- Control Business Process. Enables the measurement of an executed business process' performance.

For each Human System Activity identified before:

> Ask the stakeholders the following question: Could you please describe the corresponding System Use Case especially with regard to name, purpose, precondition, description, exceptions, frequency, quality constraints, post condition?

Ask the stakeholders the following question: Are further System Use Cases required? If yes, which ones?

Collect the identified System Use Cases in a corresponding bullet list.

For each System Function identified so far:

> Ask the stakeholders the following question: Could you please describe the corresponding System Use Case especially with regard to name, purpose, precondition, description, exceptions, frequency, quality constraints, post condition?

8. Elicitation of System Functions

Definition: A System Function is an atomic reaction (i.e., state change or response) of the system under development that is triggered by an external stimulus, e.g., an environmental change, or an explicit request of a user or an external system.

Important hint: Be aware that a set of System Functions is already implemented by default and need not to be elicited again. Consider the list of these System Functions below and inform the

stakeholders about, so that you can break discussions immediately as soon as they start asking for the collection of these common requirements. Additional requirements are of course allowed.

- PDF Exporter: allows exporting process data in PDF documents
- Calculator: allows making calculations
- Email Notification: sends emails automatically when a certain event happens
- Reminder Service: sends a reminder that a certain task has to be done

Ask the stakeholders the following question: Which specific System Functions are required?

Collect the identified System Functions in a corresponding bullet list.

For each System Function identified so far:

> Ask the stakeholders the following question: Could you please describe the System Function especially with regard to name, purpose, precondition, description, exceptions, frequency, quality constraints, post condition?

9. Elicitation of User Groups

Definition: A User Group is a group of persons with a common role who will interact with the system under development.

For each System Use Case identified before:

> Ask the stakeholders the following question: Which User Groups are performing this System Use Case?

Collect the identified User Groups in a corresponding bullet list.

For each User Group identified so far:

> Ask the stakeholders the following question: Could you please describe the User Group especially with regard to name, volume, responsibilities, IT experience, preferences, role profile?

10. Elicitation of Workplaces

Definition: A Workplace is a place including at which the system under development is used, i.e., where a user role works with the system.

For each User Group identified before:

> Ask the stakeholders the following question: Which Workplaces are used by this User Group?

Collect the identified Workplaces in a corresponding bullet list.

Important hint: Be aware that there are constraints defined for Workplaces. Hence, the Workplaces, stakeholders may ask for, are restricted as follows.

1. As mobile devices, only Apple is currently supported (i.e., iPhone and iPad). The reason for this constraint is: The rendering engine only supports the Apple technology so far. A reimplementation of this engine is much too expensive.

If the stakeholders require something that contravenes these constraints, inform them about possible high extract costs and that an expert check must be done before you can accept this requirement.

Important hint: It is not necessary to elicit or describe details about Workplaces. A pure enumeration or collection (e.g., in a bullet list) is sufficient.

11. Elicitation of Design Requirements

Definition: A Design Requirement is a constraint for the development of the system under development including security policies, style guides, desired architecture styles, COTS or open source to be used, development activities, and development technology.

Ask the stakeholders the following question: Which Design Requirements are required?

Collect the identified Design Requirements in a corresponding bullet list.

Important hint: Be aware that there are constraints defined for Design Requirements that are hard! Hence, the Design Requirements, stakeholders may ask for, are restricted as follows.

1. For customer-specific extensions of the system functionality, only implementations in Java 1.6. or higher are supported. The reason for this constraint is: The entire system is based on the Java platform and only supports this technology in the backend.
2. Only simple HTML and Java Script are supported as technologies for developing the user interfaces. In particular, specific plug-in technologies for animations or rich internet clients are not supported. The reason for this constraint is: The underlying technology does not support dynamic content.

If the stakeholders require something that contravenes these constraints, inform them that this is technically not possible.

For each Design Requirement identified so far:

> Ask the stakeholders the following question: Could you please describe the Design Requirement especially with regard to name, description?

A

TORE Elicitation Instruction

This document includes process instruction on how to elicit, negotiate and specify requirements concerning a system derived from the ARÜS BPM+ plus Product Line based on the TORE framework. For this purpose, the document provides a sequence of elicitation steps to be carried out in the described order.

The overall purpose of system derived from ARÜS BPM+ is to support the definition, development, execution and monitoring of business processes. While typical customers are enterprises, typical users include process participants, administrators, analysts, developers, and controllers.

In order to do the development work, requirements concerning all the issues mentioned below are needed by the engineers for making corresponding design or development decisions. Hence, it is indispensable that these requirements are elicited before the development can start. To make this happen, we strongly recommend performing the steps in the mentioned order as there are dependencies between the addressed issues that may impact the elicitation.

1. Elicitation of Partner Systems & Interfaces

Definition: A Partner System is an external system already available in the customer's IT landscape or to be introduced in a parallel project.

Ask the stakeholders the following question: Should the system under development be connected with SAP? O yes O no

Ask the stakeholders the following question: Should the system under development be connected with MS Office? O yes O no

Ask the stakeholders the following question: Which further Partner Systems are to be connected?

Describe the Partner Systems with regard to name, purpose, general description, system class (e.g., ERP, office, etc.), interface description.

2. Elicitation of Technical Infrastructure Components

Definition: A Technical Infrastructure is the external information technology (hardware capacity, network capacity, security systems, server operation system, database systems, user management system, etc.) whose services are used by the system under development to run.

Ask the stakeholders the following question: Which of the following operation systems is used at the server side? O MS Windows O Linux

Ask the stakeholders the following question: Which of the following database systems is used in your organization and to be connected? O Oracle O IBM O MS SQL

Ask the stakeholders the following questions: Should the system be connected with Active Directory?

O yes O no

Ask the stakeholders the following question: Do the users want to use the systems via their iPhone / iPad? O yes O no

3. Elicitation of Supported Stakeholders

Definition: A Support Stakeholder is a group of persons with a common role who will benefit from the system under development.

Ask the stakeholders the following question: Which Stakeholders are to be supported by the system under development?

Describe the Support Stakeholder with regard to name, average age, working situation, software experience, typical tasks, preferences, workplace, properties.

4. Elicitation of Stakeholder Goals

Definition: A Stakeholder Goal is a target state in the future that is different from the current state and that is worthwhile to achieve through the system under development.

Ask the stakeholders the following question: Which Stakeholder Goals are to be achieved by the system under development?

Describe the Stakeholder Goals with regard to name, description, achievement metric, achievement date.

5. Elicitation of Stakeholder Tasks

Definition: A Stakeholder Task is a specific ordering of activities with a beginning, an end, and clearly identified inputs and outputs in order to react on a certain event. Stakeholder Tasks can either be business processes or individual tasks.

Ask the stakeholders the following question: Which Stakeholder Tasks are to be supported by the system under development?

Describe the Stakeholder Tasks with regard to name, goal, trigger, priority, execution profile, precondition, input, output, resources.

Elicitation of To-Be Activities

Definition: A To-Be Activity is a step of a task that results in a valuable, stable (intermediate) state towards the result of the entire stakeholder task. In this context, "to-be" expresses that this step will be part of the stakeholder task after the system is developed.

Ask the stakeholders the following question: Which To-Be Activities are steps of the Stakeholder Tasks?

Describe the To-Be Activities with regard to name, description, precondition, post condition, trigger, responsible organizational unit, rules to consider.

7. Elicitation of System Responsibilities

Definition: A System Responsibility is a To-Be Activity that is either supported or even automated by the system under development.

Ask the stakeholders the following question: Which To-Be Activities are to be automated or supported by the system?

Mark the To-Be Activities accordingly.

8. Elicitation of Domain Data

Definition: A Domain Data is an object in the system environment about which structured information is (to be) managed.

Ask the stakeholders the following question: Which Domain Data are relevant in the Stakeholder Tasks?

Describe the Domain Data with regard to name, description, attributes, relationships.

9. Elicitation of System Use Cases (Interactions)

Definition: A System Use Case is an interaction sequence between a user and a system (including possible alternative or exceptional flows) in order to perform a human system activity.

Ask the stakeholders the following question: Could you please describe the System Use Cases for the System Responsibilities with regard to name, purpose, precondition, description, exceptions, frequency, quality constraints, post condition?

10. Elicitation of System Functions

Definition: A System Function is an atomic reaction (i.e., state change or response) of the system under development that is triggered by an external stimulus, e.g., an environmental change, or an explicit request of a user or an external system.

Ask the stakeholders the following question: Which specific System Functions a required?

Describe the System Functions with regard to name, input, output, description, exceptions, rules, quality requirements, precondition, post condition.

11. Elicitation of Design Requirements

Definition: A Design Requirement is a constraint for the development of the system under development including security policies, style guides, desired architecture styles, COTS or open source to be used, development activities, and development technology.

Ask the stakeholders the following question: Which Design Requirements are given for the system under development?

Describe the Design Requirements.

12. Elicitation of Quality Requirements

Definition: A Quality Requirement describes a non-functional property of the system such as efficiency, reliability, usability, security, maintainability and portability.

Ask the stakeholders the following question: Which Quality Requirements are required?

Describe the Quality Requirements.

Observer Checklist

Participant Code: _____

Elicitation of Partner Systems

Ask for partner systems?	O yes	O no	
SAP?	O accepted	O rejected	O tbc
MS Office?	O accepted	O rejected	O tbc
Travel Reservation Service?	O accepted	O rejected	O tbc
Own Absence List?	O accepted	O rejected	O tbc
Question on extra costs correctly answered?	O yes	O no	

[Absence List and TRS will lead to medium extra costs, SAP and Office are for free]

Ask for details of SAP and MS Office?	O yes	O no	O NA
Ask for details of TRS and Absence List?	O yes	O no	O NA

Elicitation of Technical Infrastructure Components

Ask for operation systems?	O yes	O no	
Solaris OS?	O accepted	O rejected	O tbc
Question on why is this not possible answered?	O yes	O no	O not posed

[Underlying application server only available for windows and linux]

Linux OS?	O accepted	O rejected	O tbc
Ask for database system?	O yes	O no	
MySQL?	O accepted	O rejected	O tbc
Question on extra costs correctly answered?	O yes	O no	O not posed

[medium extra costs]

Ask for user management?	O yes	O no	
Active Directory	O accepted	O rejected	O tbc

Mobile Devices → Workplaces

Ask for details of infrastructure components?	O yes	O no	

Supported Stakeholders

➔ **User Group**

Elicitation of Stakeholder Goals

Ask for stakeholder goals	O yes	O no	
Ask for details of stakeholder goals	O yes	O no	

Elicitation of Business Processes / Stakeholder Tasks

Ask for processes / tasks?	O yes	O no	
Business Travel Process	O accepted	O rejected	O tbc
Ask for details of processes / tasks?	O yes	O no	
Two start events	O accepted	O rejected	O tbc
Question on why this is not possible answered? *[process algorithm in process engine do not support this]*	O yes	O no	O not posed
EPK usage	O accepted	O rejected	O tbc
Question on extra costs correctly answered? *[high extra costs]*	O yes	O no	O not posed

Elicitation of Business Activities / To-Be Activities / System Responsibilities

Ask for activities?	O yes	O no	
Mentioned activities	O accepted	O rejected	O tbc
Ask for classification of activities?	O yes	O no	
Ask for details of activities?	O yes	O no	

Elicitation of Business Data / Domain Data

Ask for data	O yes	O no	
Travel Application Form	O accepted	O rejected	O tbc
Ask for details of data	O yes	O no	

Elicitation of Business Rules

Ask for business rules	O yes	O no	
Mentioned business rule	O accepted	O rejected	O tbc
Question on feasibility correctly answered? *[yes, no problem]*	O yes	O no	O not posed
Ask for details of business rule	O yes	O no	

Elicitation of System Use Cases

Ask for system use cases?	O yes	O no	
Activities of business travel process	O accepted	O rejected	O tbc
Implement business process UC	O accepted	O explained as not needed	
Ask for details of business travel UCs	O yes	O no	
Ask for details of implement UC	O yes	O no	O NA

Elicitation of System Functions

Ask for system function	O yes	O no	
Reminder	O accepted	O explained as not needed	
Question on extra costs correctly answered? *[no extra costs, included anyway]*	O yes	O no	O not posed
Ask for details of system functions	O yes	O no	

Elicitation of User Groups

Ask for user groups basically	O yes	O no	
Ask for users of application & approval	O yes	O no	
Ask for users of booking & accounting	O yes	O no	
Ask for users of implementation activities	O yes	O no	
Ask for details of user group	O yes	O no	

Elicitation of Workplaces

Ask for workplaces	O yes	O no	
Normal PCs	O accepted	O rejected	O tbc
Ask for mobile devices	O yes	O no	
Android?	O accepted	O rejected	O tbc
Question on why this is not possible answered? *[rendering engine only supports Apple so far. Reimplementation of this engine very expensive]*	O yes	O no	O not posed
Question on extra costs correctly answered? *[high extra costs]*	O yes	O no	O not posed
Ask for details of workplace	O yes	O no	

Elicitation of Design Requirements

Ask for design requirements	O yes	O no	
Reuse of C++	O accepted	O rejected	O tbc
Question of hard constraint correctly answered *[it is a hard constraint and not to change]*	O yes	O no	O not posed
Ask for details of design requirements	O yes	O no	

Prepared Requirements and Questions

Elicitation of Partner Systems

R1. The system shall be connected with an SAP system.

R2. The system shall be connected with MS Office.

R3. The system shall be connected with the Travel Reservation Service in the Internet.

R4. The system shall be connected with our own Absence List in der Intranet.

Q1. For which of these systems will there be extra costs?

Elicitation of Technical Infrastructure Components

R5. The system shall run on our servers with Solaris OS.

Q2. Why is this not possible?

R6. Alternative: The system shall run on our Linux servers.

R7: The system shall use our open source DB MySQL.

Q3. What will this cost?

R8. The system shall be connected with our Active Directory as user management.

R?: The system shall be accessible from Android phones.

Q?. Why is this not possible?

Q?. What would it cost to changes this?

Elicitation of Supported Stakeholders

R18. Employees

R18a. For the Application (HSA), Approval (HSA), the system shall support the scientific employees.

R18b. For Booking (HSA), Traveling, Accounting (HSA), the system shall support the administrative employees.

R18c. For the implementation activities, the system shall support the process developers.

Elicitation of Stakeholder Goals

R22. The system should improve our process performance and transparency.

Elicitation of Business Processes / Stakeholder Tasks

R9. The system shall implement our Business Travel Process.

R10. The Business Travel Process shall start either by an employee's wish to go on business trip or a project leaders assignment.

Q4. Why is this not possible?

R11. The Business Travel Process that is currently modeled in the notation of EPK should be imported in the system?

Q5. What would that cost if you would change this?

Elicitation of Business Activities / To-Be Activities / System Responsibilities

R12. The Business Travel Process shall consist of the following activities: Application (HSA), Approval (HSA), Booking (HSA), Traveling, Accounting (HSA)

Elicitation of Business Data / Domain Data

R13. The system shall manage the Travel Application Form (name, destination, date, expected costs)

Elicitation of Business Rules

R14. The system shall implement the following business rule "If the costs are higher than 1000 €, the division head must be involved to give his approval"

Q? Are you sure that this rule can be supported?

Elicitation of System Functions

R17. The system shall provide me a functionality that reminds me via email on the business trips I have to approve

Q6. There are no extra costs for this?

Elicitation of User Groups

R18. For the business travel activities, the system shall support the normal employees.

R19. For the implementation activities, the system shall support the process developers.

Elicitation of Workplaces

R20. The system shall be accessible from the normal desktop PCs.

R?: The system shall be accessible from Android phones.

Q?. Why is this not possible?

Q?. What would it cost to changes this?

Elicitation of Design Requirements

R21. The system shall be enable to reuse our existing business logic components written in C++.

Q7. Is this hard constraint or is this possible to be changed?

Elicitation of Quality Requirements

R23. The system should be reliable and highly available.

Appendix F: Experiment Results

Hypothesentestübersicht

	Nullhypothese	Test	Sig.	Entscheidung
1	Die Verteilung von correct_material_questions ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,214	Nullhypothese beibehalten
2	Die Verteilung von asked_relevant_questions ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,879	Nullhypothese beibehalten
3	Die Verteilung von asked_irrelevant_questions ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,015	Nullhypothese ablehnen
4	Die Verteilung von not_elicited_commonalities ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,570	Nullhypothese beibehalten
5	Die Verteilung von rejected_problematic_requirements ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,004	Nullhypothese ablehnen
6	Die Verteilung von accepted_feasible_requirements ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,570	Nullhypothese beibehalten
7	Die Verteilung von to_be_checked ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,570	Nullhypothese beibehalten
8	Die Verteilung von duration ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,999	Nullhypothese beibehalten
9	Die Verteilung von correct_answers ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,004	Nullhypothese ablehnen
10	Die Verteilung von satisfaction_fit ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,570	Nullhypothese beibehalten
11	Die Verteilung von realization_fit ist in den Kategorien von group identisch.	Kolmogorov-Smirnov-Test bei unabhängigen Stichproben	,125	Nullhypothese beibehalten

Asymptotische Signifikanzen werden angezeigt. Das Signifikanzniveau ist ,05.

Gruppenstatistiken

	group	N	Mittelwert	Standardabweichung	Standardfehler des Mittelwertes
correct_material_questions	M	13	,5654	,20040	,05558
	C	12	,3917	,21829	,06302
asked_relevant_questions	M	13	17,6154	2,21880	,61538
	C	13	16,7692	2,12736	,59002
asked_irrelevant_questions	M	13	,7692	,92681	,25705
	C	13	2,5385	1,12660	,31246
not_elicited_commonalities	M	13	,3846	,36251	,10054
	C	13	,1923	,32522	,09020
rejected_problematic_requirements	M	13	,7600	,27878	,07732
	C	13	,2950	,25497	,07072
accepted_feasible_requirements	M	13	12,0769	1,11516	,30929
	C	13	11,3077	1,43670	,39847
to_be_checked	M	13	1,0769	1,11516	,30929
	C	13	1,6154	,96077	,26647
duration	M	12	28,7500	5,96772	1,72273
	C	13	28,4615	6,91153	1,91691
correct_answers	M	13	,7665	,20260	,05619
	C	13	,3969	,28762	,07977
satisfaction_fit	M	13	,6362	,06049	,01678
	C	13	,5954	,07535	,02090
realization_fit	M	13	,9231	,08788	,02437
	C	13	,8269	,07216	,02001

Test bei unabhängigen Stichproben

		Levene-Test der Varianzgleichheit		T-Test für die Mittelwertgleichheit						
									95% Konfidenzintervall der Differenz	
		F	Signifikanz	T	df	Sig. (2-seitig)	Mittlere Differenz	Standardfehler der Differenz	Untere	Obere
correct_material_questions	Varianzen sind gleich	1,148	,295	2,075	23	,049	,17372	,08373	,00052	,34692
	Varianzen sind nicht gleich			2,067	22,365	,050	,17372	,08403	-,00037	,34781
asked_relevant_questions	Varianzen sind gleich	,258	,616	,993	24	,331	,84615	,85254	-,91340	2,60571
	Varianzen sind nicht gleich			,993	23,958	,331	,84615	,85254	-,91357	2,60587
asked_irrelevant_questions	Varianzen sind gleich	,053	,821	-4,373	24	,000	-1,76923	,40461	-2,60430	-,93416
	Varianzen sind nicht gleich			-4,373	23,140	,000	-1,76923	,40461	-2,60595	-,93251
not_elicited_commonalities	Varianzen sind gleich	,174	,681	1,424	24	,167	,19231	,13507	-,08647	,47108
	Varianzen sind nicht gleich			1,424	23,723	,168	,19231	,13507	-,08664	,47126
rejected_problematic_requirements	Varianzen sind gleich	,114	,739	4,438	24	,000	,46500	,10478	,24874	,68126
	Varianzen sind nicht gleich			4,438	23,811	,000	,46500	,10478	,24865	,68135
accepted_feasible_requirements	Varianzen sind gleich	,154	,698	1,525	24	,140	,76923	,50442	-,27184	1,81030
	Varianzen sind nicht gleich			1,525	22,609	,141	,76923	,50442	-,27524	1,81370
to_be_checked	Varianzen sind gleich	,173	,681	-1,319	24	,200	-,53846	,40825	-1,38104	,30412
	Varianzen sind nicht gleich			-1,319	23,486	,200	-,53846	,40825	-1,38202	,30510
duration	Varianzen sind gleich	,135	,717	,111	23	,912	,28846	2,59300	-5,07558	5,65250
	Varianzen sind nicht gleich			,112	22,909	,912	,28846	2,57728	-5,04421	5,62114
correct_answers	Varianzen sind gleich	2,022	,168	3,788	24	,001	,36962	,09757	,16823	,57100
	Varianzen sind nicht gleich			3,788	21,556	,001	,36962	,09757	,16702	,57221
satisfaction_fit	Varianzen sind gleich	,084	,774	1,521	24	,141	,04077	,02680	-,01454	,09608
	Varianzen sind nicht gleich			1,521	22,929	,142	,04077	,02680	-,01468	,09622
realization_fit	Varianzen sind gleich	,616	,440	3,049	24	,006	,09615	,03154	,03106	,16124
	Varianzen sind nicht gleich			3,049	23,124	,006	,09615	,03154	,03093	,16137

Hypothesentestübersicht

	Nullhypothese	Test	Sig.	Entscheidung
1	Die Mediane von correct_answers sind in den Kategorien von group identisch.	Mediantest bei unabhängigen Stichproben	,001[1,2]	Nullhypothese ablehnen
2	Die Mediane von asked_irrelevant_questions sind in den Kategorien von group identisch.	Mediantest bei unabhängigen Stichproben	,002[1,2]	Nullhypothese ablehnen
3	Die Mediane von rejected_problematic_requirements sind in den Kategorien von group identisch.	Mediantest bei unabhängigen Stichproben	,001[1,2]	Nullhypothese ablehnen

Asymptotische Signifikanzen werden angezeigt. Das Signifikanzniveau ist , 05.

[1] Exakte Signifikanz wird für diesen Test angezeigt.

[2] Exakte Sig. nach Fisher

Agreement or disagreement with regard to the following statements in MG													
Questions	**2**	**4**	**7**	**8**	**10**	**11**	**12**	**15**	**17**	**19**	**21**	**23**	**26**
I could answer all stakeholder's questions by only using the information given in the elicitation instruction	5	5	4	2	2	4	3	2	4	3	5	4	2
I would have required more time for achieving the same results / requirements when not using the elicitation instruction	3	5	4	5	5	5	5	5	4	5	5	5	5
I would not have achieved the same results / requirements without using the elicitation instruction	5	5	5	5	5	3	4	5	4	5	5	5	5
The elicitation instruction allowed me to deviate from it, if necessary	4	2	4	3	3	5	4	3	3	4	4	4	2
The elicitation instruction could always be used as a concrete process to follow	5	3	5	4	4	1	4	4	3	3	5	5	2
The elicitation instruction enabled me to achieve good results / to elicit quite a good set of requirements	5	4	4	5	3	5	4	4	4	4	5	4	4
The elicitation instruction enabled me to get a good understanding of the software product line on which the new system should be built	5	2	4	3	3	4	2	2	4	3	4	5	3
The elicitation instruction informed me about the information (e.g., business processes, data, use cases, interfaces to external systems, etc.) which are relevant to be discussed with the stakeholders	4	5	5	5	3	5	4	5	4	4	4	5	5
The elicitation instruction informs me whether requirements of a certain type do already exist and could be reused instead of elicited from scratch (e.g., reusable use case variants)	4	5	4	5	5	5	4	3	5	5	5	5	4
The elicitation instruction made clear in which order certain elicitation steps should be performed best	5	4	5	4	5	4	5	4	5	4	5	5	5
The elicitation instruction provided me clear how-to guidance (i.e., it clearly mentions a sequence of steps to be carried out during an elicitation process)	5	5	5	5	4	5	5	3	4	5	5	5	3
The elicitation instruction provided me with good indications to know when the elicitation is finished	5	2	4	4	4	3	5	4	4	4	5	4	4
The elicitation instruction provided me with sound knowledge about the given constraints and capabilities I have to consider	5	5	5	4	2	5	4	4	4	4	5	5	5
The elicitation instruction reflected the style in which I would also perform interviews without an elicitation instruction	4	3	4	3	3	4	2	3	4	3	4	3	1
The elicitation instruction was easy to read	4	4	5	5	4	5	5	3	4	5	5	4	2
The elicitation instruction was helpful for performing the interview	5	5	5	5	4	5	5	5	5	4	5	5	4
The elicitation instructions supported me in finding important information quickly	3	5	5	5	4	5	4	2	3	3	5	5	3
The elicitation instructions was easy to handle	3	4	4	4	4	4	5	4	3	4	5	5	4
1 = totally disagree, 5 = totally agree													

Agreement or disagreement with regard to the following statements in CG													
Questions	**1**	**3**	**5**	**6**	**9**	**13**	**14**	**16**	**18**	**20**	**22**	**24**	**25**
I could answer all stakeholder's questions by only using the information given in the elicitation instruction	2	3	3	2	3	3	3	2	2	4	3	2	2
I would have required more time for achieving the same results / requirements when not using the elicitation instruction	4	2	4	4	5	5	5	5	5	3	5	2	5
I would not have achieved the same results / requirements without using the elicitation instruction	5	5	5	5	5	5	5	5	5	2	5	1	5
The elicitation instruction allowed me to deviate from it, if necessary	5	5	3	5	4	5	3	5	3	2	5	3	4
The elicitation instruction could always be used as a concrete process to follow	4	5	4	5	4	5	5	4	4	4	3	4	2
The elicitation instruction enabled me to achieve good results / to elicit quite a good set of requirements	4	5	5	5	4	3	5	4	4	3	5	4	4
The elicitation instruction enabled me to get a good understanding of the software product line on which the new system should be built	2	4	3	4	4	2	4	4	3	4	3	2	2
The elicitation instruction informed me about the information (e.g., business processes, data, use cases, interfaces to external systems, etc.) which are relevant to be discussed with the stakeholders	5	5	4	5	5	5	5	4	4	2	5	4	4
The elicitation instruction informs me whether requirements of a certain type do already exist and could be reused instead of elicited from scratch (e.g., reusable use case variants)	5	4	1	4	3	4	3	3	3	3	3	3	2
The elicitation instruction made clear in which order certain elicitation steps should be performed best	5	5	5	4	4	4	5	1	2	3	4	4	4
The elicitation instruction provided me clear how-to guidance (i.e., it clearly mentions a sequence of steps to be carried out during an elicitation process)	5	5	5	4	4	3	5	4	2	5	4	4	4
The elicitation instruction provided me with good indications to know when the elicitation is finished	4	3	2	5	5	3	5	2	1	4	4	4	2
The elicitation instruction provided me with sound knowledge about the given constraints and capabilities I have to consider	3	5	2	5	5	3	5	4	4	2	3	2	3
The elicitation instruction reflected the style in which I would also perform interviews without an elicitation instruction	2	1	3	4	2	3	2	2	5	4	3	3	3
The elicitation instruction was easy to read	5	5	5	4	5	5	5	3	3	5	5	4	4
The elicitation instruction was helpful for performing the interview	5	5	5	5	4	4	5	4	4	4	5	5	5
The elicitation instructions supported me in finding important information quickly	4	4	5	4	5	3	5	5	1	3	5	3	4
The elicitation instructions was easy to handle	5	2	5	4	5	3	3	4	2	5	5	3	4
1 = totally disagree, 5 = totally agree													

D

POST QUESTIONNAIRE (MG)

Your general opinion: What do you like in the elicitation instruction?

2. The real-world interview which provided a good example of requirements elicitation
4. Klare Sturuktur, vorgegebene Fragen
7. Clear, summarized according to the practical purpose of the experiment easy to scan through, understandable.
8. Sequential list → **good order**, Explicit questions (not just "ask whether…)
10. The order was quite helpful, while it was allowing me as a non-expert in the field to be still capable of asking questions.
11. It was clearly categorized, but the order of the categories could be different. And it gave me good idea what and how to ask my interviewer questions
12. Clear structure, a lot information about possible features / costs for them
15. Guidance (chronological)
17. How to elicit, negotiate the requirements in general, Communication also
19. Hat klare Struktur vorgegeben, welche Fragen gestellt werden müssen. Zusatzinformationen (Was ist eine Business Activity….)
21. The questions, their purposes and constraints with their reasons – everything is very specific in the instruction as it helped me a lot to understand the whole scenario.
23. The step by step idea is great. It was not so hard to follow this instruction. Like a process / like also the idea to go from a big thing into a more detailed and graphic thing.
26. Systematic Questions

What is the most valuable information provided to you by the elicitation instruction?

2. The basic steps to perform an elicitation interview
4. Welche Bedingungen erfüllt sein müssen, was nicht möglich ist.
7. Steps on the elicitation process, clearly defined.
8. Structure, Completeness, Constraints.
10. The system requirements and the possible constraints
11. Pointing out the constraints (with some examples)
12. Information: what to ask?
15. Questions
17. how to gathering the requirements, Communication
19. -----
21. The constraints, reasons behind them and their hardness where the most valuable info as it helped me to structure my responses to the stakeholder.
23. The elicitation of the Business Process with the activities and data.
26. The sample questions to start with and important points.

What should be improved in the elicitation instruction?
2. Separate elicitation instruction and product specification – will make the elicitation easier to understand and read 4. Schritt von 3 bis 9 geht von der Prozess Ebene runter auf die Aktivitätsebene und Schritt 6 geht wieder hoch auf die Prozess Ebene. Vielleicht ist es ein wenig verwirrend zunächst auf einer Ebenen zu bleiben. 7. Some more examples on some steps like: Evaluation of Business Processes, Activities, Data, and System Use Cases. However that will overload the document 8. Constraints list could be marked so that one can find them more easily (in the actual interview) 10. If the questions would be in bold font, the interviewer can ask the questions quicker. 11. The questions checklist could be more explicit and the order could be more abstract. I mean to understand all the Business Goals and the Processes first and then jump to all technical issues. 12. ----- 15. Questions highlighted (I did this in the preparation) 17. ----- 19. ----- 21. Maybe, a graphical representation of process, activity, data and their interrelationships would have helped more to understand the development guideline. 23. Maybe give some more exceptions handlers like: If A then B, If C then D…. 26. Maybe less text, bold text
What did you miss in the elicitation instruction?
2. At some points I couldn't find what I was looking for (product specification details) as fast as I needed. 4. Verbindung zw. System Functions und dem abzubildenden Prozess Non-Functional Requirements 7. ---- 8. ---- 10. Some technical details 11.---- 12. Questions: Standard / Professional version 15. Figures 17. ----- 19. ----- 21. The prioritization of requirements 23. ---- 26. Technical details

Appendix G: Case Study Material

Tailoring Measurement

Performed by: _____

No	Step	Understandability[1]	Applicability[2]	Input[3]	# produced elements	Total effort[4] (person minutes)	Help required?
1	Characterization of Software Product Line (SPL)						
2	Identification of Architectural Element Types						
3	Identification of Architectural Elements						
4	Characterization of Supported Flexibility Classes						
5	Identification of Flexibility Assumptions						
6	Characterization of Development Phases						
7	Identification of Development Activities						
8	Elaboration of Decisions and Corresponding Information Needs						
9	Determination of Relevant Issues						
10	Determination of Conceptual Relationships						
11	Definition of AERE Elicitation Instructions						
	TOTAL						

[1] I understand what I have to do [1 = totally disagree, 2 = rather disagree, 3 = neither agree nor disagree, 4 = rather agree, 5 = totally agree]
[2] I can perform the step without problems [1 = totally disagree, 2 = rather disagree, 3 = neither agree nor disagree, 4 =rather agree, 5 = totally agree]
[3] What has been taken to perform the step [only document, only expert interviews, both]
[4] Including effort for involved experts, if necessary

What do you like in the tailoring approach?

Which parts of the tailoring approach do you find helpful?

What was problematic for you to do (and why)?

What was very easy for you to do (and why)?

Appendix H: Case Study Results

Case Study Resuts

	I understand what I have to do					
	1	2	3	4	5	AVG
Characterization of Software Product Line (SPL)	4	5	5	5	4	4,6
Identification of Architectural Element Types	5	4	4	4	3	4
Identification of Architectural Elements	4	5	4	2	3	3,6
Characterization of Supported Flexibility Classes	4	3	5	4	3	3,8
Identification of Flexibility Assumptions	3	4	5	5	2	3,8
Characterization of Development Phases	4	5	5	5	2	4,2
Identification of Development Activities	5	5	5	5	3	4,6
Elaboration of Decisions and Corresponding Information Needs	5	4	5	5	3	4,4
Determination of Relevant Issues	4	4	4	4	3	3,8
Determination of Conceptual Relationships	4	4	5	5	3	4,2
Definition of AERE Elicitation Instructions	5	5	5	5	2	4,4

1 = totally disagree, 5 = totally agree

	I can perform the step without problems					
	1	2	3	4	5	AVG
Characterization of Software Product Line (SPL)					4	4
Identification of Architectural Element Types					3	3
Identification of Architectural Elements					3	3
Characterization of Supported Flexibility Classes	5	3	4	5	3	4
Identification of Flexibility Assumptions	3	4	3	4	2	3,2
Characterization of Development Phases	5	4	5	5	2	4,2
Identification of Development Activities	5	5	4	5	3	4,4
Elaboration of Decisions and Corresponding Information Needs	4	4	3	4	3	3,6
Determination of Relevant Issues	4	4	5	5	3	4,2
Determination of Conceptual Relationships	5	4	4	3	3	3,8
Definition of AERE Elicitation Instructions	5	5	5	5	2	4,4

1 = totally disagree, 5 = totally agree

	Required time per processsed element (in min)					
	1	2	3	4	5	AVG
Characterization of Software Product Line (SPL)					30,00	6,00
Identification of Architectural Element Types					12,00	2,40
Identification of Architectural Elements					4,50	0,90
Characterization of Supported Flexibility Classes	1,50	1,67	0,83	1,00	8,57	2,71
Identification of Flexibility Assumptions	8,00	3,00	2,20	2,50	13,33	5,81
Characterization of Development Phases	1,67	1,67	1,67	1,00	15,00	4,20
Identification of Development Activities	0,43	1,07	0,71	0,57	1,76	0,91
Elaboration of Decisions and Corresponding Information Needs	1,07	1,07	1,00	2,00	4,09	1,85
Determination of Relevant Issues	0,33	0,38	0,33	0,33	17,14	3,71
Determination of Conceptual Relationships	0,33	0,67	0,28	0,62	13,33	3,05
Definition of AERE Elicitation Instructions	1,00	1,00	1,00	1,00	30,00	6,80

Each column reflects one participant. The numbers in the cells of the first and second sub-table are ratings on the 5-point-Likert-scale where 1=totally disagree, 5=totally agree.

What do you like in the tailoring approach?

1: Easy to use, not much pre-knowledge necessary, good guidance
2: A lot of steps are automatically done or partially done. Each step is separated from the other.
3: Systemantic guidance through elicitation process preparation (only relevant aspects will be elicited in interviews)
4: Guidance and Tool-Tipps, automatic generation

Which parts of the tailoring approach do you find helpful?

1: Guidance through the whole process. Automatic recommendations for single properties within the steps
2: Definitions of questions for the application eengineer, rich of mining an issue is reduced
3: in the tool, the instructions are very helpful as well as the questions that should be asked

What was problematic for you to do (and why)?

1: N/A
2: mapping of RE-elements to architectural elements, because it is hard to match between solution and problem space
3: initially, it was problematic to assign the correct issues. In general, the steps 2-5 and 9 were compared to 6-8 a bit more complicated as there were concepts I'm not that familiar with. However, the provided guidelines were helpful to understand what to do.
4: Conceptual Model (Relationships / Cardinalities), Understanding of Architectural Element Types, Reference Issues

What was very easy for you to do (and why)?

1: Interviewing Expert through clear advices / guidance by the tool
2: mapping of elements to phases was quite self-explaining since the method has separated the elements very clear
3: especially steps 6 to 8 were easy as I'm also familiar with these concepts.
4: Confirmation of autogenerated content

Feedback:

- Abstrakte und interessante Herangehensweise an die Anforderungserhebung
- Teilweise schwer auf Produkt abzubilden, da Unterscheidung zwischen Customizing und neue Komponente teilweise schwierig zu ziehen ist.
    - Customizing bei Produkt bezieht sich nicht nur auf Parameter vorgeben, sondern auch auf programmatische Anpassungen
- Bedienung gewöhnungsbedürftig
- Für Leute, die nicht sehr vertraut sich mit dieser Art der Erfassung erfordert es mehr als einen Durchlauf

Appendix I: Project Analysis (State of Practice)

This table shows data of five industrial AE projects in which a document analysis software was individually adapted and integrated at the customer's site.

	Projekt-Messwerte							
		Projekt 1	Projekt 2	Projekt 3	Projekt 4	Projekt 5	Durchschnitt
Projekt	**Projektdauer in Monaten**	13	9	9	21	11	13
	Projektgröße in PT	630	350	180	1000	170	466
	RE-Aufwand in PT (regulär)	90	50	25	100	35	60
RE	**Länge RE in Wochen** (regulär)	10	8	10	4	4	7
	Anzahl RE-Workshops (regulär)	12	7	10	15	5	10
	Anzahl RE-Abstimmungen (regulär)	30	8	0	0	0	8
Anforderungen	**Anzahl vereinbarter Anforderungen**	600	400	8	800	19	365
	Explizit antizipiert Anforderungen (ca.)	50%	50%	50%	80%	80%	62%
	Implizit antizipiert Anforderungen (ca.)	25%	30%	25%	10%	10%	20%
	Nicht antizipierte Anforderungen (ca.)	25%	20%	25%	10%	10%	18%
	Verworfene Anforderungen	5%	5%	10%	3%	0	5%
	Von Experten geprüfte Anforderungen	30%	20%	0%	100%	0%	30%
Ver-handlung	**Nachverhandlungsgespräche** (zusätzlich)	30	10	2	5	1	10
	Nachverhandlungsaufwand in PT (zusätzlich)	30	10	5	30	1	15
	Nachverhandlungsgespräche nach Implementierungsbeginn	15	6	0	1	0	4
	Anteil der Nachverhandlungen im RE in PT	25%	17%	17%	23%	3%	17%
	Anteil der Nachverhandlungen an allen RE-Terminen	42%	40%	17%	25%	17%	28%
Er-fahrung	**Wissen der REler über die Produktplattform**	teilweise	wenig	vollständig	vollständig	viel	viel
	Wissen der REler über Entwicklungsstrategie	wenig	wenig	vollständig	vollständig	viel	viel
	Projekterfahrung	teilweise	teilweise	teilweise	teilweise	viel	teilweise

	Durchschnittliche und interpolierte Werte			
Mittel-werte	Erhobene Anforderungen pro RE-Workshop / -Abstimmung	21	Stück	
	RE-Workshops / -Abstimmungen pro Woche	2,4	Stück	
	Erhobene und spezifizierte Anforderungen pro PT im RE (Effizienz)	6,1	Stück	
	Von Experten geprüfte Anforderungen	110	Stück	
Interpolierte und hochgerechnete Werte	Nachverhandelte Anforderungen vor Impl. (interpoliert über Anforderungen pro PT)	93	Stück	25%
	Nachverhandelte Anforderungen pro Nachverhandlungsgespräch vor Impl.	10	Stück	
	Verzögerungen durch Nachverhandlungen vor Impl. (interpoliert über RE-Workshops pro Woche)	4,0	Wochen	
	Verzögerung durch Nachverhandlung vor Impl. pro nachzuverhandelnder Anforderung	0,04	Wochen	
	Nachverhandelte Anforderungen nach Impl. (interpoliert über nachver. Anforderungen pro Gespräch)	42	Stück	
	Verzögerungen durch späte Nachverhandlungen (Annahme eines Faktor 5)	9,1	Wochen	
	Insgesamt nachverhandelte Anforderungen	135	Stück	37%
	Insgesamte Verzögerung durch nachzuverhandelnde Anforderungen	13,1	Wochen	26%

Appendix J: Calculation of Expected Improvements

As shown in Appendix I, only about 60% of the requirements in an AE project are explicitly anticipated on average in practice. For the elicitation of the other 40%, there is no explicit support. Hence, the fit of these requirements depends on the experience of the involved people and therefore partially on luck. We assume that only half of these requirements fit straightaway (50:50 chance) and do not lead or rework or renegotiations. Thus, only 80% of the elicited requirements are expected to fit directly in today's practice, while 20% of the requirements have to be renegotiated. This is even a quite optimistic assumption, as the interpolation shown in Appendix I comes to the result that almost all requirements that are not explicitly anticipated (37%) have to be reworked in a project, which will take about 26% of the entire project duration. For the rest of this calculation, however, we act on the assumption that about 20% of the calendar time spent in a project is needed due to non-fitting requirements in an average AE project today. According to our project analysis shown in Appendix I, these 20% amount to about 10 weeks.

However, when using the thesis tailoring approach, requirements engineers are explicitly informed about the feasibility of (the 20%) implicitly anticipated requirements and about the constraints that exist for (the 20%) non-anticipated requirements. Thus, it is assumed that the fit of only 10% of the requirements still depends on luck and has to be validated by involved SPL experts, which would result in rework for only about 5% of the requirements. In sum, 95% of the elicited requirements are therefore expected to fit directly, which would be an improvement of about 18% compared to today's practice. However, as the actual distribution of explicitly anticipated, implicitly anticipated, and non-anticipated requirements may vary among different SPLs, we act on the assumption that a **fit improvement of 15%** is realistic on average.

Thus, when using the thesis approach, only 5% of the requirements would have to be renegotiated. Furthermore, as the requirements engineers would be aware of the requirements that have to be checked by an SPL expert, it is assumed that all re-negotiations could even be done before the start of the implementation (which is a significant benefit compared to today's practice). According to the project data shown in Appendix I, this rework would lead to an overall delay of about one week. Hence, in contrast to today' practice, 90% of the delay could be saved, which would result in an overall reduction of time to market by again 18%. However, as the actual distribution of requirements may vary among different SPLs, we act on the assumption that a **time-to-market reduction by 15%** (approx. two months per project year) is more realistic.

Appendix K: Initial Issue List

	Requirements Specification				Business Modeling		Misc.	
	IEEE 830	IEEE 1362	IEEE1233	Volere	ARIS	Zachman	RUP	TORE
Business Activity					X	X	X	X
Business Area							X	
Business Event				X	X	X	X	
Business Object						X	X	
Business Objects	X			X	X	X	X	X
Business Process				X	X	X	X	X
Business Role					X	X	X	
Business Rule						X	X	
Business Service				X		X	X	
Human System Activity	X	X	X	X		X	X	X
Interaction Data	X		X					X
Operation Modes	X	X	X					
Organizational Unit		X			X	X	X	
Partner Systems	X	X	X	X	X		X	
Physical Backend Environment	X	X	X	X				
Project	X	X	X	X			X	
Quality Characteristic	X	X	X	X			X	
Realization Policy	X	X		X			X	
Regulation	X		X	X			X	
System Function	X		X	X				X
System Interface	X		X			X	X	
System-System Interactions	X		X	X				
Technical Infrastructure Component	X	X	X	X		X	X	
UI Area						X	X	X
UI Style				X				
Usage Profile	X	X	X					
User Role	X	X	X	X			X	
Workplace			X	X		X		

Lebenslauf

Name	Sebastian Adam	
Anschrift	Lessingstraße 7 67663 Kaiserslautern	
Geburtsdatum	03.12.1979	
Geburtsort	Idar-Oberstein	
Familienstand	ledig	
Staatsangehörigkeit	Deutsch	
Schulbildung	1986-1990 1990-1999	Grund- und Hauptschule Westrich, Baumholder Gymnasium an der Heinzenwies, Idar-Oberstein Abschluss: Abitur
Zivildienst	1999-2000	Zivildienst im Kinder- und Jugendheim der „kreuznacher diakonie“, Niederwörresbach
Studium	2000-2005	Studium der Angewandten Informatik Technische Universität Kaiserslautern Abschluss: Diplom
Berufstätigkeit	2005-heute	Wissenschaftlicher Mitarbeiter Fraunhofer IESE, Kaiserslautern

Kaiserslautern, den 19.12.2012

PhD Theses in Experimental Software Engineering

Volume 1	**Oliver Laitenberger** (2000), *Cost-Effective Detection of Software Defects Through Perspective-based Inspections*
Volume 2	**Christian Bunse** (2000), *Pattern-Based Refinement and Translation of Object-Oriented Models to Code*
Volume 3	**Andreas Birk** (2000), *A Knowledge Management Infrastructure for Systematic Improvement in Software Engineering*
Volume 4	**Carsten Tautz** (2000), *Customizing Software Engineering Experience Management Systems to Organizational Needs*
Volume 5	**Erik Kamsties** (2001), *Surfacing Ambiguity in Natural Language Requirements*
Volume 6	**Christiane Differding** (2001), *Adaptive Measurement Plans for Software Development*
Volume 7	**Isabella Wieczorek** (2001), *Improved Software Cost Estimation A Robust and Interpretable Modeling Method and a Comprehensive Empirical Investigation*
Volume 8	**Dietmar Pfahl** (2001), *An Integrated Approach to Simulation-Based Learning in Support of Strategic and Project Management in Software Organisations*
Volume 9	**Antje von Knethen** (2001), *Change-Oriented Requirements Traceability Support for Evolution of Embedded Systems*
Volume 10	**Jürgen Münch** (2001), *Muster-basierte Erstellung von Software-Projektplänen*
Volume 11	**Dirk Muthig** (2002), *A Light-weight Approach Facilitating an Evolutionary Transition Towards Software Product Lines*
Volume 12	**Klaus Schmid** (2003), *Planning Software Reuse – A Disciplined Scoping Approach for Software Product Lines*
Volume 13	**Jörg Zettel** (2003), *Anpassbare Methodenassistenz in CASE-Werkzeugen*
Volume 14	**Ulrike Becker-Kornstaedt** (2004), *Prospect: a Method for Systematic Elicitation of Software Processes*
Volume 15	**Joachim Bayer** (2004), *View-Based Software Documentation*
Volume 16	**Markus Nick** (2005), *Experience Maintenance through Closed-Loop Feedback*

Volume 17 **Jean-François Girard** (2005), *ADORE-AR: Software Architecture Reconstruction with Partitioning and Clustering*

Volume 18 **Ramin Tavakoli Kolagari** (2006), *Requirements Engineering für Software-Produktlinien eingebetteter, technischer Systeme*

Volume 19 **Dirk Hamann** (2006), *Towards an Integrated Approach for Software Process Improvement: Combining Software Process Assessment and Software Process Modeling*

Volume 20 **Bernd Freimut** (2006), *MAGIC: A Hybrid Modeling Approach for Optimizing Inspection Cost-Effectiveness*

Volume 21 **Mark Müller** (2006), *Analyzing Software Quality Assurance Strategies through Simulation. Development and Empirical Validation of a Simulation Model in an Industrial Software Product Line Organization*

Volume 22 **Holger Diekmann** (2008), *Software Resource Consumption Engineering for Mass Produced Embedded System Families*

Volume 23 **Adam Trendowicz** (2008), *Software Effort Estimation with Well-Founded Causal Models*

Volume 24 **Jens Heidrich** (2008), *Goal-oriented Quantitative Software Project Control*

Volume 25 **Alexis Ocampo** (2008), *The REMIS Approach to Rationale-based Support for Process Model Evolution*

Volume 26 **Marcus Trapp** (2008), *Generating User Interfaces for Ambient Intelligence Systems; Introducing Client Types as Adaptation Factor*

Volume 27 **Christian Denger** (2009), *SafeSpection – A Framework for Systematization and Customization of Software Hazard Identification by Applying Inspection Concepts*

Volume 28 **Andreas Jedlitschka** (2009), *An Empirical Model of Software Managers' Information Needs for Software Engineering Technology Selection A Framework to Support Experimentally-based Software Engineering Technology Selection*

Volume 29 **Eric Ras** (2009), *Learning Spaces: Automatic Context-Aware Enrichment of Software Engineering Experience*

Volume 30 **Isabel John** (2009), *Pattern-based Documentation Analysis for Software Product Lines*

Volume 31 **Martín Soto** (2009), *The DeltaProcess Approach to Systematic Software Process Change Management*

Volume 32 **Ove Armbrust** (2010), *The SCOPE Approach for Scoping Software Processes*

Volume 33 **Thorsten Keuler** (2010), *An Aspect-Oriented Approach for Improving Architecture Design Efficiency*

Volume 34 **Jörg Dörr** (2010), *Elicitation of a Complete Set of Non-Functional Requirements*

Volume 35 **Jens Knodel** (2010), *Sustainable Structures in Software Implementations by Live Compliance Checking*

Volume 36 **Thomas Patzke** (2011), *Sustainable Evolution of Product Line Infrastructure Code*

Volume 37 **Ansgar Lamersdorf** (2011), *Model-based Decision Support of Task Allocation in Global Software Development*

Volume 38 **Ralf Carbon** (2011), *Architecture-Centric Software Producibility Analysis*

Volume 39 **Florian Schmidt** (2012), *Funktionale Absicherung kamerabasierter Aktiver Fahrerassistenzsysteme durch Hardware-in the-Loop-Tests*

Volume 40 **Frank Elberzhager** (2012), *A Systematic Integration of Inspection and Testing Processes for Focusing Testing Activities*

Volume 41 **Matthias Naab** (2012), *Enhancing Architecture Design Methods for Improved Flexibility in Long-Living Information Systems*

Volume 42 **Marcus Ciolkowski** (2012), *An Approach for Quantitative Aggregation of Evidence from Controlled Experiments in Software Engineering*

Volume 43 **Igor Menzel** (2012), *Optimizing the Completeness of Textual Requirements Documents in Practice*

Volume 44 **Sebastian Adam** (2012), *Incorporating Software Product Line Knowledge into Requirements Processes*

Software Engineering has become one of the major foci of Computer Science research in Kaiserslautern, Germany. Both the University of Kaiserslautern's Computer Science Department and the Fraunhofer Institute for Experimental Software Engineering (IESE) conduct research that subscribes to the development of complex software applications based on engineering principles. This requires system and process models for managing complexity, methods and techniques for ensuring product and process quality, and scalable formal methods for modeling and simulating system behavior. To understand the potential and limitations of these technologies, experiments need to be conducted for quantitative and qualitative evaluation and improvement. This line of software engineering research, which is based on the experimental scientific paradigm, is referred to as 'Experimental Software Engineering'.
In this series, we publish PhD theses from the Fraunhofer Institute for Experimental Software Engineering (IESE) and from the Software Engineering Research Groups of the Computer Science Department at the University of Kaiserslautern. PhD theses that originate elsewhere can be included, if accepted by the Editorial Board.

ISBN 978-3-8396-0514-1

Fraunhofer IESE

Technische Universität Kaiserslautern
AG Software Engineering

PhD Theses in Experimental Software Engineering