Masterthesis

A Framework for Goal-oriented Process Configuration

Context-Specific Configuration of SE processes in SPEM

Philipp Diebold

9/19/2012

Department of Computer Science

University of Kaiserslautern

in Cooperation with

Fraunhofer Institute for Experimental Software Engineering

First Examiner:

Prof. Dr. Dr. h.c. Dieter Rombach

Second Examiner:

Dr.-Ing. Andreas Jedlitschka (IESE)

Fraunhofer IESE

Statement of Authorship:

Except where reference is made in the text of "A framework for goal-oriented process configuration", this contains no material published elsewhere or extracted in whole or in part from a thesis presented by me for another degree or masters. No other person's work has been used without due acknowledgement in the main text. This thesis has not been submitted for the award of any other degree or masters in any other tertiary institution.

Bretzenheim, September 19, 2012

(Philipp Diebold)

Acknowledgements

This master thesis and the research presented would have never been possible without the support and understanding of my family and friends. I was very lucky to have friends and colleagues who supported me in different ways in these exhausting months and during my whole study.

In the first place, I thank Dr. Andreas Jedlitschka from the Fraunhofer IESE for being the supervisor of this thesis and for supporting me during the research and writing period. He offered me the opportunity to do this thesis within the context of the ARAMiS project which is partly handled by the Fraunhofer IESE and the Software Engineering Research Group: Processes and Measurement. It was very interesting and stirring to work in a big project together with industrial partners.

I would also like to thank Dr. Jens Heidrich from the Fraunhofer IESE and Dr. Christiane Plociennik from the Software Engineering Research Group of the University of Kaiserslautern. Jens Heidrich was my first contact person at the IESE and he offered me some possible thesis topics and then referred my supervisor, Andreas Jedlitschka. In addition Christiane Plociennik was a big help for me, because she is the person responsible for the ARAMiS Project from the Software Engineering Research Group of the University. She also supported me in some tasks, e.g., selection of technologies provided by the industry partners, by offering me help from her research assistant(s).

Special thanks go to all the friends and my family who directly supported me and my research work. Some of my fellow students assisted me as a kind of counterpart for discussing ideas and implementations. Especially I want to give thanks to Kevin Krüger, Thilo Rauch and Nena and Eric Egli for reviewing and proof-reading of the written report.

Executive Summary

This research for this thesis was conducted to develop a framework which supports the automatic configuration of project-specific software development processes by selecting and combining different technologies: the Process Configuration Framework. The research draws attention to the problem that while the research community develops new technologies, the industrial companies continue only using their well-known ones. Because of this, technology transfer takes decades. In addition, there is the fact that there is no solution which solves all problems in a software development project. This leads to a number of technologies which need to be combined for one project.

The framework developed and explained in this research mainly addresses those problems by building a bridge between research and industry as well as by supporting software companies during the selection of the most appropriate technologies combined in a software process. The technology transformation gap is filled by a repository of (new) technologies which are used as a foundation of the Process Configuration Framework. The process is configured by providing SPEM process pattern for each technology, so that the companies can build their process by plugging into each other.

The technologies of the repository were specified in a schema including a technology model, context model, and an impact model. With context and impact it is possible to provide information about a technology, for example, its benefits to quality, cost or schedule. The offering of the process pattern as output of the Process Configuration Framework is performed in several stages:

I	**Technology Ranking:**	1	Ranking based on Application Domain, Project & Impact
		2	Ranking based on Environment
		3	Ranking based on Static Context
II	**Technology Combination:**	4	Creation of all possible Technology Chains
		5	Restriction of the Technology Chains
		6	Ranking based on Static and Dynamic Context
		7	Extension of the Chains by Quality Assurance
III	**Process Configuration:**	8	Process Component Diagram
		9	Extension of the Process Component Diagram
		10	Instantiation of the Components by Technologies of the Technology Chain
		11	Providing process patterns
		12	Creation of the process based on Patterns

The effectiveness and quality of the Process Configuration Framework have additionally been evaluated in a case study. Here, the Technology Chains manually created by experts were compared to the chains automatically created by the framework after it was configured by those experts. This comparison depicted that the framework results are similar and therefore can be used as a recommendation.

We conclude from our research that support during the configuration of a process for software projects is important especially for non-experts. This support is provided by the Process Configuration Framework developed in this research. In addition our research has shown that this framework offers a possibility to speed up the technology transformation gap between the research community and industrial companies.

Zusammenfassung

Diese Forschungsarbeit wurde durchgeführt um ein Framework zu entwickeln, welches automatische einen Projekt-spezifischen Software Entwicklungsprozesses durch das Auswählen und Kombinieren verschiedener Technologien bereitstellt: das Process Configuration Framework. Das adressierte Problem in der Forschung ist dabei, dass in der Forschung neue Technologien entwickelt werden, diese die Industrie jedoch nicht verwendet. Daraus ergibt sich, dass der Transfer von Technologien meist Jahrzehnte dauert. Zusätzlich ist es der Fall, dass es keine Einheitslösung für ein Software Projekt gibt. Dies bedeutet, dass für ein Project mehrere Technologien kombiniert werden müssen.

Das in dieser Arbeit entwickelte und erläuterte Framework behandelt ein Großteil dieser Probleme, durch eine engere Verknüpfung von Industrie und Forschung sowie das Unterstützen von Software Unternehmen bei der Auswahl der passenden Technologien für den Software Entwicklungsprozess. Dabei wird die fehlende Technologie Transformation durch das Technologie Archiv ersetzt, welches im Process Configuration Framework verwendet wird. Der Prozess wird durch die SPEM Prozess Pattern konfiguriert, welche für jede Technologie einzeln bereitgestellt werden. Damit ist es den Unternehmen möglich den Prozess individuell zu erstellen.

Die Technologien wurden in einem speziellen Schema erstellt, welches das Technologie-, Kontext sowie das Einfluss-Modell beinhaltet. Durch den Kontext und den Einfluss ist es möglich Informationen der Technologien bereitzustellen, wie zum Beispiel Vorteile der Qualität, Kosten oder Zeit. Das Bereitstellen der Prozess Pattern als Ergebnis des Process Configuration Frameworks wird in mehreren Stufen durchgeführt:

I	**Technology Bewertung:**	1	Bewerten von Anwendungsdomäne, Projekt & Einfluss
		2	Bewerten von Umgebung
		3	Bewerten von Statischem Kontext
II	**Technology Kombination:**	4	Erstellung aller möglichen Technologieketten
		5	Einschränkung der Technologieketten
		6	Bewerten von Statischem und Dynamischem Kontext
		7	Erweiterung der Ketten um Qualitätssicherung
III	**Prozess Konfiguration:**	8	Prozess Komponenten Diagramm
		9	Erweiterung des Prozess Komponenten Diagramm
		10	Instanziierung der Komponenten durch Technologien der Technologiekette
		11	Bereitstellen der Prozess Pattern
		12	Erstellung des Prozesses basierend auf den Pattern

Zusätzlich zu der Entwicklung des Process Configuration Framework wurde die Effektivität und Qualität durch einen Fallstudie evaluiert. Dabei wurden von Experten manuell erstellte Technologieketten mit den automatisch erstellten Ketten des Frameworks mit den Konfigurationen der Experten verglichen. Dieser Vergleich zeigte, dass die Ergebnisse des Frameworks ähnlich waren und somit ein guter Vorschlag sind.

Die Folgerung aus der Forschungsarbeit war die Wichtigkeit der Unterstützung der Prozesskonfiguration von Software Projekten, im speziellen für Nichtfachleute. Diese Unterstützung wird durch das in dieser Arbeit entwickelte Framework bereitgestellt. Außerdem wurde gezeigt, dass das Framework die Möglichkeit bietet die Technologietransformation zwischen Industrie und Forschung zu beschleunigen.

Table of Contents

Table of Figures

Table of Tables

Table of Rules

1 Introduction

The domain of software engineering has advanced rapidly over the last decades. Today, software is a part of almost every other industry, such as automotive, telecommunication or health care [Bir97]. To address requirements of the different domains and individual organizations, a large number of different methods, techniques and tools (technologies) (e.g., [KB09]) evolved. Some address the whole development process, for example, Requirements-Based Engineering, whereas others address a tiny aspect during the development, such as a safety analysis method. Each technology[1] is assumed to have a specific impact on either product quality, development costs, or project schedule. In many domains, aspects of product quality, such as safety, are becoming important key performance indicators along with project costs and schedule [Jed09].

Typically, models such as the CMMI [CKS11] do not explicitly define which technologies have to be applied. Further complicating the situation, "there is no one-size-fits all", or as S. Fraser wrote in [FM08], there is no silver bullet available in software engineering. This implies that it is almost impossible to find a technology that supports all needed elements of individual software projects. Thus the combination of appropriate technologies to be used for a project is a challenging task.

Some technologies are a better fit for specific projects than others. This leads to the problem of finding the most appropriate technologies for the specific projects. This choice is very difficult because of different selection criteria, e.g., context and different needs of the stakeholders. Some criteria focus on quality aspects and others try to optimize the costs or schedule. All these aspects have to be considered when configuring a "best-fit" development process. This complex task of selecting the technologies for configuring the development process is often applied by experts (or a group of experts in the Software Engineering Group).

In addition to this dependence on experts, there is the problem of making new technologies availability to industry. This is because typically, "technology transfer takes on the order of 15 to 20 years to mature a technology to the point that it can be popularized and disseminated to the technical community at large" [RR85]. Also most often new technologies are directly borrowed from friendly or rival companies [JCF07].

To solve these problems, we propose a means to support the selection of well-suited (research) technologies and configuring a context-specific process: the framework for goal-oriented process configuration. Our research focuses on process patterns and combines the most appropriate technologies for a specific project context in order to configure the most appropriate process. This solution should not replace the expert. Rather it should provide non-experts a guideline for the technology selection and process creation.

[1] Technology definition of [BCR01a]

1.1 Research Approach

During the work for this research, several research methods such as literature review and experiments were applied. The next figure (cf. Figure 1) summarizes these different steps and gives an overview of the flow of the research work. It contains six major steps.

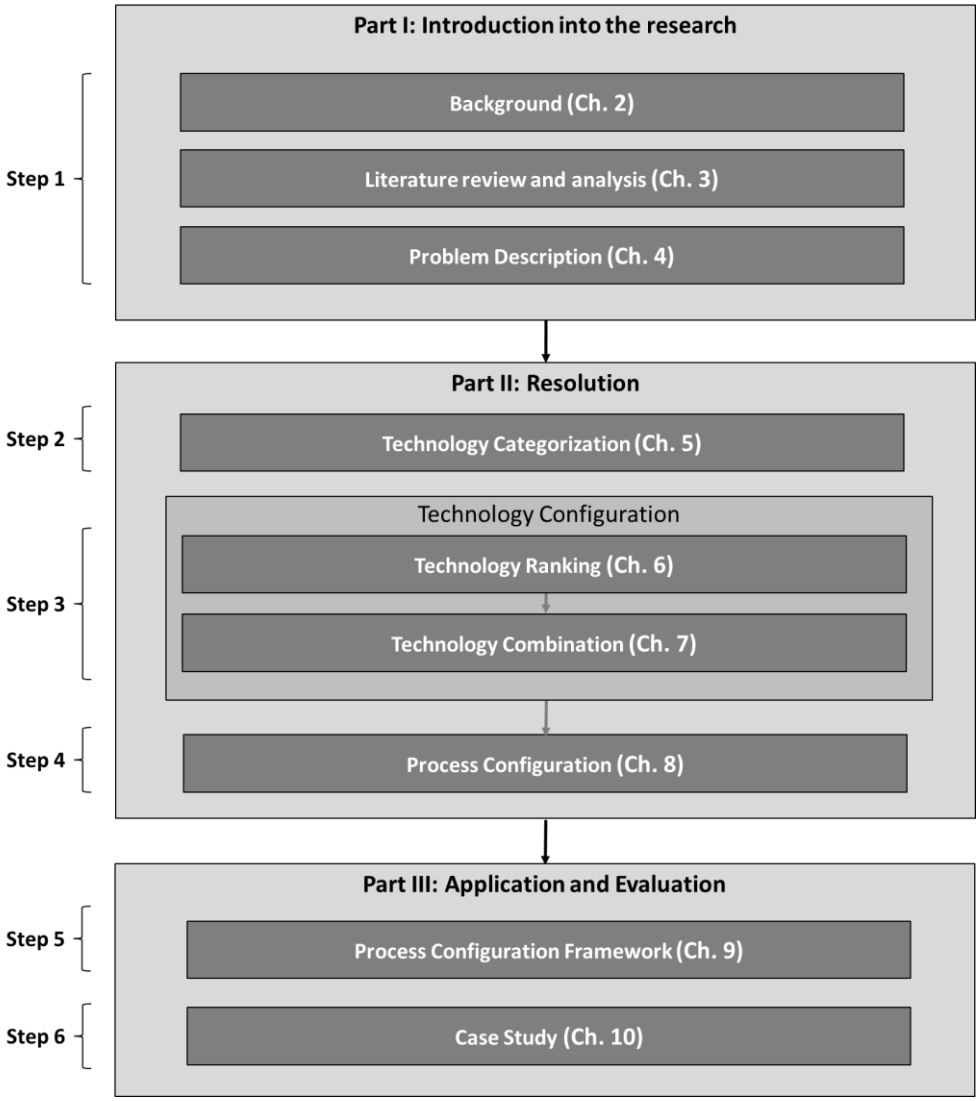

Figure 1. Workflow of the Research

In Part I: **Introduction into the research**:

In step one, we investigate the exact problem by using different research approaches. We conduct a systematic literature review in order to find more details about the domain and problem. These findings about the state-of-the-art research led to a precise picture of the domain and how similar problems were solved.

In Part II: **Resolution**:

The second step deals with the categorization of the technologies. We adapted the categorization schema of [Jed09] with regard to our findings of step one. This adaption includes only those attributes and elements that were required to change the schema for needs of the following steps. In doing this, we assume that technologies are described and stored according to the characterization scheme evolved from step two.

In step three, we select the Technology Chain best fitting to the specific project. In particular, we develop the concept for the ranking of the technologies with regard to their appropriateness based on specific inputs for context and impact. Therefore, we conceptualized a prioritization mechanism for the different attributes, based on rules. Further, we present how we combine the ranked technologies to Technology Chains by specifying combination parameters. The combination is done for the entire software life-cycle, by considering whether a technology fits to prior phases and to the users input.

Step four, the last step of the resolution, is about the process configuration. Based on the outcome of this step, we specify context-specific and optimal process patterns for each software project. This will be realized with a general modeling method, the Software & Systems Process Engineering Meta-model Specification [OMG08].

In Part III: **Application & Evaluation**:

Part three contains two additional research steps, which combine the approaches of part two and also deals with an evaluation of the Process Configuration Framework.

The fifth step contains the explanation of the overall Process Configuration Framework. In particular, we explain the phase-based (and dynamic) approach of the framework by using the approaches presented in part two.

Step six presents the evaluation of the Process Configuration Framework. We conducted a case study with experts from Fraunhofer IESE[2]. Due to their roles they are involved in industry projects thus having relevant experience.

[2] Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern

1.2 Outline of the Thesis

This research workflow is presented in three different parts:

Part I Introduction into research

This part introduces the research.

Chapter 2 Provides an overview of the research background.

Chapter 3 Describes the state-of-the-art by a discussion about the related work.

Chapter 4 Gives a detailed description of the problem which will be solved and analyzed in the other two parts.

Part II Resolution

This part describes the different steps of the technology ranking and combination right up to the process configuration.

Chapter 5 Provides a description of the technology schema based on [Jed09] and specifies its use for this specific research.

Chapter 6 Describes the ranking of the most appropriate technologies based on context and impact input of the software company.

Chapter 7 Describes the concatenation of technologies of the different life cycle phases and combines them to a Technology Chain.

Chapter 8 Specifies the Technology Chain of Chapter 7 as process patterns, based on the process meta-modeling language SPEM [OMG08].

Part III Application & Evaluation

The last part describes the application of the developed framework in Part II and their evaluation.

Chapter 9 Builds the two different Process Configuration Framework approaches based on the stages of Part II.

Chapter 10 Evaluates the developed framework by performing a case study.

Chapter 11 Summarizes the contributions and describes the limitations and lessons learned.

Part I: INTRODUCTION INTO RESEARCH

The main objective of this thesis is to provide a Process Configuration Framework. This framework configures a context-specific software development process in SPEM by using a number of most appropriate technologies. This research is located at the intersection of software processes and decision-making, in particular, technology selection and combination. The portion of the thesis introduces the research areas.

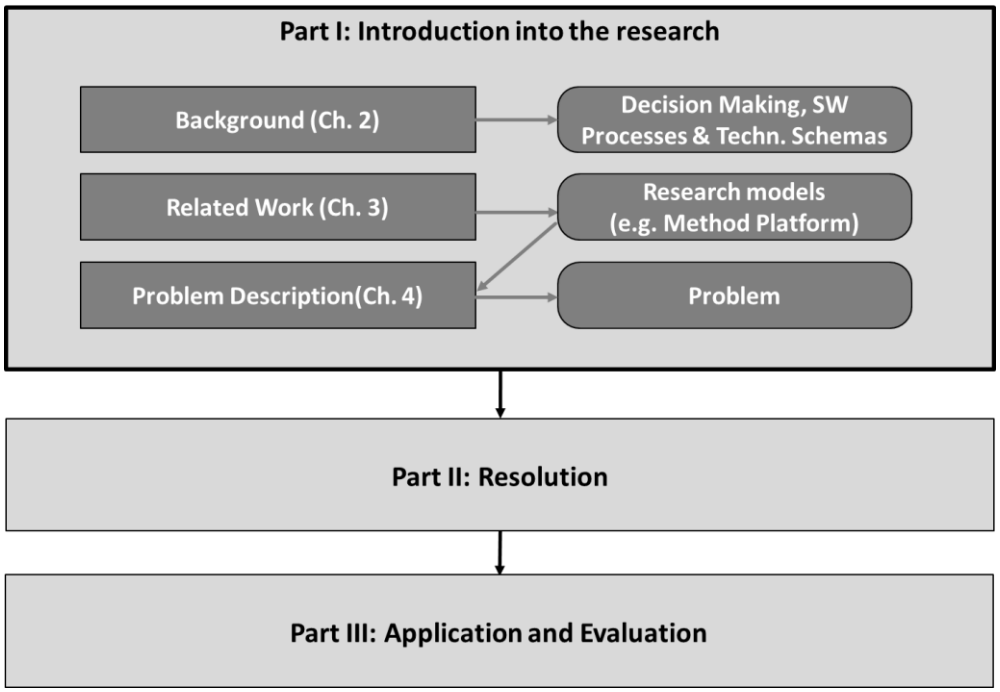

Figure 2. Workflow of Part I: Introduction into the Research

The "Background" chapter (cf. Chapter 2) deals with the background knowledge of decision theories, the software development life-cycle as well as software processes. The decision theory is needed for the selection of the technologies (cf. Section 2.1) as it is for the combination of the technologies of the different life-cycle phases (cf. Section 2.2.1). Software processes are needed for the last stage of the framework, which should transfer the best technologies of the chain to process patterns. One outcome of this chapter refers to the theory of decision-making process. The section on software processes yields knowledge about processes and their modeling. The last background section deals with existing categorization schemas of technologies, how they evolve and the one used in our research.

The chapter "Related Work" (cf. Chapter 3) is about the state-of-the-art in this research area. The topic of this thesis arises from the German research project ARAMiS [KIT11], which includes a section about a continuous development method, more precisely a method platform (German: "Methodenplattform"). Therefore, this chapter includes a systematic literature review of this topic.

The final chapter, “Problem Description“ (cf. Chapter 4), defines the problem addressed in this thesis. This is accomplished by drawing conclusions from the findings of the previous two chapters which are based on different literature research methods.

2 Background

The purpose of this chapter is to describe the background of our research. This simplifies the understanding of the context in which this thesis takes place.

The main findings of this chapter are:

- *Understanding of the background regarding decision theory and software processes*
- *Explanation of the life-cycle phases for a software development project*
- *Explanation of SPEM and the concepts of this language used in this thesis*
- *Introduction of different technology categorization schemas and their evolution*

The first section starts with a brief overview of the topic of decision-theory (cf. Section 2.1). This is necessary to get a better understanding for later decisions about technology ranking and combination. Those will be treated in PART II: RESOLUTION, especially Chapter 6 and 7.

The second section starts with a brief overview of software processes which will be followed by a detailed description of the software development life-cycle. This is very important for the combination of the technologies in a later chapter (cf. Chapter 7). The last subsection deals with aspects of process modeling, where we focus on a Meta modeling language called SPEM.

For later needs the last section contains information about the existing technology categorization schemas. Within this section the most important schemas are briefly introduced. In addition to this, the evolution and merging of the schemas over time is depicted. The latest and one of the biggest schemas of Jedlitschka [Jed09] *is then particularized in Chapter 5 because it is the one used in our research.*

2.1 Decision Theory and Decision-Making

The task of decision making and the decision theory are everyday tasks, because everyone needs to decide between different possibilities in daily life. These decisions are mainly solving everyday problems by doing the thing, which seems to be the right one. These kinds of decisions in a more formal way are also used in research in many different domains, not only software engineering.

Applying different approaches, such as the experience factory [BCR01a] combined with the Quality Improvement Paradigm (QIP) [BCR01b], is important in software engineering. The decision making incorporates those approaches because the experience factory provides a holistic approach for learning from existing (software project) knowledge. Those can be reused for the decisions of the subsequent projects.

Decision support should be applied throughout the entire life-cycle of the software development, as explained by [Ruh03]:

> "For the requirements, analysis, design, construction, testing and evolution phases, decision makers need support to […] rank, select or reject candidate products, processes or tools."

There are several approaches for performing the decision making, regardless of the life-cycle phase. The most used approaches in decision theory will be introduced in the next paragraphs.

One of the most common decision-making approaches deals with fuzzy logic [Ros10]. This theory started in 1965 with [Zah65] and is based on many-valued and probabilistic logic. In contrast with traditional logic, which has only true or false values, fuzzy logic variables have a truth value between 0 and 1. This means it deals with a kind of partial truth. The representation of such values is mainly based on different functions. For example the meanings of cold, warm and hot are represented by functions mapping a temperature scale. Each point of the scale has a number of "truth values", where each of those values is representing one function.

In contrast to the approaches based on fuzzy logic there are some approaches using probabilistic decision theory. This theory is based on probability (e.g., of experts or combined with knowledge base) or different technologies working with probability, such as probabilistic networks. Example approaches which are often used are stochastic petri nets [Mar89]. In addition there are also other probabilistic approaches, e.g., probabilistic decision trees [KM86].

The third theory type has already been mentioned as extension of the probabilistic ones, knowledge-based decision theory [Doy02]. Approaches working with this type of theory, need some kind of knowledge, such as context data of different projects or other data, e.g., from the experience factory [BCR01a]. The usage of data-driven approaches is a chance and weakness together because it is difficult to get (and access) the data. But with this data from some experience base the decisions are more precise. "Data-driven decision making must not cause us to lose sight of the larger context, what I call knowledge-based decision making." [Doy02]

In addition to the three decision theories mentioned, there are several other theories in the literature. These other theories are generally interconnected and often a combination of several theories.

There are different approaches for implementing decision theory. The rule-based systems [Hay85] are based on top of the above mentioned theories. This is the case because within this approach, rules need to specify how the decision takes place. In these rules different decision theories can be used (e.g., rules for implementing fuzzy logic decisions). All of the rule-based approaches are structured in a very similar way. They first contain a *condition part* describing what needs to be fulfilled for applying the rules (similar to the IF-part in programming languages). Second, they include an *action part* describing the action when the condition is fulfilled (similar to the THEN-part in programming languages). In addition to the rule-based approaches there exists also similar ones dealing e.g., with decision tables.

The rule-based systems have been mentioned in this chapter because this is the implementation of the decision system we are using in this thesis. The decision theory mainly used in this research is based on existing knowledge as it will be explained later. For developing and implementing the rule we use Drools Expert [JBo12], the rule engine of Drools – a business logic integration platform.

2.2 Software Processes

In the domain of software engineering, processes assume a major role because quality aspects and project goals are more and more influenced by using project specific processes.

Processes are defined in several different ways, and some of these definitions will be covered in the next paragraphs. Most of the different software (development) process definitions deal with the context. Other similarities are the transformation of input products into outputs, possible refinements with sub-processes as well as performance by humans, machines or both together [Hei11].

An abstract definition of a process is given by [Ost87]: "While a process is a vehicle for doing a job, a process description is a specification of how the job is to be done. Thus cookbook recipes are process descriptions while the carrying out of the recipes are processes." This definition is especially effective because it is based on a familiar activity.

In [FH93] a process is defined as "a set of partially ordered steps intended to reach goal" which is a generic definition, not only software engineering specific. There are many possible definitions, but this one best fits this research project as it includes steps and goal orientation. In [Lon93] a software process model is defined as "An abstract software process description. It can be more or less formal." This definition is an improvement of the process definition of [FH93], which also deals with the process description as a specification. Such software process models use different notations (e.g., graphical or textual [natural language, machine readable]) and different abstraction levels (e.g., live-cycle level, engineering process level or atomic step level). In general we use the highest abstraction level in this research, the life-cycle level. But the static approach (cf. Section 9.2) as well as the more dynamic approach (cf. Section 9.3) can also be used in lower levels, e.g., inside one SE phase. In addition to this level we introduce a standardized general modeling notation used for processes, SPEM [OMG08].

The next two sections cover the aspects of the software life-cycle phases and the SPEM modeling notation. First in the subsection about life-cycle phases (cf. Section 2.2.1) the different phases of a software project will be explained. Then Section 2.2.2 contains the details of the *Software & Systems Process Engineering Meta-Model* published by OMG in 2008 [OMG08].

2.2.1 Life-cycle- / SE Phases

The different life-cycle phases of a software project are a major part of the logical separation of different project activities. Each of the SE phases concludes in the completion of a document / SE object and the achievement of a milestone.

A general software development project contains several different life-cycle phases, which can be referred to different names in literature. Two very important software engineering books of Summerville [Sum07] and Jalote [Jal08] specify SE phases in a similar way. For example Jalote in [Jal08] provides the following four phases:

> *Requirements Analysis*, *Software Design*, *Coding* and *Testing*

Summerville [Sum07] uses similar phases:

> *Software Specification*, *Software Design* & *Implementation*, *Software Validation* and *Software Evolution*

In addition to phase definitions provided by important software engineers there are also standards, e.g., the ISO 12207 [ISO08]. This international standard provides five different activities around the software life-cycle, whereas the most important for us are the development activities. This activity includes seven steps which can be aggregated to the definition of life-cycle phases we are using in this thesis:

Specification/Requirements, Architecture, Design, Implementation, Verification & Validation

The comparison of Table 1 between the different phases of [ISO08], [Sum07] and [Jal08] shows, that some of the provided phase classifications are more detailed than others. Therefore, we tried to use a classification taking all needed aspects into account. Although some of the literature classifications combine them to one phase, Table 1 shows that the first five phases of our classification are consistent with the other ones. Only the verification phase is new. This is needed for assuring that all aspects of quality assurance [Rak01] can be covered by the schema, because validation only covers the testing aspects.

Table 1. Different life-cycle phase classifications

Summerville [Sum07]	Jalote [Jal08]	ISO 12207 [ISO08]	Diebold
Specification	Requirements Analysis	Requirements	Specification
Design & Implementation	Design	High level Design	Architecture
		Module Design	Design
	Coding	Coding	Implementation
Validation	Testing	Module-, Integration- & System testing	Validation
-	-	-	Verification
Evolution	-	-	-

The starting phase is the *Specification*. This phase does not necessarily need input to start, because it is the initial phase, and often the problem description is elaborated with the customer at this time. The output of this phase is a specification document often called the requirements specification document. It includes all the functionality of the software and constraints about it. This phase can be subdivided into two major aspects, the problem understanding and then documenting the requirements.

The next phase, called *Architecture*, is not directly specified in the other classifications. In [Jal08] and [Sum07] the architecture is included in the Design phase. In contrast, [ISO08] specifies it as High level design because it focuses on divide and conquer and the high-level interaction. This phase works on planning a solution for the problem based on the requirements document. The output of this step is an architecture document/model, which is usually a set of diagrams (e.g. UML diagram). This phase is the step from the problem domain into the solution domain [RII05]. Within this phase the software system is divided into subsystems and components to get a better understanding of the structure of the system. The focus of this is on the interaction between the different components to behave as a single system.

In contrast to the architecture phase, the *Design* phase deals with lower level concepts. The design is sometimes called detailed or module design (e.g., [ISO08]). As the architecture covers the aspect of *what* modules are needed, the design specifies *how* the modules can be implemented. The focus of this phase is on designing the logic of the individual modules. The inputs of this phase are the architecture document as well as (optionally) the specification. The output consists of the design document/model, which contains the detailed description of each module to simplify the implementation in the next phase.

After completing the design, the *Implementation* takes place. Because the previous phases contain the major decisions, the main implementation task is the translation of the architecture and design into source code. During this translation, programming language specific details need to be decided. Implementation has a big influence on the following phases because of its strong interconnection with the code. After the completion of the code, the integration of the different modules into one single system is also part of this phase.

The *Testing* phase, also called *Validation*, is one of the two quality assurance phases. The goal of this phase is the detection of software defects in the code. This is done in different levels of abstraction. On the lowest level there is *module testing* where the different components are tested individually. Afterwards they are integrated, which is validated by using *integration testing*. The last testing part is the *system testing*. This tests the system against the system requirements that were developed during the specification phase. The *acceptance testing* works similar to the system testing but it is performed in the real environment. The outputs of this phase are the test cases including test data. Some software companies document the test cases which are implemented in the testing phases during specification.

Verification is the second quality assurance phase. It is the only one with connection to most of the other phases, because it verifies the output against the output of the previous phases, e.g., the requirements document against the problem description. Therefore, in this phase all inputs are possible. Verification should detect uncovered aspects, errors, etc. of the actual document. This is done for example with inspection techniques [GG94].

2.2.2 Process Meta-Model

Today there exist several different process models. All these different models use various notations and specifications. Therefore, the Object Management Group (OMG)[3] developed a specification which is a standard for modeling processes, called *Software & Systems Process Engineering Meta-Model* (*SPEM*) [OMG08]. Because the developed framework will have process patterns as an output, we want to model this output with a standard. We use the SPEM notation and explain it in the following paragraphs.

In general, meta-modeling is the construction of rules, constraints, models and theories applicable and useful for modeling classes of problems. A meta-model is a model one abstraction layer higher than the original model, which gives the possibility of defining models. The definition of such meta-models is derived from the *MetaObjectFacility* (MOF) [OMG11] by the OMG.

SPEM was first developed by the OMG in version 1.0 in 2002 and then refined in 2008 [OMG08]. It is a meta-model based on MOF. Because it supports concepts for modeling, documenting, presenting, exchanging and executing of methods and processes, we chose this model. The structure of SPEM is depicted in Figure 3, where the different packages were interconnected with a merge-relationship.

[3] Consortium for modeling and developing model-based standards

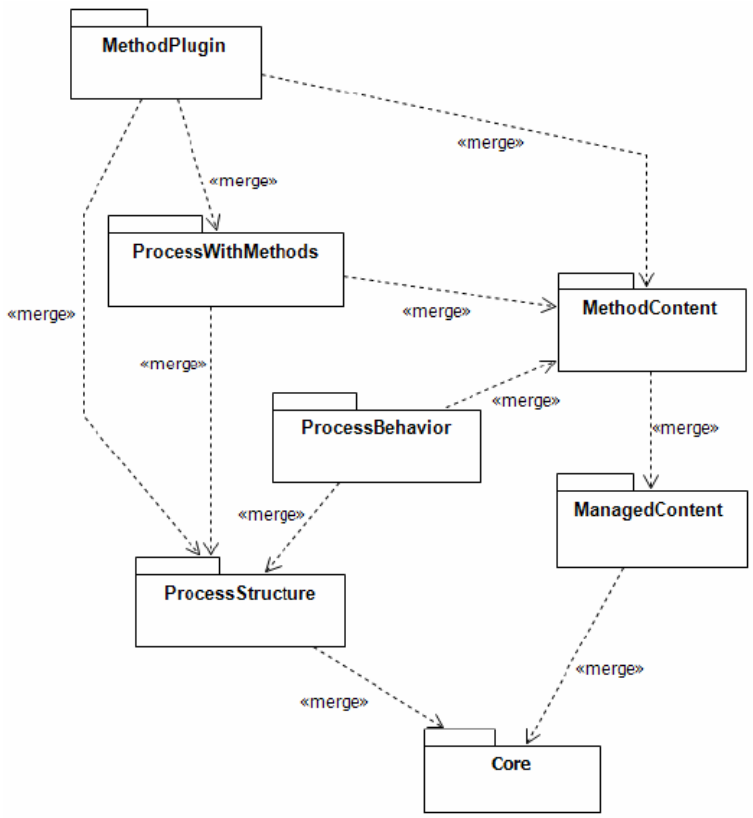

Figure 3. Structure of SPEM 2.0 Meta-Model [OMG08]

Figure 3 shows the basic package on the bottom, the *Core*. It contains the basic classes and abstractions for all other packages of the meta-model. The *ProcessStructure* is built upon the Core and contains the basic classes for defining process structures. The *ProcessBehaviour* is an extension of the ProcessStructure by using the MethodContent. The goal of this package is enhancement by defining execution semantics. This package also gives the possibility of extending elements with external behavioral models. The *ProcessWithMethods* merges the same packages as the ProcessBehavior but with the focus on method elements. This package enables the possibility of modeling processes as method element instances. The elements of the MethodContent serve as the definition point, which can be instantiated in processes. The package *ManagedContent* gives the possibility of combining textual descriptions with process element. An extension of the ManagedContent is the *MethodContent*, which enables the possibility of defining basic elements for modeling methods (e.g. roles, work products and tasks). The last package is the *MethodPlugin*. By merging the ProcessStructure, ProcessWithMethods and MethodContent, this package uses all possibilities of SPEM. It enables the concepts of method libraries and processes, which increase reuse. The reuse aspect works mainly by using same elements of the method content for several processes. During this instantiation an adaption of the elements is possible. The later parts of the framework using SPEM are based on the *MethodPlugin*.

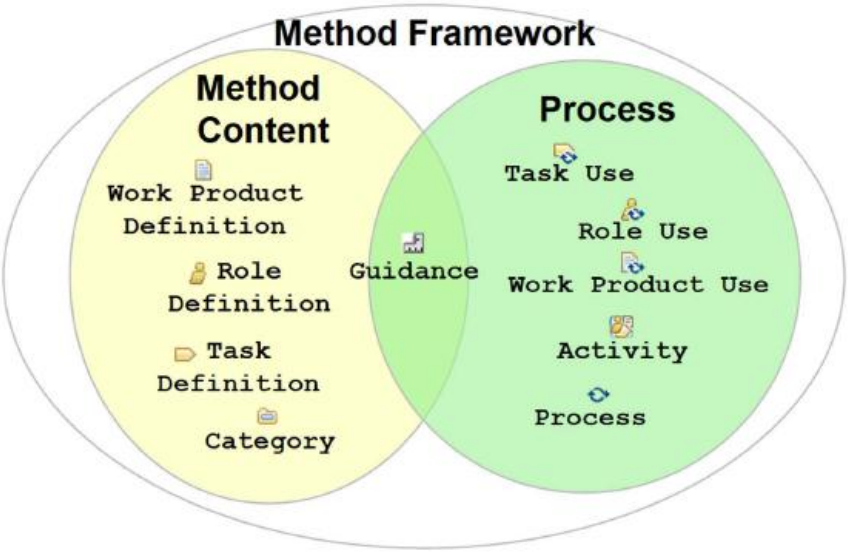

Figure 4. Key terminology of SPEM mapped to Method Content versus Process [OMG08]

Inside the SPEM specification there is the differentiation of static (method content) and dynamic content (processes) as shown in Figure 4. This is derived from the Rational Unified Process (RUP) [EM03]. The Method Content elements are used for defining elements and describing *what*, *how* and *who*. Examples are Roles, Work Products and Tasks. Processes are more focused on the project flow. They instantiate elements of the method content for defining processes. Each usage of an element creates a new instance (Task-, Role- and Work Products Use). Figure 4 gives an overview of all the different elements of SPEM and their affiliation.

Not all the different elements of Figure 4 are explained in this thesis. Only the concepts used in this thesis will be introduced when they are needed, further information can be found in [OMG08]. Using all the possible elements, SPEM supports different kinds of modeling processes. Most important to our work are the process patterns and their refinements explained in [OMG08]. They will be used in Chapter 8 because the outcomes of the framework are those process patterns.

2.3 Technology Categorization Schemas in Software Engineering

The software domain has a wide range of technologies and schemas which results from the different phases of the development life-cycle (cf. Section 2.2.1). One example is the categorization schema for selecting software testing techniques [Veg02], but there are also many reasons for the numerous different categorization schemas.

For the framework we are developing in this thesis, a general technology categorization schema is needed because the approach covers technologies from all different life-cycle phases. Therefore, some of the existing schemas are irrelevant for our work, although they will be briefly used in the next paragraph to show the evolution of such schemas.

In literature we find a huge number of existing schemas, which overlap or have merged over time to become complete. Figure 5 provides an overview of the most important categorization schemas over time. The figure also contains the relationship between them:

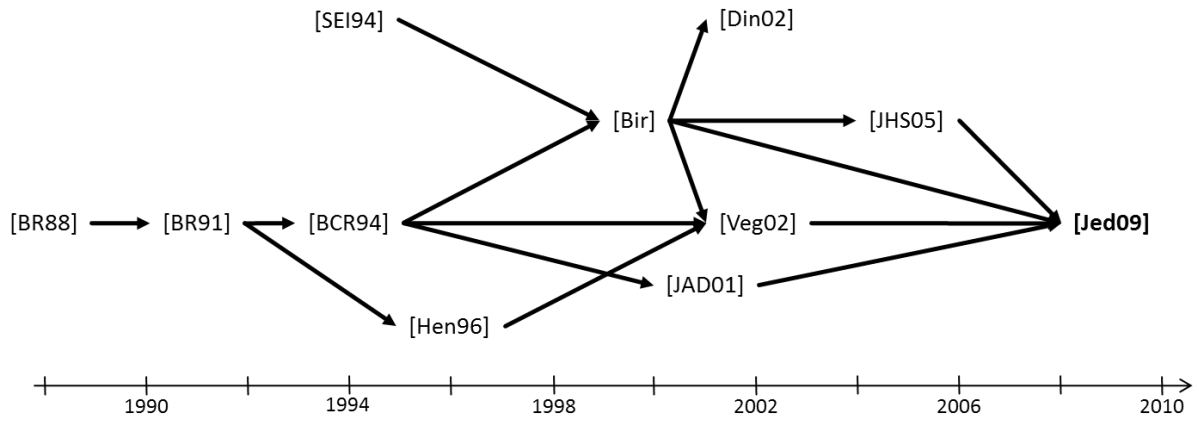

Figure 5. Evolvement of Technology Categorization Schemas

The beginning of technology schema development (Figure 5, left) started in the late 80's by Basili and Rombach [BR88]. This was then refined twice. The first time was in 1991 [BR91] and the second revision, in 1994, was called the *experience factory* approach. ([BCR94] is the edition which preceded [BCR01a]). Simultaneously to the experience factory approach of Basili, Rombach and Caldiera, the Software Engineering Institute (SEI) published the *C4 Software Technology Reference Guide* (SEI C4 TRG) [SEI97], which provides a catalogue of around 60 technologies. Both starting points result into two different lines. The line started by the experience factory results in several publications (e.g. [Hen96], [JAD01]). The work of the SEI was not used much. The first important publication using the SEI C4 TRG was [Bir] by Birk. This publication was the first to merge both lines into one schema. Afterwards, more publications appeared using Birk's work as a starting point. [Veg02] and [Din02] use it for their research.

The newest schema is [Jed09] by Jedlitschka developed in 2009 (Figure 5, right). This is a refinement and also generalization of the approach of Birk. However, this schema also incorporates publications of Jedlitschka: [JAD01] and [JHS05]. Because [Jed09] is the result of all the evolution of the different schemas we will use and adapt this schema for our needs. This adaption is explained in Chapter 5.

3 Related Work

This chapter provides state-of-the-art overview of the domain we are working in and where the research takes place.

The main purposes of this chapter are:

- *Understanding of method platform and similar frameworks*
- *Comparison between different method platforms and their realization*
- *Description of existing Process Creation or Process Configuration approaches*

The first section contains a systematic literature review regarding method platform and similar frameworks. This is done in the formal and systematic way of [Kit04]*, and all the results of this will be presented in this section.*

The remaining section deals with the process creation and process configuration. Within this, we want to discuss existing approaches for creating and configuring software processes. Because most of the publications in literature are about business process creation, this section is rather short.

3.1 Method Platform – A literature review

Platforms can be considered a collection of similar things. Such platforms were used in almost every domain for clustering special things. This is especially true in the software domain, in which methods, technologies, and tools were clustered in a separate platform.

To get an overview of existing method platforms and equivalent frameworks, we reviewed the existing literature to gather existing knowledge about such frameworks. In this thesis we are also developing such a framework/platform. The following systematic literature review was performed similar to the procedure of [Kit04].

3.1.1 Background – Method Platform

Before defining the objectives of this review, we need to define a method platform in order to determine which frameworks belong to it or similar to it, and which do not belong to it.

For *method platforms* there are no definitions given in the literature, which means we cannot refer to one given definition but have to define it ourselves. As mentioned in the beginning of this section, a platform is collection of things, similar to a repository or knowledge base. In this case, the platform contains methods and technologies, which are sometimes seen as the same thing. In this research we take the technology definition given by [BCR01a]. Therefore, the platform must contain technologies. In addition to this kind of repository, such platforms have an input as well as an output. Most often a single technology or a selection of technologies is the output of such method platforms. The selection of this output is mainly done based on the input and some decision-making in between. For such platforms there is no unique and general input similar to the technology as output. In addition to the input and output, some kind of decision-making is necessary to belong to the domain of method platforms. There are several ways to decide which of the technologies in the platform are best suited to the input. Some examples for decision making have already been introduced in Section 2.1.

One purpose of this research is the characterization of a method platform seen from different literature sources in the domain of computer science, particularly in software engineering. The other purpose of this systematic review is the comparison of existing method platforms. This comparison is done by using different aspects, e.g., the domain, input, decision-making and output.

For this reason, the objective of this literature review is to answer the following research questions:

1. What is currently known about method platforms and their benefits and limitations?
2. What are these platforms used for?
3. In what way do they differ and what impact does this difference have for the method platform? What are the advantages and disadvantages of the different method platforms?

3.1.2 Review Method

Informed by the established method of systematic reviews [Kit04], we separate the review in different stages:

- research questions
- inclusion criteria
- exclusion criteria
- search strategy
- quality criteria
- data extraction

Because the research questions had already been defined in the previous subsection, we then dealt with the different inclusion and exclusion criteria. Papers were eligible for inclusion in this review if they presented something about a method platform or similar frameworks. This was checked by using the definition of Section 3.1.1. The various search terms were specified in order to determine whether a publication included a method platform. Those can be found three paragraphs below. It was also necessary for the papers to pass the minimum threshold defined by the quality criteria in a later paragraph. This systematic review only included papers published after year 2000. This was because method platforms and similar frameworks were not available before that time, which we discovered during the first search of different databases. As additional inclusion we considered publicatiosn only in English because most of the literature in the searched databases is in English. Publications in German and other languages do not consider this topic in the software domain.

Studies were excluded if their (main) focus has nothing to do with a method platform or a similar framework. It was sometimes difficult to get the main focus and it was possible that some inappropriate papers were chosen. Furthermore, all papers that did not belong to the domain of computer science were excluded. There were several papers which also belonged to another domain (e.g. electronics) because of the strong interconnection of the different domains. These papers were included because they contain information about method platforms used in computer science as well as the application domain.

The search strategy included several electronic databases which included the conferences and journals which are the most relevant for this topic. During the first stage of selection we found the most important papers based on the number of publications. These following electronic databases were searched:

- ACM Digital Library
- IEEE Xplore
- ScienceDirect – Elsevier
- SpringerLink

Figure 6 shows the systematic review process and the number of papers identified at each stage. In Stage 1, all the metadata and abstracts of the publication in the included electronic databases were searched using the following search terms:

(1) "method platform"
(2) "technology selection"
(3) "technology selection framework"
(4) "technology decision framework"
(5) "process selection framework"

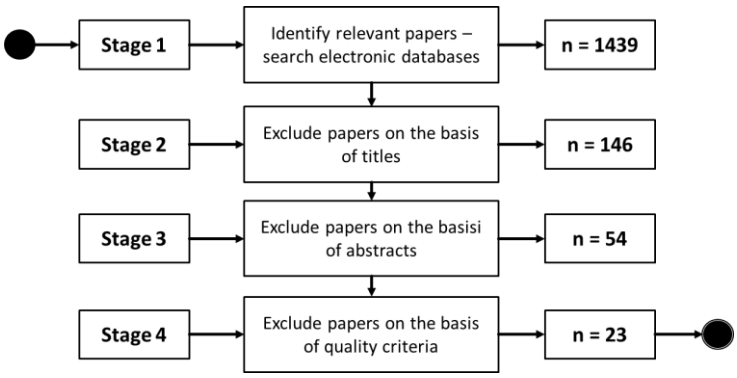

Figure 6. Stages of the paper selection process

All these search terms for articles which belong to the topic of method platforms were combined using the Boolean "OR". This means the article needed to include only one. This results in the search term of:

1 OR 2 OR 3 OR 4 OR 5

This paragraph contains information about the citation management, retrieval and inclusion decisions used during the stages of the selection process.

Relevant citations from Stage 1 (n = 1439) were entered and stored in Mendeley[4]. This was done to get an overview of all the relevant papers and to record the source of each citation. A similar categorization was done for the other subsequent steps of the paper selection process. All data of relevant papers from the different stages of the process were saved in specific files and were presented in the Appendix A.

[4] Software for managing and sharing research papers, discovering research data and collaborating

At Stage 2 (n = 150) we went through all the titles of the papers that resulted from Stage 1, to determine their relevance regarding method platforms. At this stage, papers that were clearly not about method platforms or similar frameworks were excluded. For example, because our search strategy included the term "technology … platform", we got several results dealing with different kinds of technology platforms. Papers with titles that indicate clearly that the papers were outside the scope of the systematic review were excluded. Sometime the titles are no clear indicators of the papers topic. In such cases we included the papers for reviewing in the next stage because it could not be decided whether they fit the scope or not. At the end of this second stage only 150 papers were included, which means almost 90% of the papers of the first stage were excluded because of their title.

At Stage 3 (n = 54), papers were excluded if their main focus had nothing to do with method platform or selection framework. This was indicated by means of the paper's abstract. During this stage, we found abstracts of variable quality. Some of the abstracts were missing, poor and/or misleading. If the topic was unclear from the title, abstract, and keywords, the papers were included for a detailed quality assessment in the last step.

After Stage 3, the 54 remaining papers were assessed according to five criteria. The following criteria are adaptations and selections of those given in [Pyr99] and [TF84]. The adaptation and selection of only a part of these criteria was needed because they have a huge range. We only used some general parts and specific questions about instruments, which was in our case the platform. Those five criteria cover the main quality issues that needed to be considered when validating the different publications identified during the review. The quality assessment issue was the trustworthiness of the data and information about the method platforms presented in the papers. We included several criteria that were related to the quality of describing method platforms or similar frameworks, their aims and contexts. Therefore, each study was assessed according to whether:

1. The title of the paper was sufficiently specified.
2. The researcher began by specifying the problem area. Was the importance of the specific problem domain shown?
3. There was a detailed description of the methodology or at least another source where additional information could be obtained.
4. There was some kind of explanation about the input, the decision-making and the output of the method platform. (Similar to the definition of Section 3.1.1)
5. There was some evidence (e.g. for temporal stability, content validity, empirical validity) given about this approach. Was an Experiment, Case Study or something similar performed?

Table 2. Data extraction form

Paper / Publication Description		
1.	Paper identifier	Unique id for the specific paper
2.	Date of data extraction	
3.	Publication Year	
4.	Publication Type	Journal article / conference paper
5.	Publication Source	Which journal / conference?
6.	Publication Title	
Method Platform(or similar framework)		
1.	Name	
2.	Input	
3.	Decision-making method	What theory is the base for the decision? (e.g., fuzzy logic, , expert-based, … see Section 2.1)
4.	Output	
5.	Description	Or reference of the description
6.	Used in	Application domain
7.	Used for	Purpose of the method platform
Reported Results		
1.	Benefits	
2.	Drawback	
3.	Evidence	e.g., authors assumption, lessons learned, observed effect in laboratory/practice, confirmed hypothesis, ...

During quality assessment, we extracted the data from each of the 23 papers remaining after Stage 4. The data was included in this systematic review according to a predefined extraction sheet (cf. Table 2). This form made it possible to record the full details of each paper and to specify how each of them addressed our research questions. Some papers did not provide all the data we tried to record in Table 2. For this reason the data table in the appendix (cf. Appendix A) has some empty attributes. The publication description, data about the method platform (e.g. name, input, output, etc.) and the reported results by the authors of the publications were noted in a qualitative and descriptive analysis using MS Excel[5].

In contrast to similar systematic literature reviews, we did not perform a synthesis of the findings because the purpose of this review was scoping of the domain of method platforms. We provided an overview of different platforms in this domain. These findings are presented in the next subchapter using mainly descriptive analysis technique [Tro06].

3.1.3 Results

We finally identified 23 papers on the topic of method platform or similar frameworks. Most of the selected papers did not contain the term method platform but a related term, e.g., selection or decision framework. The number of resulting papers was about 2% of the papers after the first stage (n = 1439) of selecting publications from the electronic databases by using specific search terms. The specific data for the different stages was also separated for the different databases and can be found in Appendix A.

As already mentioned in the review method section, namely in the inclusion and exclusion part, we considered only papers which belonged to the computer science domain. Thus this does not mean that there were papers which used parts of computer science for the method platform but generally belonged to another domain.

[5] Spreadsheet application for calculating and different kinds of analysis

The analysis of the domains of the different approaches specified in the publications resulted in eight different domains. These domains were *computer science (9), manufacturing (3), economics (2), ecology (2), electrical engineering (2), health care (2), human resources (1)* and *telecommunication (1)*. The most common domain was the computer science domain with almost 39% of the publications. Within those nine papers some were general papers, others are from bioinformatics. The majority belonged to *software engineering*. In these six publications, main parts of the software engineering life-cycle were covered, e.g., testing (3), verification & validation (1) and architecture (1).

Because the papers arose from different domains, they were published in wide range of conferences and journals, which are also in different domains. Only a small number of them were mentioned more than once: Conferences on *Empirical Software Engineering (2), Power Engineering (2), Communication (2), Management of Engineering & Technology (2)* and Journals on *Information & Technology (3), Systems (3).*

The year of publication was also relevant for this analysis because the topic could be outdated. We started the analysis of method platforms in 2000 because there was no appearance in the literature before that date. The trend over the years is shown in Figure 7. It shows the published papers regarding the discussed topic from 2000 until today. Although there were not that many publications, over the last decade the number of papers has increased (cf. Figure 7, trend marked as dashed line).

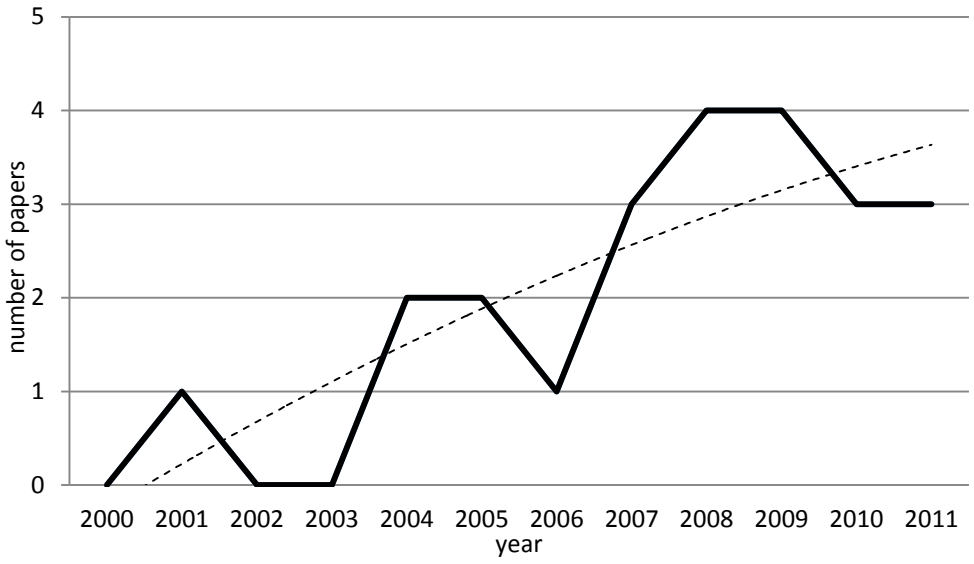

Figure 7. Method Platform trend over time

More important than the analysis of the occurrence of method platforms over time are the attributes of method platforms and how they are satisfied. These are the input, output and the decision-making in between, which is derived from the definition given above.

The definition of method platforms given above starts with the input for such frameworks. This input can be of any kind. The different kinds of inputs collected from the publications are *technologies (7), context (6), problem or goal (5), defects, database, boundary conditions, user preferences* and *requirements (each 1)*. It is obvious from the number of indications, that there are three main input possibilities: (a set of) technologies, the context or the problem/goal. Those possibilities can be classified in two categories. One has a set of technologies as input and selects the most appropriate output. The other group uses context, goal or problem as an input for selecting the best output. In this case there exists already a set of technologies where the selection takes place.

The next part of such a platform is the decision-making process, which is the most important step. This is the case because in this part the decision takes place based on the input. It needs to be decided which of the technologies is the most appropriate and will be provided as output. This is easier with using the second category of input because the platform can select the optimal technology quite directly from the specific context, problem or goal which is given. The other category instead gives a set of technologies but no direct hint how to select the optimal one. In this case it is more difficult to decide because there is no input simplifying the decision. In addition there were different decision-making approaches used in the publications: *fuzzy logic (7), rule-based (3), probability-based (2), TOPSIS*[6] [SSL07] and other approaches which can be seen in Appendix A. Most of the frameworks use fuzzy logics, e.g., fuzzy sets, fuzzy AHP[7] or fuzzy TOPSIS [BR06]. The second and third most often used approaches are rule- or probability based approaches. We also assume that most of the method platforms use some kind of rule-based approach on top which is not explicitly specified in the publications. Approaches using probability values appear also in some frameworks. The probability-based ones work with some kind of prediction, which is based on the probabilities, context and knowledge from previous input-output-combinations.

The specification of the output of such method platform in literature contains no different outputs. This is the case because almost half of the 23 resulting papers directly use a technology as output and the others something similar which is related to technologies, e.g., combination of technologies or ranked technology list.

3.1.4 Conclusions

The method platforms and similar frameworks presented above are spread over a wide range of (sub) domains and almost a decade of time. They give a good overview regarding what is important for such platforms and how this can be realized. As given in the definition in the background section, they need an input, a decision-making process mainly using the input, and an output (cf. Figure 8). The input can be a set of technologies or some kind of context, problem or goal description. The processing of the input takes place in the decision-making process, which is responsible for selecting the output. For the realization of this part there are different approaches and they were introduced briefly in Section 2.1 about decision theory. The output is a technology or something related to this.

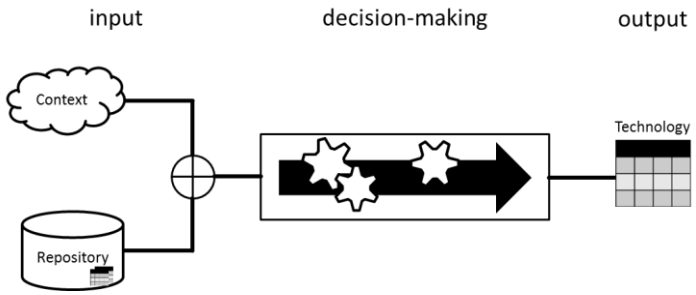

Figure 8. Graphical Representation of a Method Platform

[6] Technique for Order Preference by Similarity to Ideal Solution
[7] Analytic Hierarchy Process

3.2 Process Configuration / Creation

The term process configuration which was also part of the title of this thesis could be misleading in the software domain, because it was often used for business processes [GWJ91], [AHW03], [Aal11], etc. This was not the focus we were looking for because we are interested in the configuration of the software development process.

In the software engineering domain two different aspects of processes were considered. First the overall development process described the process procedure, e.g., waterfall, iterative, prototyping. The second aspect was the process itself which described the methods used in the different life-cycle phases and how they are related. This was independent from the development process because it was needed no matter which one is chosen. This was the reason because a software development had a predefined ordering already introduced in Section 2.2. For the further part of this thesis we were only interested in the second aspect, the first one was covered by the Technology Categorization (cf. Chapter 5).

In this thesis process configuration was chosen as a term, because it should cover the process creation in an automatic way. This aspect was rarely covered in the software domain because the companies only used a limited set of technologies in their process and used the overall development process. The decision which technology takes place manually based on their favorite technologies and experience.

There were only a very small number of approaches that try to cover the same problem. For example in [CKO92] the authors focus on a process-driven software development environment. Their paper describes an approach of modeling the input representations used for development of process by transforming it into an output representation for process engineering.

4 Problem Description

The purpose of this chapter is to describe and define the problem, which will be solved in this research. The approach for solving its solution will also be described shortly.

The main findings of this chapter are:

- *Providing a detailed description of the problem we are working*
- *Describes an overview of how the solution to the problem is discovered*
- *Give a brief description of the research goals*

In the first section a detailed description of the problem is given based on the problems identified in the previous chapters. This is done by capturing the findings and problem of the background and related work section and merging them together into one problem. Besides this some questions out of the literature are also addressed belonging to the definition of the problem. Therefore, they are also provided with the solution in this thesis.

The second section gives a brief overview of how the specified problem of the previous section can be addressed. Therefore, this section describes the research approach for getting the solution to the problem.

The last section briefly introduced the research goals based on the previous problem description and the evaluation of the framework.

4.1 Defining the Problem

As already introduced in Chapter 1, the issue of creating the most appropriate process for a software development project before the start, arises due to the following circumstances:

- Technologies are not applicable to the same extent in different projects with different context and impact. And there is no one-size fitting solution for each project available. [FM08]
- Software companies do not invest that much time and money in the software process.
- Technologies delivered from research often lack relevance and the transfer "takes on the order of 15 to 20 years to mature a technology to the point that it can be popularized and disseminated to the technical community at large" [RR85].
- Often companies only introduce "new technologies directly just from friendly or rival companies" [JCF07] .
- Combining technologies is a complex task because of different technology specifications and also the quality of the combination of the different technologies.

From the literature review (cf. Section 3.1) of existing method platforms or similar frameworks we concluded that there is a need of a framework including a method platform which is based on some technologies and their empirical data. Those empirical data consists of the *context* the technology has been applied in and the *impact* the respective technology has in the context. That information is addressed in different technology categorization schemas from which one will be used in the Section 5.1. But the decision-making part of platforms with a general schema is currently not addressed.

From the analysis of existing process configurations or -creations approaches we came up with the knowledge that processes creation is more important in the business area then in the software domain. This is the reason why there are no approaches describing an automatic or semi-automatic approach for configuring a process. Furthermore there is no approach trying to build the software process specific for the different projects and their needs.

Concluding from the analysis of the state-of-the-art, there is no approach (in the literature) available that addresses afore mentioned issues. Summarizing the previous sections, the state-of-the-art with regard to method platforms and semi-automatic process configuring can be summarized as follows:

- Few generic technology schemas are available that provide some good support for selection and combination.
- Different selection strategies are available but most of them are only used in very specific domains.
- Process configuration/creation is more important for business processes than for the software development process.
- Combining technologies of a generic schema for modeling a process is not addressed in the literature.

During this research, we will also address some aspects and questions revealed in existing literature. For example a part of our research is the selection of most appropriate technologies with respect to the context which is relevant for [MRB82] and [Tet90]. This addressed aspect is also taken part of our research because it is needed as a step of the solution approach. We could not integrate their solution in our work because of the usage of a more generic technology model.

4.2 Approaching the Solution

To solve the issues identified in the previous sections, this research provides a framework that helps software companies configuring their process for specific software projects. This framework should deal as an interconnection between research and industry. Today's problem is that "Technology transfer takes on the order of 15 to 20 years to mature a technology to the point that it can be popularized and disseminated to the technical community at large" [RR85]. In addition to this problem the framework integrate new technologies faster and not only technologies "directly from friends or rival companies" [JCF07]. This framework uses all (also new) technologies and provides the best combination of technologies for the specific project to the software company.

The solution partially originates from ideas of the ARAMiS project[8] [KIT11]. We enhance these ideas (e.g. the modeling with SPEM) by extending a method and process platform to an automatic way of configuring the software process for specific projects. This is the case because we wanted to provide a more generic result which can also be used in other projects.

Different research methods have been applied in the course of this research, e.g., literature survey and a case study. Figure 9 summarizes the flow of the research work which will be briefly introduced in the next paragraphs.

In Step 1, we introduce the research. First the background of this work is discussed which is needed to get the understanding for the following steps. To further explore the problem within the state-of-the-art we conducted a literature review. Finally the problem is defined.

[8] The author of this thesis was partially involved in the ARAMiS-project [KIT11] at the Fraunhofer IESE and the Software Engineering Research Group of the University of Kaiserslautern.

In Step 2, the technology categorization schema of [Jed09] is introduced, refined and adapted to the specific need of this research. In addition to the adapted work of [Jed09] there is also an Input model specified. If the information of Chapter 5 is not enough the Appendix B provides XML schemas for most of the models used during this thesis.

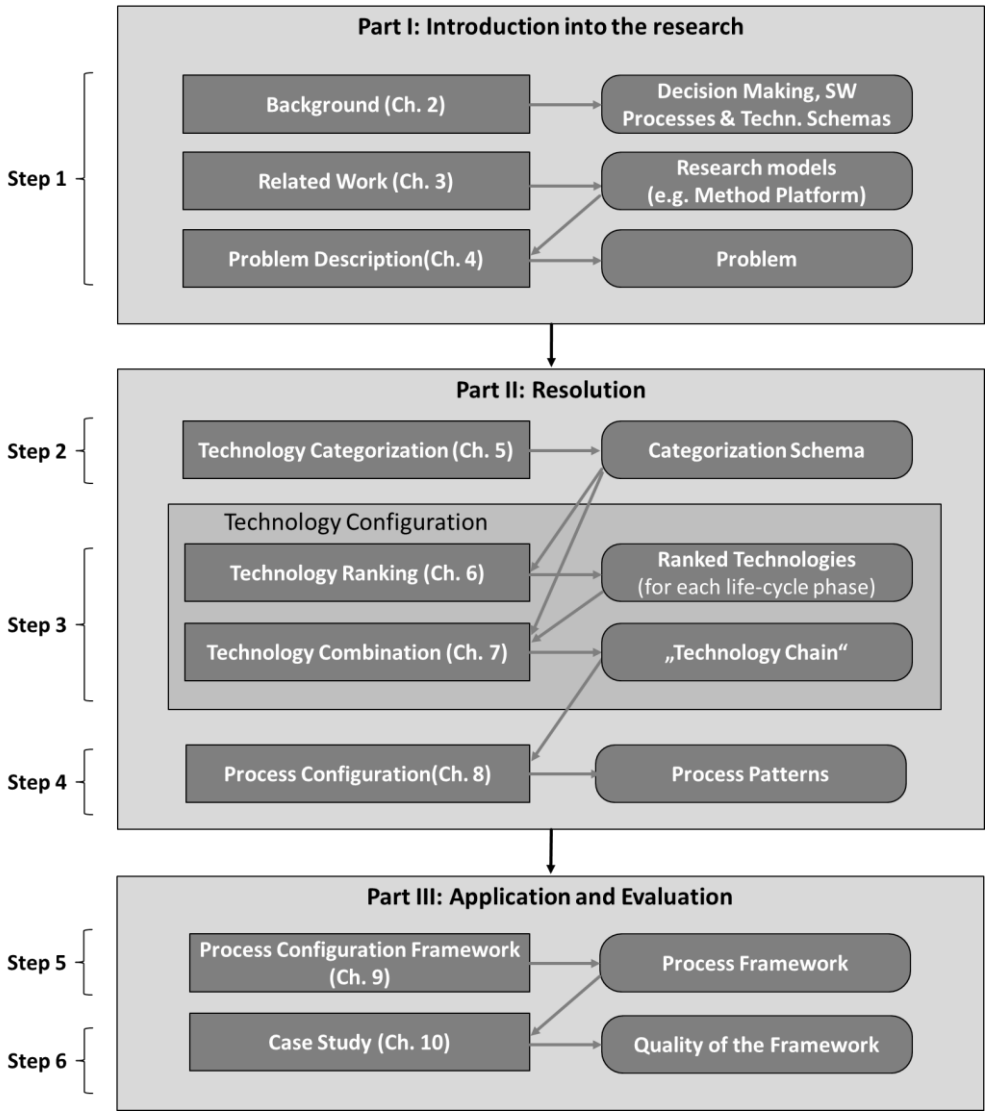

Figure 9. Detailed Workflow of the Research

The next two chapters together build the technology configuration step (Step 3). In this section the technology ranking is the first step by assigning a value to the different technologies based on their matching to the respective project context. Afterwards the technology combination uses the ranked technologies and combines them to all possible Technology Chains. This combination is independent of the project context. It uses the information of the different technologies and the ranking values calculated before. At the end of this step the outcome is the Technology Chain with the highest value.

In Step 4, the process configuration, the output of the technology configuration is used. It transforms the technologies of the chain to process patterns. Those patterns are presented in the SPEM notation which we already introduced in Section 2.2.2. Then it is easy to configure the overall development process by using the patterns.

The first step of Part III develops the two possible approaches for the Process Configuration Framework by cascading the approaches of the previous steps. The phase-based as well as the dynamic approach are explained. Although the static one using the SE phases is more important, because the other one is briefly mentioned and moved to future work.

The last step was conducted to evaluate the Process Configuration Framework of Step 5, which is the fusion of the Steps 2 - 4 (cf. Part II: RESOLUTION). We planned and conducted a case study with experts to verify our research goals and questions.

4.3 Research Goals

Because the problem was already described in a subchapter this section will focus on the research goals. Those are strongly interconnected with the evaluation in Chapter 10.

The generic research goal is the conceptual development of the Process Configuration Framework to solve most of the problems described in the previous chapters. With this framework we intent to simplify the development of software projects by providing a process at the beginning of the work.

For the evaluation, we formulate the following hypothesis: The Process Configuration Framework provides process patterns for modeling a process which is comparable good as a process of an expert. Comparable good has two different meaning. First it could mean that the chains using the same technologies. Second it could mean that they are similar in their Technology Chain value as well as in the top chains, e.g. the Top 5. The resulting configuration is among the alternatives the expert would consider.

Part II: RESOLUTION

In Part I: INTRODUCTION INTO RESEARCH, we provided the underlying concepts and the background which is necessary to understand the following research. Also the state-of-the-art of method platforms was presented by means of a literature review (cf. Section 3.1). The last chapter of the first part specified the research problem.

The purpose of Part II of this thesis is the resolution of the specified research problem. This is done by (1) developing a technology categorization model, (2) specifying how technologies are ranked based on input model , (3) combining the most appropriate technologies from the previous step to find the most appropriate Technology Chain and last (4) configuring a process based on this chain. This Technology Chain is specified in more detail with process pattern based on SPEM [OMG08]. Figure 10 shows the workflow for this resolution part.

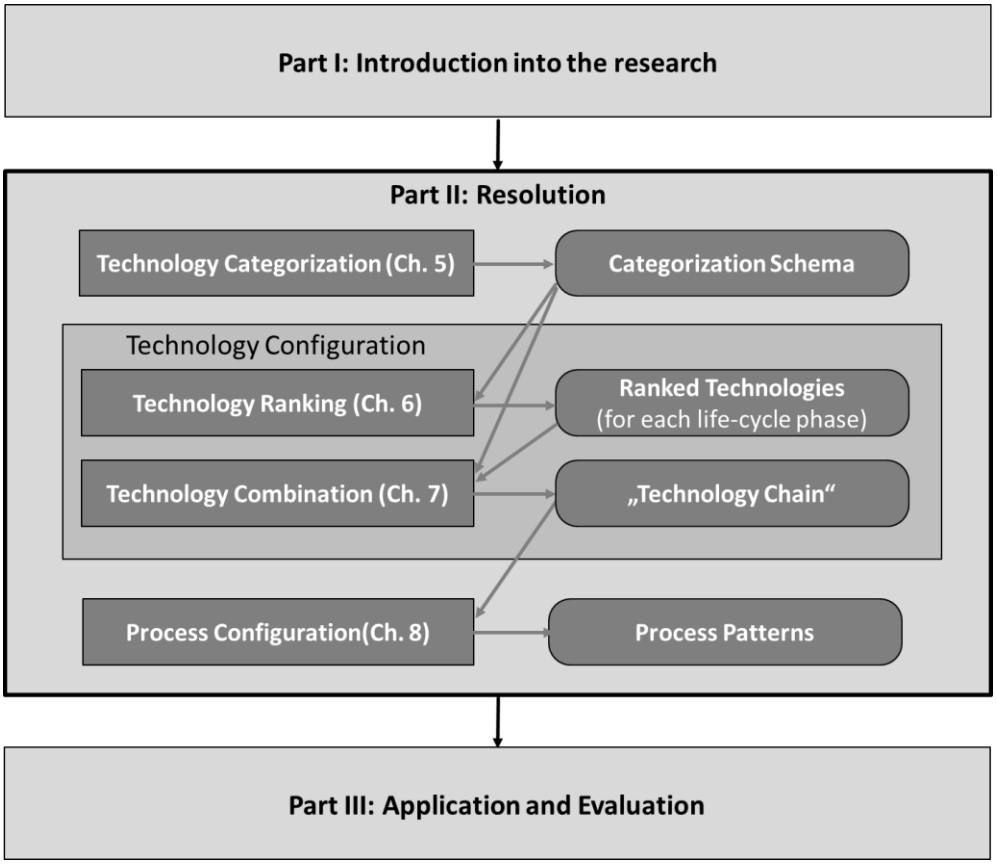

Figure 10. Workflow of Part II: Resolution

In order to work with a general schema, no matter which life-cycle phases the technology is applied in, we first describe and adapt the schema of [Jed09]. Section 5.1 gives a detailed description of the schema, which was briefly mentioned already as part of Section 2.3. This section contains only the needed parts for the following sections and explains their importance.

In Chapter 6 we describe how technologies are ranked based on a given user input. This technology ranking is based on different approaches, which had been introduced in the background chapter. We use a specific ranking system for sorting the different technology based on the matching of the context and the impact. After knowing the quality of the matching attributes the ranking value for each technology is specified using rules.

Using these ranked technologies, we combine the technologies of the different life-cycle phases in Chapter 7. This is one part within a step-by-step approach. First we build all possible Technology Chains. After this we restrict those chains, so that the number of possible chains decreases. At last the remaining chains are then ranked similar to the previous chapter, based on a ranking system and rules.

Chapter 8 of this part deals with the modeling of the technologies of the chain as a process pattern. It generates the output of the framework, the detailed process pattern. These patterns can then be used for configuring the development process.

5 Technology Categorization

The purpose of this chapter is to give a detailed explanation of the categorization schema of [Jed09]*, which is used and adapted for this research. Why this schema is the best in our context has already been explained in Section 2.3. We could not use it in the original way and need to adapt it.*

The main contributions of this chapter are:

- *Understanding of the different models and their relationships*
- *Providing a detailed Specification of the adapted Technology model*
- *Providing a detailed Specification of the reduced Context model*
- *Providing a detailed Specification of the reduced Technology model*

The first section contains a detailed description of the three models of the categorization model of [Jed09]. *These specifications of the different attributes used in this schema are specific customized for our purpose. This means for each important attribute we give a detailed description how to specify this item, to use it in the right way in the later steps, e.g., ranking or combining the technologies or the transformation to SPEM.*

The other section instead specifies the model for the input of the software company/user. This input model contains specifications of attributes of the categorization model and some additional ones. Most of those attributes are used for ranking and combination of the technologies (cf. Chapter 6 and 7). The other attribute are used for the restriction of Technology Chains.

5.1 Categorization Model

The model for categorizing and describing technologies used in this thesis is based on the model presented by Jedlitschka in [Jed09]. The reason for this model is the coverage of all needed attributes for this research work and the explanation of Section 2.3. In contrast to the other schemas, e.g., [Bir], it is more detailed and does not provide that much unnecessary attributes.

We need to give a specification of the different attributes, adapt and change some elements and attributes. In addition some existing attributes need to be refined in detail. Therefore, the next subsections provide a detailed description of the adopted models of [Jed09]. Only the important attributes for this research will be discussed. For all those models we also specified XML Schemas [HKS02] which are attached to this thesis in Appendix B.

The three different models, describing the categorization schema, can also be seen as layers of an overall categorization schema, as depicted in Figure 11. Together those models contain all needed information about technologies. The technology model is the mandatory part that optionally contains references (0..n) to the context or the context and impact model. Only the impact model alone is not possible (1..n), because the impact of the respective technology can only be measured in a specific context.

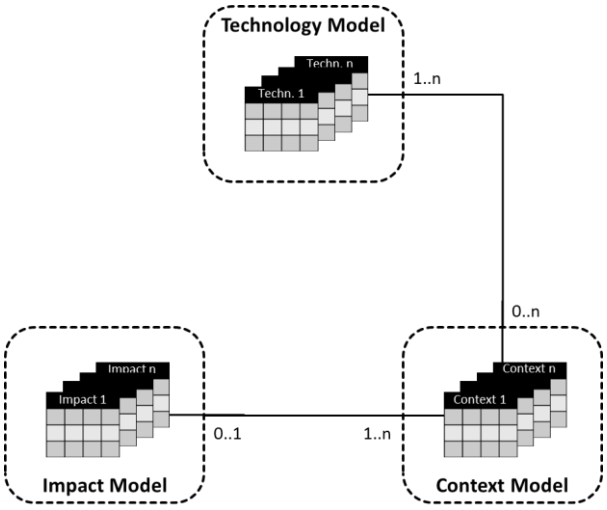

Figure 11. Relation of the three models

5.1.1 Technology Model

The technology model is the main model containing the general content about the technology and the reference to the both other models (cf. Figure 11, the references are in the XML documents, not in Table 3). Table 3 presents the elements and attributes of the extended technology model based on [Jed09]. Besides the new *SPEM model* attribute there is only one major change from the original one of [Jed09]. This is the new *Static Context* element, which contains the attribute of the *Prerequisites* element of the original context model. Those have been transferred to the technology model because we wanted to avoid redundancy of each context model.

Table 3. Model for Technology based on [Jed09]

Element	Attributes	Example	Comment
Name	Name	Checklist-based Reading	What is the name of the technology?
	Abbreviation	CBR	Is there a commonly used abbreviation available? If yes, what is it?
Type	Type	Method	Is it a method (inspections), technology (refactoring), tool (eclipse), p*aradigm (OOSE), technology (.net)?*
Description	Short description	"Inspection method using checklist-based reading to find defects in requirements documents"	Brief description capturing the main elements of the class technology. What is it good for and when will it be applied?
	Long description	"…", or a reference to literature	
	SPEM model	(Graphical Representation of the technology, based on Tasks, Roles, WorkDocuments, Guidances, etc [OMG08])	What are the tasks / roles /documents and guidances (tools, templates, checklists, …)?
	Belongs to family	E.g., CBR belongs to reading techniques	Allows for classification of technologies.
	Complements	E.g., structural testing and functional testing	What are complementary technologies, i.e., which technologies fit best?
	Literature	Literature ("see [XYZ1995], …"), Web pages ("see URL"), …	Documentation, e.g., a reference to a text book describing all the details of the technology
	Background	The technology was developed by …	Where does the technology come from? Who has developed it?
	Alternatives	Perspective-based reading is an alternative approach	Are there any alternatives for the technology available?

Element	Attributes	Example	Comment
Static Context	Qualification required	Inspectors need to understand the format of the requirements documents.	What type of experience is required from the people applying the technology? If experience is not appropriate, the respective costs have to be added or the technology needs to be rejected.
	Training required	Two person days of initial training	What kind of training is required?
	Experience required	Student developers in their fourth year of studies / professional testers with 10 years of experience with XYZ testing.	What kind of people have applied the technology?
	Input (SE object) = Applied on	(Requirements Spec., Architecture Model, Design Model, Code, …)	What kind of documents have to be available before the technology can be applied? What is the required format?
	Output (SE object)	(Requirements Spec., Architecture Model, Design Model, Code, …)	What kind of documents will be available after having applied the technology? What is the format?
Applied in	SE phase	Conceptualization, Analysis, Design, Development, Evolution / Maintenance, Reengineering	In which phase(s) of the development was the technology applied?

The element *Name* only contains two attributes, the *Name* and the *Abbreviation* of the name. Both attributes are specified as plain text. Furthermore, the abbreviation is the short form of the name.

The second element is the *Description* element which contains several attributes describing the different aspects. The important attributes are the *Long-* and *Short description*, *Complements*, *Alternatives* and the *SPEM model*. There is also the *Literature* attribute used as reference if more detailed information is needed. Four of the five attributes are from [Jed09], except the *SPEM model*. The two *Description* attributes are plain text. Instead the *Complements* and *Alternatives* provide references to other technologies. Alternative technologies can be used instead of the described one. The complement references the technologies which fit to the actual one. The *SPEM model* attribute is new with respect to [Jed09]. It is later needed for the specification of the process pattern, which was required for performing the framework. Therefore, we integrated the *SPEM model* into this technology model. This model should describe the workflow graphically (e.g., with an activity diagram [DH01]) by using the different SPEM objects, such as activities, tasks, roles, work documents and guidance. We need to integrate the Method Content as well as process patterns (part of process) into this attribute (cf. Chapter 5 and Appendix B.1).

Another new part of this model is the *Static Context* element, which is the *Prerequisites* element of the context model of [Jed09]. This element was transferred because those parts of the context model are similar for each context of the technology. This should avoid redundancy. The Static Context element includes the requirements attributes such as *Qualification*, *Training*, *Experience* and the *Input* and *Output*. The qualifications are different types which are required from the technology performer. Instead the experience attribute describes the experience needed to apply this technology in the best way, e.g., a professional inspector works better than a novice. The required training for the technology is the kind of training and the needed time. In addition it is important to know whether the technology or a similar training has already been applied. The *Input* and *Output* attribute describe the SE objects needed as input for the technology and providing the output. Both attributes serve as the interface between the technologies of the different phases. Because the *Applied On* attribute of [Jed09] described the input before, it is no longer needed. Examples for such SE objects are Requirements Documentation, Architecture or Design Models, etc.

The *Applied In* element includes only the *SE phase* attribute. This describes the phase of the software engineering life-cycle (cf. Section 2.2.1) in which the technologies are applied.

5.1.2 Context Model

The context model is a dependent model of the technology containing detailed information about the context where the technology was applied in. Table 4 presents the elements and attributes of the reduced context model based on [Jed09]. The only change from the original context model of [Jed09] is the absence of the *Prerequisites* element, which was transferred to the technology model (Table 3 ,*Static Context* element). Therefore, this model only presents the dynamic context which increases the flexibility of the categorization schema.

Table 4. Model for Context based on [Jed09]

Element	Attributes	Example	Comment
Application domain	Industrial sector	Automotive, telecommunication, electronics	In which industrial sector has the technology been applied?
Project	Size	10 person months	What was the size of the project the technology has been applied.
	Kind of software	Embedded system, Information system	What kind of software was developed?
	Type of project	Maintenance project	In what kind of project was the technology used?
Environment	IT environment	Tools, hardware	Is a specific environment required? Is a specific IT infrastructure required?
	Development environment	Programming Language	What kind of development environment is required, e.g., software architecture, or development language?
	Development process	V-Model	What kind of development process is required?
	Paradigm	Object-oriented	What development paradigm is required?
	Interdependency	The technology cannot be used together with …	Are there any interdependencies with other technologies?

The *Application Domain* element specifies the domain the technology has been applied in. This is done by using the *Industrial Sector* attribute which is one specific domain. Some example domains are Automotive, Railway, Avionic, Health Care, Telecommunication, Electronics and many more.

Another context aspect is specified within the *Project* element, containing all needed information about the project context. The detailed information are the *Size* and the *Kind* of project. The size is given in person months. The kind of the project can be chose between Information System and Embedded System. For Example in the automotive domain there are Information Systems, e.g., multimedia systems, as well as Embedded Systems, e.g., airbag systems.

The *Environment* element is a more difficult than the others because it cannot be specified in an easy way of using numeric or binary values. The attributes describing the environment are the *IT environment*, *Development environment*, *Development process*, *Paradigms* and *Interdependency*. The *IT environment* contains a set of tool, hardware, software and other requirements needed for this technology. All these different IT environment types are represented as plain text. The *development environment* is another important aspect which describes which kind of development environment is required. Similar to the IT environment this attribute can contain a set of type, e.g., a given programming language or a given architectural style. The *development process* describes a specific process type the project used, e.g., V-model XT [TUM09]. There is only one process type for each context. The *paradigm* is the most important aspect of the environment attributes, because similar to

the process it is important for the combination. For example object-orientation leads to another architecture, design and implementation than functional. The last environment attribute is the *Interdependency* which provides a set of technologies that cannot be used together with the actual one.

5.1.3 Impact Model

The impact model is a dependent model of the technology and context (cf. Figure 11). It contains detailed information about the impact of the technology in the respective context. Table 5 presents the elements and attributes of the impact model based on [Jed09]. In contrast to the other two models the only change of the original one is the non-usage of the *Return*, *Impact on process outcome*, *Impact on productivity*, *Latency in SE phase*, *Description* and *Reference* elements. These attributes are also not included in Table 5:

Table 5. Model for Impact based on [Jed09]

Element	Attributes	Example	Comment
Impact on the dev. (project) costs	Training	Quantification	What are the costs for training?
	Introduction	Quantification	What are the costs for the introduction of the technology? This includes, for example, consultancy costs and acquisition costs.
	Application	Quantification	What are the costs for the application of the technology (for example, due to effort, infrastructure and licenses)?
	Maintenance	Quantification	What are the costs for the maintenance of the technology?
	Total cost of ownership	Quantification	What are the total costs of ownership? This incorporates all costs related to the technology.
	Project	Quantification	Are there any impacts the technology has on the costs of the project?
Impact on the quality of the product	ISO 9126[9]	Reliability of the product	What is the impact of the technology on certain elements of product quality?
	ISO 9126[9]	Comprehensiveness of a requirements specification document	If necessary, further quality attributes can be defined and the respective impact can be reported.
	Product	Quantification	Are there any impacts the technology has on the quality of the product?
Impact on SE object	SE object	Effort estimation, requirements, architecture, code, process model, …	On which SE object was an impact recognized?
Impact on the development (project) schedule	Introduction	Quantification	What is the schedule for the introduction of the technology?
	Application	Quantification	What is the schedule for the application of the technology?
	Latency time	Quantification	How long does it take until the technology has become routine?
	Project	Quantification	Are there any impacts the technology has on the schedule of the project?
	Time to market	Quantification	What is the impact of the technology regarding time to market?
Latency in SE Phase	In SE phase	Conceptualization, Analysis, Design, Development, Evolution / Maintenance, Reengineering	In which phase(s) will the impact be recognized?

The *Impact on the development costs* element specifies the cost of the respective technology by using several attribute factors. The *Project* attribute defines whether there is an impact on the project costs using a binary value. The *Training*, *Introduction*, *Application* and *Maintenance* attributes specify detail costs as values which are then summed up in the *Total cost of ownership* attribute.

[9] ISO 9126 stands for the list of quality attributes that can be listed here.

The most important impact element is *Impact on the quality of the product*, because quality is one of the most important project success factors [FW06] and key performance indicators [Jed09]. The *Product* attribute works similar to the project attribute of the cost element. The other both *ISO 9126* attributes specify possible product quality characteristics and sub characteristics of [ISO01].

The *Impact on SE object* element only contains the *SE object* as attribute. This attribute contains the objects which were influenced in some way by the actual technology. Possible examples for those are already mentioned in Section 5.1.1.

The *Impact on the development schedule* element deals with the time aspect. The attributes of this element are ordered similar to the cost element. Because the *project* attribute specifies the binary value and the *Introduction*, *Application* and *Latency time* attribute specify detail time as values. All of them were then summed up in the *Time to market* attribute.

5.2 Input Model

The Input model is a mixture of parts of the technology, context and impact model. This is the reason because it describes the (possible) input data needed to perform subsequent steps. The elements and attributes used in the input model are specified in Table 6. Attributes from the previous described models are not further characterized.

Table 6. Model for Input (based on Table 3, Table 4 and Table 5)

Element	Attributes	Example	Comment
Application domain	Industrial sector	Automotive, telecommunication, electronics	In which industrial sector has the technology been applied?
Pre-requisites	Qualification required	Inspectors need to understand the format of the requirements documents.	What type of experience is required from the people applying the technology? If experience is not appropriate, the respective costs have to be added or the technology needs to be rejected.
	Training required	Two person days of initial training	What kind of training is required?
	Experience required	Student developers in their fourth year of studies / professional testers with 10 years of experience with XYZ testing.	What kind of people have applied the technology?
	Existing SE Object	Requirements, architecture, code, …	Which objects of the development process did already exist?
	Preset Technologies	Model Driven Design	What technologies does the Technology Chain have to use?
Project	Size	10 person months	What was the size of the project the technology has been applied.
	Kind of software	Embedded system, Information system	What kind of software was developed?
Environment	IT environment	Tools, hardware	Is a specific environment required? Is a specific IT infrastructure required?
	Development environment	Programming Language	What kind of development environment is required, e.g., software architecture, or development language?
	Development process	V-Model	What kind of development process is required?
	Paradigm	Object-oriented	What development paradigm is required?
Impact on quality of product	ISO 9126[9]	Reliability of the product	What is the impact of the technology on certain elements of product quality?
Impact on dev. costs	Total cost of ownership	Quantification	What are the total costs of ownership? This incorporates all costs related to the technology.
Impact on dev. schedule	Time to market	Quantification	What is the impact of the technology regarding time to market?

The input model of Table 6 only contains two new attribute, the *Existing SE Object* and the *Presetting Technologies* attribute. Although the *Existing SE object* is similar to the SE object attribute of Table 5 and the *Input* and *Output* attribute of Table 3, it will be detailed because of later needs. It describes a SE object, which already exists at the start of the software project, e.g., a problem description. It is really important because it tremendously influences the software development process by defining the starting point of the Technology Chain. In contrast to this the *Preset Technologies* works as another restriction part later on. This is the case because only chains with the specified technologies are used and preset technologies are fixed for their phases.

This model is the base of the first stage of the overall Process Configuration Framework, especially the technology ranking specified in Chapter 6.

We are now discussing the importance of the different attributes of the input model (cf. Table 6). The different attributes are not of the same weight, especially in different software domains they differ. This lead to the conclusion that each company uses its own importance values for the attributes. Because this is not feasible at the introduction of the framework, we provide an initial configuration which can be changed user specific. Later on we also want to integrate a kind of improvement of those initial values with each usage of the framework und change of the values. However, this can be very complex because of possible contrasting values.

Table 7. Initial configuration for the Technology Ranking

Category		Element		Attribute	
Context	66%	Application Domain	25%	Industrial Sector	100%
		Project	25%	Size	55%
				Kind	45%
		Environment	35%	IT Environment	10%
				Development Environment	20%
				Development Process	40%
				Paradigm	30%
		Prerequisites / Static Context	15%	Qualification	45%
				Training	30%
				Experience	25%
Impact	34%	Quality	50%	ISO 9126	100%
		Development Costs	25%	Total Cost	100%
		Development Schedule	25%	Time to Market	100%

The example values given in Table 7 are only initial value, created based on the authors' opinion and ranked-based factor weighting (Ranked-Order Sum [EM07] and Rank-Order Centroid [BB96]). For using those values later on they will be normalized [Pru81] so that all low-level attributes (Table 7, third column) are added to 100%.

The input model can be separated into impact and context on the category level. Under those both categories there are the elements and attributes of the input model (cf. Table 6) needed for the technology ranking. Only the *Presetting technologies*, *Interdependencies* and the *Existing SE objects* are missing because they are only needed for the combination. All those elements and attributes are annotated with importance values. Because those values can be set user specific and the initial values are not based on empirical data, we will not provide an explanation of those example values.

6 Technology Ranking

The purpose of this chapter is the ranking of the individual technologies for each of the software development life-cycle phases based on the input model. Therefore, the output of this chapter is not the best technology for each phase, but rather a ranked list of technologies, in the order of the matching quality to the given input.

The main findings of this chapter are:

- *Description of the ranking process*
- *Description of the rules for performing the ranking*

The first section describes how the rating of the different technologies takes place. Some of the tasks can be performed in parallel for all the technologies and others need to be performed in a predefined sequence.

The other section deals with the prioritization of the technologies by describing the mechanism for the rating. This mechanism is based on rules. All the rules for the different attributes will be explained in detail to get the understanding how the ranking process takes place.

6.1 Ranking Process

Within this subchapter, we detail the process of how the ranking is performed on a conceptual level, before we further detail the rules implementing the ranking in detail. This process has several steps because it can be seen as a step-by-step ranking. These are often used in ranking and selection processes, i.e., in web portals such as IEEE Xplore. Similar to web portals where not all the users information will be requested directly, instead they are requested step-by-step. Because the users input comes from the input model (cf. Section 0) most of its attributes are used for the ranking process.

The different steps used for the ranking are:

1. Ranking based on *Application Domain*, *Project*, *Impact on Quality of Product*, *Impact on Development Costs* and *Impact on Development Schedule*
2. Ranking based on the *Environment*
3. Ranking based on parts of the *Static Context*

The first step takes the information which can be selected or entered by the users directly. Those selection inputs are for the attributes which have a specific range, e.g., the project kind attribute can only be Information or Embedded System. Instead the entered ones are for the attributes with a numeric value, e.g., project size in person months.

During the next step the restriction is performed based on the different attributes of the environment. This step has two sub steps, first the selection of the *single-value attributes* and then of the *multi-value attributes*. Both cases of this step are based on the selection of the possibilities. This means based on the first sub step the possibilities of each attribute are provided and the user selects the supported ones. This is less complex with the single-valued ones, because of fewer possibilities.

The last restriction step is very similar to the previous one because there is again a selection of static context possibilities. However, this time we do not use all attributes, only the training, qualification and experience are important here. For those three attribute the user gets the possibility to select the fulfilled ones.

Based on those three steps of ranking, all technologies are assigned with the ranking values. Now it is possible to sort the technologies based on this values and also for the different SE phases. This is then needed and used in the next stage as input for the creation of the Technology Chains.

6.2 Ranking Value

After the description of the technology ranking process, the rules responsible for this ranking will be explained in detail. The process describes the different steps of it and which attributes are used in them. This section will not provide the same order of rules, because here we only differentiate between context and impact. This is done to use the concept of rule templates provided by [JBo12]. Before defining the rule templates for the different attributes, the ranking value of the technologies are described.

As already mentioned in the process section the technologies are rated with a ranking value. This value is based on an interval scale from 0% up to 100% for each of the technology-context-impact triples. The configuration (initial of Table 7) will be used for the rules with a normalization to stay in the predefined range. The end ranking value defines the quality of the technology match to the given context and impact. The higher the rating value the better the technology matches. The ranking value of all the technologies starts at 0%. This means all of them start at the same level. For each checked attribute the value will be increased if attributes match. The increase of the ranking value is done by maximal adding the predefined importance value (cf. Table 7) to the existing value.

6.2.1 Context Attributes

This section introduces the rules about the attributes belonging to the context part of the input model (cf. Table 6). They are separated in subsections so that we only need to provide a minimum number of rules or rule temples for describing their work.

6.2.1.1 Application Domain, Project & Environment

All the attribute of the application domain, project and environment elements are included into this subchapter because this seven attributes can be merged into only three different rules. Within this we are using rule templates because the *Industrial Sector* and the *Project Size* are grouped together. As well the *Project Kind*, the *Development Process* and the *Paradigm* are merged. Last the *IT-* & *Development environment* result in a rule template.

The first two attribute are combined in one rule template because of their similar function. Both create a similarity value between the users input attribute and the attribute specified in the context model (cf. Section 5.1).

```
rule "C_SimilarityValue_Rule"
  when
          $t : Technology()
          $t.models.length()!=0
          $c : $t.getContext() from $i : $t.models
          $c.get@{element}.get@{attribute}()!=''
  then
          $sim := SimilarityAP(Input.get@{element}.get@{attribute},
                               $c.get@{element}.get@{attribute});
          $i.incRankingValue(
                    $sim*Importance.get@{attribute}Importance());
end
```

Rule 1. Rule for Technology Ranking based on *Industrial Sector* & *Project Size*

This rule (cf. Rule 1) checks in the conditional part, whether there is at least one context (`line 4`) and a value for the respective attributes (`line 6`). For getting the attribute values we need to iterate over the different context model (`line 5`). All those conditions must be fulfilled to fire the action part. This consists of two stages, the calculation of the similarity and the increase of the ranking value. The calculation of the similarity values (`line 7-8`) is different for the two attributes and therefore outsourced into a specific Java method, which deals with the different inputs. This value can only range from 0 to 1 to not exceed the range. The similarity of the numeric values of the *Project Size* can be calculated using difference normalization in the range of 0 and 1:

$$\left(\frac{\text{Context. @\{attribute\}}}{\text{Context. @\{attribute\}} + |\text{Context. @\{attribute\}} - \text{Input. @\{attribute\}}|}\right) x2 - 100$$

In contrast the similarity of the different domains is more difficult, because there is no value for comparison. There are domains which were more similar to a domain then others. Because the complexity of this classification of the similar domains exceeds the scope of this thesis it results in some future work (cf. Section 11.4). The second stage of the action part is the increment of the ranking value (`line 9`). Here the existing value is increased by the similarity value multiplied with the respective importance value.

The three attributes, *Project Kind*, *Development Process* and *Paradigm* can be merged into one rule template because their working is in a similar way. This is the case because there is only a check needed whether the input data is exactly the same to the context data.

```
rule "C_BinaryValue_Rule"
  when
          $t : Technology()
          $c : $t.getContext() from $i : $t.models
          $t.models.length()!=0 &&
          $c.get@{attribute}().isSet@{element}()= true
          $i.get@{attribute}().get@{element}() ==
          $c.get@{attribute}().get@{element}()
  then
          $i.incRankingValue(Importance.get@{attribute}Importance());
end
```

Rule 2. Rule for Technology Ranking based on *Project Kind, Development Process & Paradigm*

The conditional part of Rule 2 replaces the last statement of Rule 1 by using two other checks. First the optional element needs to be available (`line 6`) and then the input attribute value must match the value of the context attribute (`line 7-8`). Therefore, the action part of this rule contains the increment of the actual ranking value by the respective importance value (`line 10`).

The reason for combining the *IT- & Development environment* into one single rule template is that both attributes contain a set of values of their kind and it needs to be checked how many of the context required objects are fulfilled by the users input.

```
rule "C_Collection_Rule"
  when
          $t : Technology()
          $c : $t.getContext() from $i : $t.models
          $t.models.length()!=0 && $l: $it.getObjectArray().length()!=0
          $is: Input.get@{attribute}()
          $at: Object() from $c.getEnvironment.get@{attribute}()
          $at memberOf  $is
  then
          $i.incRankingValue(
                    1/$l*Importance.get@{attribute}Importance());
end
```

Rule 3. Rule for Technology Ranking based on *IT- & Development environment*

In contrast to the previous rules, Rule 3 needs to iterate over a set of values and checking all of them. These results in a more complicated rule template (cf. Rule 3). The first stage of the condition checks whether there is a context model (`line 5`). After this we iterate over the different objects of the context set (`line 7`) and check whether they are member of the input set (`line 8`). Instead the action part works similar to the both other rules. Rule 3 inserts a fraction of the importance value for each fulfilled object of the set (`line 10`).

6.2.1.2 Static Context

All attributes of the *Static Context* (new technology model) element needed for the ranking are included in here because they can either be summarized in one rule or they require a different rule template. The first rule works with the *Training* attribute and the second one is a rule template describing the operation of the *Qualification* and *Experience*.

```
rule "SC_BinaryValue_Rule"
  when
          $t : Technology()
          $c : $t.getStaticContext()
          $c != Null
          ($t.getApplied() == true ||
           $c.getTraining().getKind() == Input.getTrainingKind())
  then
          $t.models.incRankingValue(
                              Importance.getTrainingKindImportance())
end
```

Rule 4. Rule for Technology Ranking based on *Training*

In contrast to the previous rules, Rule 4 is the first rule for a single attribute. Another difference in the condition is the use of the static context element on the technology model (`line 4`) which does not result in iteration as it is the case with the context models. Similar to the other rules this rule also checks for existence (`line 5`). The last stage of the conditional part is about whether the technology has been applied (`line 6`) or the training has been performed already by the user (`line 7`). The action part is similar to Rule 2 and adds the importance with regard to training. Because these attributes are in the technology model, we need to increase the ranking value of all technology-context-impact-triple of this technology.

The *Qualification* and *Experience* attributes of the static context element uses similar functions, which is the reason for combining them into a rule template. Both attributes use a set of values for the attribute, similar to Rule 3. They differ, because they use a similarity value, describing the similarity of the input value with the needed value of the technology.

```
rule "SC_SimilarityCollection_Rule"
  when
          $t : Technology()
          $c : $t.getStaticContext()
          $it: $c.get@{attribute}Array()
          $c != null && $l : ($it.length()!=0)
          $is: @{attribute}() from Input.get@{attribute}()
          $cs: @{attribute}() from $it
  then
          $sim := SimilaritySC($is,$cs);
          $t.models.incRankingValue($sim/$l *
                                Importance.get@{attribute}Importance());
end
```

Rule 5. Rule for Technology Ranking based on *Qualification* & *Experience*

The condition of Rule 5 works similar to Rule 4 but needs to check a set of values. The variable assignment for the set (`line 5`) and the verification whether the set contains elements (`line 6`) are new. Then `lines 7-8` provide the iteration over the different set elements of the input and context model. The action part instead is a mixture of Rule 1 and Rule 3 because we are using a similarity value (`line 10`) and then increase the ranking value (`line 11-12`) by a fraction of the similarity value multiplied with the importance value.

For the similarity of the qualification there is a check whether the qualification of the user matches the needed qualification. This means we only check whether there is the same kind independent of the achievement of this qualification. Examples for such qualifications can be specific certificates or job titles, e.g., Scrum Master or Requirements Engineer.

Instead for the experience is more difficult to get a similarity value. It describes the needed background of the people applying the technology, consisting of two parts. First the distinction between the profession: novice, student, professional or specific professional. Professional means working in the domain of software and specific professional refers to a specific software section, e.g., Architect or Requirements Engineer. Second part is the duration of the experience, describing the time they are working in their profession. For this both parts of the experience attribute the similarity function needs to provide an output. This similarity value is illustrated in Table 8 and will be described in detail. Before starting the explanation of the table the hierarchy of the professions is important:

novice < student < professional < specified professional.

Table 8. Similarity of the Experience

Input Value	Ratio	Context Value	Similarity
profession	>	profession	100%
profession	=	profession	Max. 100% Min. 50%
profession	<	profession	Dist. = 1: 40% Dist. = 2: 20%
novice	<	profession (≠ novice)	0%

The case that the input profession is higher than the one of the context results in the full value, independent of the duration. The case when both, input- and context profession are the same is the most difficult one because the duration needs to be used. If the provided duration in the profession is higher or equal to the needed one the full importance value will be added. In the other case the following function calculates the similarity value:

$$\left(\frac{\text{Context.@\{attribute\}}}{\text{Context.@\{attribute\}} + |\text{Context.@\{attribute\}} - \text{Input.@\{attribute\}}|}\right)$$

The last case is a lower input profession than the needed one (cf. Table 8, row 3 and 4). If this is the case and the input is novice, the value is 0. In the other cases the distance of the professions results in the values presented in Table 8.

6.2.2 Impact Attributes

This section introduces the rules dealing with the attributes of the impact part of the input model (cf. Table 6). They are separated in subsections so that we only need to provide a minimum number of rules or rule temples for describing their work.

6.2.2.1 ISO 9126

Similar to the training attribute of Rule 4 the *ISO 9126* element with its attributes results in another rule which cannot be combined with others. This results from the complexity of the ISO 9126, the different characteristics and their sub characteristics.

```
rule "ISO9126 characteristics"
  when
          $t : Technology()
          $i: $t.getImpact() from $m : $t.models
          $t.models.length() != 0
          $i.getOnProductQuality().getProduct() == true &&
          $l: ($i.getOnProductQuality().getISO9126Array().legth()!=0)
          $in: String() from Input.getISO9126()
          $im: String() from $i.getOnProductQuality().getISO9126Array()
  then
          $sim := SimilarityISO($in,$im)
          $i.incRankingValue($sim/$l *
                              Importance.getISO9126Importance())
end
```

Rule 6. Rule for Technology Ranking based on *ISO 9126*

Rule 6 works similar to some previous rules. It also checks if there are context and impact models (`line 5`), if there is an impact on the product quality (`line 6`) and whether the set of ISO qualities is not empty (`line 7`). In contrast the action part was the more difficult one, because including all possibilities would result in a blowup. We decided to outsource it again in a similarity function (`line 11`). This is the case because we have four different cases (cf. Table 9) which could only merged into more rules with much static checking because of the different (sub-) characteristics of the ISO 9126. Last stage of the action is the increase of the ranking value (`line 12`).

Table 9. Similarity of the ISO 9126 (Sub-) Characteristics

Input Value	Ratio	Context Value	Similarity
Characteristic	=	Characteristic	100%
Sub-Characteristic	=	Sub-Characteristic	100%
Sub-Characteristic	≠	Sub-Characteristic	0%
Characteristic	∋	Sub-Characteristic	100/#subchar %
Sub-Characteristic	∈	Characteristic	100%

The values of Table 9 can be explained very easy, because there are two general cases. The first case covers the similar levels of input and context value (Table 9, row 1 - 3). The second case deals with different levels of values (Table 9, row 4 - 5). In this case we need to check if the lower sub characteristic is element of the upper one. If the user input is a top element then similarity is a fraction, because the technologies only support parts of it. In the other direction we add the full value because if a technology only has a characteristic this implies also all sub characteristics.

6.2.2.2 Development Costs & - Schedule

The development costs and schedule are included into this subchapter because they can be merged into only one single rule. This is the case because both elements have similar attributes in the input model (*TimeToMarket* and *TotalCosts*) and behave similar. They contain a *Project* attribute which is a Boolean value describing if there is any impact. The other attribute gives the numeric value for the respective impact.

```
rule "I_SimilarityValue_Rule"
  when
          $t : Technology
          $im: $t.getImpact() from $i : $t.models
          $t.models.length() != 0
          $im.get@{element}().getProject() == true
  then
          $sim := SimilarityCS(Input.get@{element}.get@{attribute},
                               $c.get@{element}.get@{attribute})
          $i.incRankingValue($sim *
                          Importance.@{element}@{attribute}Importance())
end
```

Rule 7. Rule for Technology Ranking based on *Development Costs* & - *Schedule*

Rule 7 checks whether there are existing impact models, similar to previous rules (`line 5`). In addition the conditional part also checks whether there is an impact on the element (`line 6`). After this the action part is similar to the project size attribute specified in Rule 1. Because the attribute are also using numeric values the same similarity function of the *Project Size* attribute can be used (`line 8-9`). After this the ranking value is increased by the importance value (`line 10-11`).

6.3 Chapter Summary

The purpose of this chapter was to describe the ranking of the individual technologies, formalized by means of the implemented rules for performing the selection. Throughout this chapter, we first described the process of the step-by-step ranking. Then, we presented the implemented rules which perform the ranking of the technologies in detail. Those are divided into context and impact rules.

7 Technology Combination

The purpose of this chapter is to describe the combination of technologies. This is the next step of the framework after the ranking values are assigned to all technologies. With the lists of technologies for the different phases the combination starts and tries to find a best possible technology combination, called "Technology Chain".

The main findings of this chapter are:

- *Description of the combination process*
- *Introduction of the Technology Chain concept*
- *Extension of the Technology Chain by Quality Assurance concepts*
- *Explanation how to restrict and rank Technology Chains*
- *Description of the restriction and ranking rules*

The first section is about the overall process of combining the technologies to get a Technology Chain. There is a detailed description of the process, which specifies the steps needed to be performed to get the right outcome.

The second section of this chapter includes all the information about the chaining step. There all possible Technology Chains are built based on the technologies sorted by their SE phase.

The next two sections contain the details about the restriction and ranking of the Technology Chains. It contains the restriction of the chains by some attributes as well as the ranking by other attributes. The rules implemented for this will be explained in detail in the third and fourth section.

The last section of this chapter contains information about the extension of the Technology Chain concept by quality assurance technologies. It deals with the possibility of plugging verification technologies between the SE phases of the chain.

7.1 Combination Process

Within this section of the Technology Combination chapter the process of how the technologies are combined is described in detail. This is important to get the understanding of the overall combination process. It comes in several steps, which are listed below. The restriction as well as the ranking described in this chapter is performed in a similar way as the technology ranking (cf. Chapter 6).

The combination process consists of five steps:

1. Creation of all possible *Technology Chains*
2. Restriction based on *Existing SE Objects*, *Preset Technologies* and *Interdependencies*
3. Ranking based on *Static Context* (Interface [*Input / Output*] & *Complement*)
4. Ranking based on *Dynamic Context* (*Paradigm* & *Development Process*)
5. Extending the Technology Chain with *Quality Assurance Technologies*

In the first step all possible Technology Chains are created. This is done based on the information concerning the SE phases assigned to the technologies by the *SE phase* attribute. Each technology of one phase is combined with all of the next phase. For example all technologies of the specification phase are combined with all architecture technologies. From a pure calculation point of view, the obviously huge number of Technology Chains is not an issue. But many combinations might not be meaningful. This is further addressed in the next step, i.e., the restriction phase.

The restriction phase uses three attribute for reducing the number of chains. The *Preset Technologies* of the input model (cf. Table 6) restrict the Technology Chains by using only those chains that contain the specified technologies. This reduces the number tremendously. The next restriction attribute is the *interdependencies* of the technology model (cf. Table 3). Therefore, we check for all the technologies in a chain whether there are negative interdependencies. If this is the case the respective Technology Chain is deleted. The last restriction step is based on the *Existing SE Objects*, e.g., an existing Requirements Specification. In this case we reduce the chains by using the existing object.

The following two steps concern the ranking and will be described together. The first ranking is performed based on the *Static Context* element of Table 3. The main part of this static context is the ranking based on the interface matching. In addition the static context also contains the *Complement* attribute. The dynamic aspect specified in the context model (cf. Table 4) then also increases the combination value if the technologies in one chain use the same *Paradigm* and/or *Development process*. The outputs of this step are the chains of the previous step with a value. This is a calculation based on the combination value and the ranking values for each technology (cf. Table 10).

In the last step it is possible to extend the standard Technology Chain with quality assurance technologies, especially by an appropriate verification technology for each SE phase.

7.2 Creation of all Technology Chains

Before starting with the explanation of the Chaining Step the concepts of a Technology Chain will be introduced. A Technology Chain is a set of technologies which used only one technology from each SE phase (cf. Section 2.2.1). Therefore, the standard Technology Chain includes five phase: *Specification*, *Architecture*, *Design*, *Implementation* and *Validation*. Because there is only one quality assurance (QA) part in the chain, the *Validation*, we provide an extended Technology Chain (cf. Section 7.5).

The chaining step itself uses all provided technologies to create all possible Technology Chains. Therefore, the technologies were used which are already sorted based on their SE phases in the technology ranking. All the possible Technology Chains are then created by combining each technology of phase X with all technologies of X+1. For example Figure 12 depicts that each of the specification technologies is combined with all architecture technologies. This is analogously done for all the other phases as Figure 12 provides an example.

7.2.1 Results of Chaining Step

The result of the Chaining Step which is the first step of the technology combination results in all possible Technology Chains as it is illustrated in Figure 12:

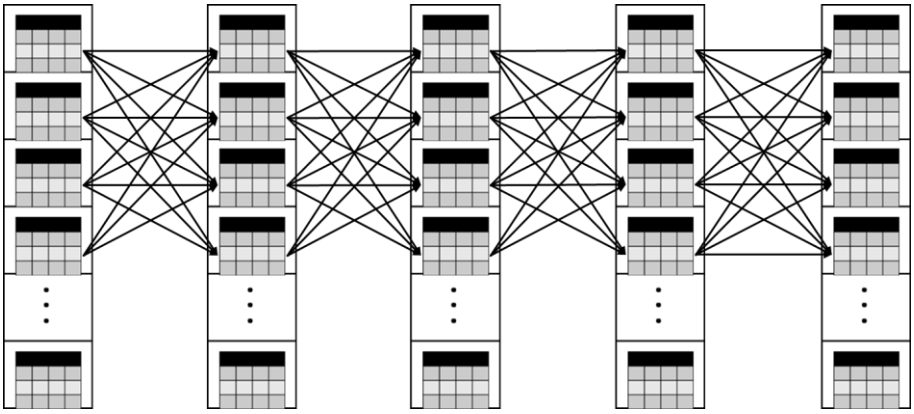

Figure 12. Result of the Chaining Step

7.3 Technology Chain Restriction

After the description of the general process of combing the technologies and providing details of the Chaining Step in the previous section, we detail the rules for the restriction of the Technology Chain. The rules for the restrictions are formalized by Drools, a business rule management framework [JBo12]. This step of the Technology Combination uses the result of the previous step as its input. This means we restrict on all the created chains of the Chaining Step.

The Technology Chain restriction is performed in three steps, whereas the order of the steps is of no importance, considering:

- Restriction based on *Preset Technologies*
- Restriction based on *Interdependencies*
- Restriction based on *Existing SE Objects*

7.3.1 Preset Technologies

In some software projects it is possible, that there are technologies required by some stakeholder, e.g., customer, government. Therefore, the *Preset Technologies* attribute is specified in the input model and can include a set of fix technologies. All other Technology Chains not using these technologies are deleted.

```
rule "Preset Technologies"
  when
          $tc: TechnologyChain()
          $it: Technology() from Input.PresetTechnologies
          ($tc.getSpecification().getName()   != $it.getName() &&
           $tc.getArchitecture().getName()    != $it.getName() &&
           $tc.getDesign().getName()          != $it.getName() &&
           $tc.getImplementation().getName()  != $it.getName() &&
           $tc.getValidation().getName()      != $it.getName())
  then
          $tc.delete();
end
```

Rule 8. Rule for Technology Chain Restriction based on *Preset Technologies*

Rule 8 specifies the checking of presetting technologies. In this condition it is checked for all input technologies (`line 4`) whether there is no matching technology in the phases of the chain (`line 5-9`). The action part deletes the Technology Chain in case there is no match (`line 11`).

7.3.2 Interdependencies

The *Interdependency* attribute which is part of the environment element of the context model is very important for the technology combination. This is the case because it also restricts the chains if there are two or more technologies which cannot be combined. Each technology can have such interdependent technologies.

```
rule "Interdependencies"
  when
          $tc: TechnologyChain()
          $ts: $tc.get@{phase}()
          $c : $ts.getContext() from $ts.models
          $t : String() from
               $c.getInterdependencies().getTechnologyModelArray()
          ($tc.getSpecification().getName()   == $t ||
           $tc.getArchitecture().getName()    == $t ||
           $tc.getDesign().getName()          == $t ||
           $tc.getImplementation().getName()  == $t ||
           $tc.getValidation().getName()      == $t)
  then
          $tc.delete();
end
```

Rule 9. Rule for Technology Chain Restriction based on *Interdependent Technologies*

Rule 9 implements the deletion of chains using interdependent technologies. We are using a rule template again in this case because we would need one rule for each phase. The conditional part consists of two stages, first the navigation to the interdependency attribute (`line 3-7`) and then the check. For all the interdependencies of a technology we check whether independent technologies are part of the chain (`line 8-12`). If this is the case the action part fires and the chain is deleted (`line 14`), similar to Rule 8.

7.3.3 Existing SE Objects

The starting point of the Technology Chains is defined by checking whether there already exists a SE object which can be used. In contrast to the both other restriction possibilities this restriction does not delete Technology Chains but it reduces the number of phases and technologies. In this case the chain is filled with some “empty” technologies at the beginning. For this purpose the following Rule 10 was developed.

```
rule "Existing SE Object"
  when
          $tc: TechnologyChain()
          $i : Input (ExistingSEObject != null)
  then
          $tc.restrictTechnologyChain($i.getExistingSEObject());
end
```

Rule 10. Rule for Technology Chain Restriction based on *Existing SE Objects*

This rule only contains one real condition which checks whether there exists an SE object (`line 4`) or not. In the positive case the action starts by using a Java method of the Technology Chain object, which takes the SE object and reduces the chain. This is done by first checking the document type of the object and then the phase it belongs to. After this is established the technologies and their in-/outputs before the SE objects are deleted and the SE object is inserted to the outputs.

7.3.4 Results of Restriction Step

The result of the Restriction Step is a smaller set of possible Technology Chains. An example of such a decreased set of technologies is depicted in Figure 13. However, Figure 13 only includes a *Preset Technology* (Figure 13, third technology in third list) as well as an *Existing SE Object* (Figure 13, Requirements Specification). Interdependent technologies are not possible to illustrate in a figure:

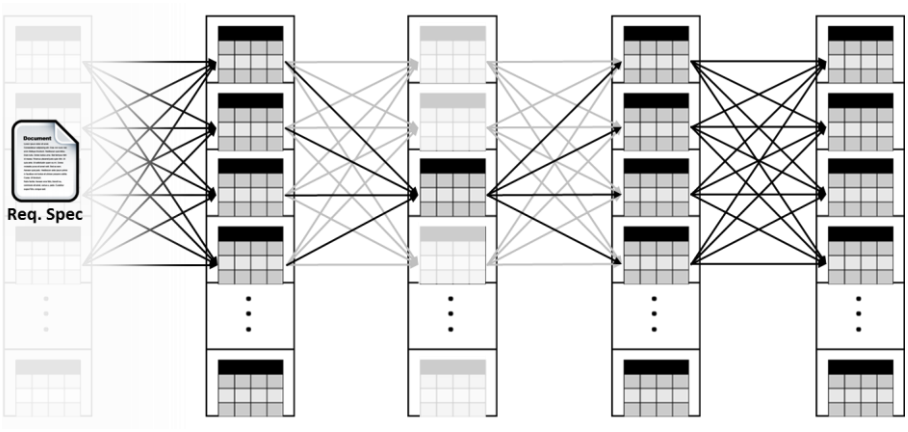

Figure 13. Result of the Restriction Step

7.4 Technology Chain Ranking

Now the third step is the ranking of the Technology Chains based on their static (*Static Context* element of Table 3) and dynamic context (cf. Table 4). This ranking is described in the same order as it was introduced in the process section above. Similar to all previous rules we are using Drools [JBo12] for defining the following ranking rules. The Ranking Step uses the remaining Technology Chains of the previous step as its input.

The Technology Chain ranking is performed in two steps, considering:

1. Ranking based on Static Context
2. Ranking based on Dynamic Context

Before defining the rules for the different attributes, information about the ranking values of the Technology Chains is needed.

The final decision criterion for the most appropriate Technology Chain is the *TechnologyChainValue*. This is composed of the overall combination value and the ranking values of the technologies in the chain. Similar to the ranking of the single technologies all importance values of this chapter can also be adapted by the user for specific needs using the configuration (cf. Table 10). Some initial values are needed for using the framework. Therefore, the combination as well as the ranking of the technologies will be treated with the same importance.

Table 10. Initial configuration for the Technology Chain Ranking

Category		Element/Attribute	
Combination value	50%	Interface (Input/Output)	65%
		Complement	20%
		Paradigm	10%
		Development Process	5%
Ranking values (of the technologies)	50%	Specification Technology	20%
		Architecture Technology	20%
		Design Technology	20%
		Implementation Technology	20%
		Validation Technology	20%

Similar to the technology ranking the decision which Technology Chain to select is performed based on ranking the Technology Chains. The value for the combination is called *combination value* and stored for each chain. Similar to all other values this one is also normalized [Pru81] from 0% to 100%. It defines the quality of the matching of the different technologies, by increasing the value for matching attributes. This means the higher the combination value the better the Technology Chain.

The combination value is made up of four different attributes that will be later defined in rules: *Interface*, *Complement*, *Paradigm* and *Development Process* (cf. Table 10). The interface matching between the inputs and outputs of the different technologies is the main part, because if they do not fit an extra transformation is needed. The complements of technologies of the chain are also used for increasing the value, because it contains good predefined matching technologies. The last two from the dynamic context are the *Paradigm* and *Development Process* which are less important. There we check how many of the paradigm or process are similar. Because of this importance ordering (Ranked-Order Sum [EM07] and Rank-Order Centroid [BB96]) and our opinion we defined the values of Table 10. Although the framework provides the possibility of user specific adaption of the values, similar to the configuration of the technology ranking (cf. Table 7).

With the combination value and the highest ranking values of the different technologies it is possible to calculate the overall *TechnologyChainValue*. We take the highest ranking value of each technology because it is the one which matches best to the project needs. The *TechnologyChainValue* is used for the selection of the output, which is the chain with the highest Technology Chain value.

7.4.1 Static Context

This section introduces the rules for ranking the different Technology Chains. Thereby we use the attributes belonging to the *Static Context* of the technology models (cf. Table 3). They are separated in subsections which provide the details of the rules used for those two static aspects.

7.4.1.1 Interface Matching

This part of the technology combination, especially of the Technology Chain ranking, is the most important one, as Table 10 shows. Therefore, we try to focus our work especially on this matching part.

The interfaces used for this matching and ranking are the ports of one technology to others. Each technology has two ports, an incoming and an outgoing one. They are described by the *Input* and *Output* attribute of the *Static Context* element (cf. Table 3), which provide/use one or more *SE object*. Although those objects were already described in Section 5.1.1, more information about them is needed for the following matching and subsequent ranking. An SE object includes three sub specifications: document type, format and description. The document type can be a Requirements

Specification, Architecture Model, Design Model, Source Code or Test Cases. And each of those types has different formats described during the explanation of the next rule.

Because the document type and the format of such objects are mandatory, the following rule describing the interface matching will use those. The differences of the varying formats of the document types will be explained during the rule introduction. For explaining how the matching and the subsequent ranking takes place only one rule is needed. This is the case because the process is the same, no matter which phase or object is observed at the moment. Because they only differ in the various document formats, this is outsourced in a method for simplicity reasons.

```
rule "Interface Match Ranking"
  when
          $tc: TechnologyChain()
          $i from $is: $tc.getInputs()
          $i.getDocumentType() memberOf $tc.getOutputDocumentTypes
          $o from $os: $tc.Outputs(DocumentType == $i.DocumentType)
  then
          $sim := SimilarityInterface($i,$o);
          if ($sim <= 1) {
            $tc.addTranformation($i,$o);
            $tc.addOutput($i);
          }
          $tc.incCombinationValue($sim/$is.length() *
                                  Importance.getInterfaceImportance());
end
```

Rule 11. Rule for Technology Chain Ranking based on *I/O-Interface Matching*

Rule 11 is the most complex rule of the whole thesis, because of two nested iterations and a check in one of them. The conditional part starts by iterating over the set of inputs (`line 4`). This set contains the inputs of all technologies of a chain. Then we check whether the document type of the input object is part of the set of all document types of the outputs (`line 5`). After this the second iteration takes place, walking thought all the outputs with the same document type (`line 6`). Instead the action part contains three stages. Those stages only differ in one aspect from most of the previous rules, the usage of the similarity value. The first stage is the creation of the similarity value based on the input and output (`line 8`), which will be further refined in later paragraphs. Then there is a check whether the input and output match exactly (`line 9`). If this is not the case a transformation between the two formats of a SE object is needed (`line 10`). In addition to that, the new output of the transformation is stored in the output set (`line 11`). Then the last action increases the combination value based on the number of inputs, the similarity between in- and outputs and the importance (`line 13-14`).

The similarity function used in Rule 11 is the most complex part because it needs to specify the quality of the matching of two SE objects. As already briefly introduced above we are only using this function for two SE objects of the same document type (`line 6`). Therefore, the following explanation of this function is based on the different formats (sometimes also kind of granularity) of the SE objects. There is the possibility of not specifying a format in input documents. This means that all formats are possible.

Table 11. Similarity values of the different Requirement formats

Format 1	Format 2	Similarity
plain text	plain text	100%
plain text	structured text	50%
plain text	diagram	30%
structured text	structured text	100%
structured text	diagram	80%
diagram	diagram	100%

The requirement specification object mainly includes aspects about the data as well as functional and non-functional requirements. For those requirements exist several styles, which differ e.g., in their notation [Lau02]. Most important for our similarity function are the different notations specified by [Lau02]: *diagrams* (e.g. context-, use case-, activity-, and dataflow diagrams), *plain text* (e.g. task descriptions) and *structured text* (e.g. tables). This differentiation also includes another common requirements separation, between informal (≅ plain text) and formal documents (≅ structured text) [FKV91]. Those three formats differ in their similarity as depicted in Table 11. Three of the six possible cases treat the same formats (Table 11, row 1, 4 and 6). In this case the value is assigned by 100%. Within the other cases, diagram is very similar to structured text, because they could be easily mapped to each other. Both text formats behave similar, because they both use different text formats. Only the mapping between plain text and diagram is very difficult.

As already mentioned in Section 2.2.1, architecture and design are very similar. Therefore, both SE phases will be treated together in this paragraph. In those phases many different stakeholders are involved which require different notations for their viewpoint. This results in often using several views for complexity reduction, which represent a single, well-defined perspective. Unfortunately there is no general set of views that adequately describes all possible architectures of each existing software system. Therefore, the views need to be selected individually for each project. Often the 4+1-view model of [Kru95] is used. As a consequence the formats of the architecture and design model used in this research are a set of different views. The specification of the similarity value between two architecture or design objects is easy. This is done by comparing the needed views with the provided views.

Not only the architecture and the design use similar formats. This is also the case for the implementation and validation phase, even if they use different document types. The type of the implementation is source code, instead the validation has the test cases. The different formats of both can easily be defined by specifying the programming language [Veg02] used in the SE phase. This results in a specific SE object format for each possible programming language. Of course the same programming language results in a similarity of 100%. But if they do not match, a similarity value can be assigned by checking the programming paradigms. There are four main paradigms, *object-oriented*, *imperative*, *functional* and *logical* programming [Nør10]. In addition [KLM97] extended those by the *aspect-oriented* programming. There are also some more paradigms and for some of the paradigms (imperative & declarative) there are also refinements. According to whether the formats of both objects are in the same paradigm (50%) or sub-paradigm (70%) a different similarity value is assigned.

7.4.1.2 Complement

As defined in Section 5.1 the *Complement* attribute of the technology model includes references to good cooperating technologies. Therefore, we need to check whether the reference in this attribute matches with one of the other technologies in the chain. If such complementary technologies are existing the combination value is increased.

```
rule "Complement Ranking"
  when
          $tc: TechnologyChain()
          $a : $tc.get@{phase}().getDescription()
               .getComplements().getTechnologyModelArray() != 0
          $t : String() from $a
          ($tc.getSpecification().getName()  == $t ||
           $tc.getArchitecture().getName()   == $t ||
           $tc.getDesign().getName()         == $t ||
           $tc.getImplementation().getName() == $t ||
           $tc.getValidation().getName()     == $t)
  then
          $tc.incCombinationValue(1/$a.length() *
                  (Importance.getComplementImportance()/5));
end
```

Rule 12. Rule for Technology Chain Ranking based on *Complementary Technologies*

The rules responsible for the complement are merged, in the rule template Rule 12, because without the merging, a rule for each SE phase would be needed. The first part of the condition checks whether the attribute is not empty (`line 4-5`). In this case the combination value is not increased. In the other case the objects of the complement set (`line 6`) are checked, whether they match with a technology of the chain (`line 7-11`). If one technology matches, the action part takes place. Within this the combination value is increased by a fraction of the importance value (cf. Table 10). The size of the fraction depends on the number of complements per phase (`1/$a.length`) as well as the number of phases.

7.4.2 Dynamic Context

This section introduces the rules about the dynamic context attributes belonging to the context model (cf. Table 4). In addition to the static context we are now extending the chain ranking with two dynamic attributes. They are called dynamic because they are part of the context models and they differ depending on the project context. In the following we are considering the context-impact-triple with the highest ranking value.

The *Paradigm* and *Development Process* attribute of the Environment element can be merged into one single rule template because of their similar behavior. Both attributes are stored as string and only provide one value. Therefore, the goal is to check whether the technologies of the chain work in the same environment, e.g., with the same paradigm or in the same development process.

```
rule "DynamicContext Ranking"
  when
          $tc: TechnologyChain($n : @{attribute}.length() != 0)
  then
          for (int i=0;i<=$p;i++) {
             for (int j=0;j<i;j++) {
                if ($tc. @{attribute}s[i] == $tc. @{attribute}s[j])
                {
                    $tc. @{attribute}s.remove(j);
                    i--;
                }
             }
          }
          $tc.incCombinationValue((5-$n)/4 *
                    Importance.get@{attribute}Importance());
end
```

Rule 13. Rule for Technology Chain Ranking based on *Paradigm* & *Development Process*

The condition of Rule 13 works with Technology Chains with a non-empty set of the attributes (`line 3`). In contrast to the other rules this one has a more complex action part. The rule works on the set of the attribute, which is filled with the values of the best matching context of each technology. Based on this set the rule checks how many different values appear in the chain (`line 5-12`). This number can range from one (all use the same) to five (all use different) paradigms. In the last step of the action the combination value is increased by a fraction of the importance value (`line 13-14`). The size of the fraction depends on the number of different paradigms or development processes. A low number results in a high value and vice versa.

7.4.3 Results of Ranking Step

The result of the Ranking Step is the Technology Chain with the highest *TechnologyChainValue*. Therefore, we need to calculate the Technology Chain value for all the chains remaining from the Technology Restriction. This value is calculated based on the combination value of the technology combination and the ranking values of the technology ranking. How this calculation is done depends on the configuration of the framework which is initially set to the values of Table 10.

After the calculation of this value for all remaining chains, the one with the highest *TechnologyChainValue* is used. Such ranking result is presented in Figure 14:

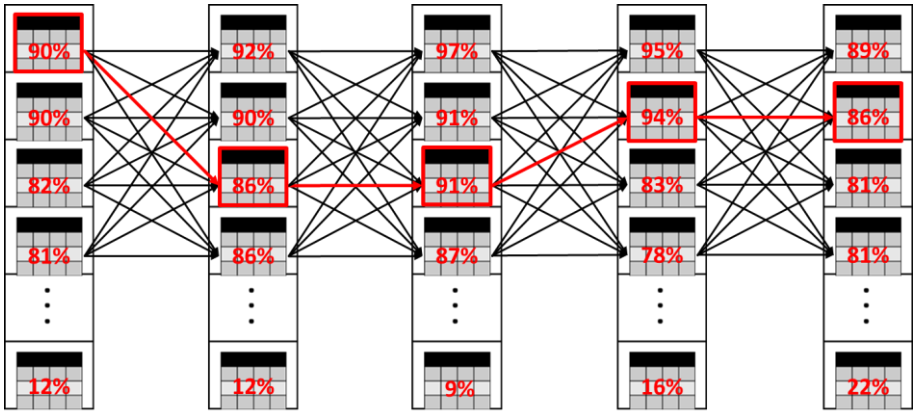

Figure 14. Result of the Ranking Step

7.5 Technology Chain Extension by Quality Assurance

Within this subchapter the extension of the standard Technology Chain is described. The purpose of this step is to provide a possibility of extending the standard Technology Chains by quality assurance (QA), i.e., verification technologies. To perform this step the output of the ranking is used which is a standard Technology Chain.

The three steps used for extending the Technology Chain by quality assurance are:

1. Using *Technology Chain* with highest *TechnologyChainValue*
2. Ranking of *Verification Technologies* for the specific project
3. Extending the Technology Chain with *Quality Assurance* based on combination and ranking value

The first step of the extension is only selecting the Technology Chain with the highest Technology Chain value. Therefore, one of the ranked Technology Chains of Section 7.4 is selected.

During the next two steps a verification technology per SE phases (except the validation) is added to the chain. In general this is done by using the ranking value of the technologies and a combination value of the verification technologies with the phase technology. The assignment of the ranking value for each verification technology works similar to the phases of a standard chain explained in Chapter 6. Then the combination of the verification technologies with the chain takes place on those values. The verification technologies always belong to an SE phases, although their phase attribute in the template (cf. Table 3) is *Verification*. Another characteristic of them is exact similarity of input and output. This is the case because the all those technologies verify a specific SE object and then correct/improve it based on e.g., a defect list. This similar input and output gives the possibility to plug the verification behind the technology of the different phases.

The technology chosen for the phases is selected based on the combination- and ranking value. The combination value is more important in this case because a transformation between a technology and its verification does not make that much sense. The combination value of the verification technologies uses an adaption of the interface matching of Rule 11.

7.5.1 Results of Technology Chain Extension Step

The result of the Technology Chain Extension Step is a Technology Chain which is extended by a Verification technology for each of the SE phases of the standard chain. This is presented in Figure 15 which provides a graphical description of both kinds of Technology Chains, standard and extended. There are two different categories for those phases, development (Figure 15, white boxes) and QA phases (Figure 15, grey boxes):

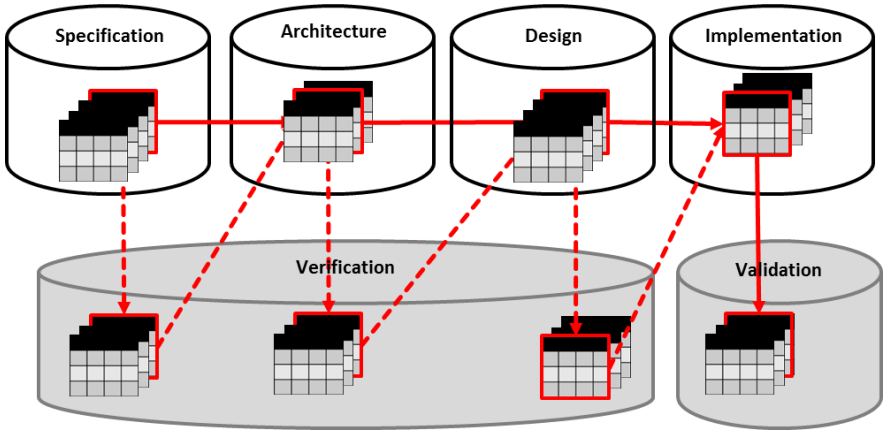

Figure 15. Result of the Technology Chain Extension Step

7.6 Chapter Summary

The purpose of this chapter was to describe the combination of the ranked technologies, formalized by means of the implemented rules for performing the different steps that are included in the combination. Throughout this chapter, we first created all possible Technology Chains by combination. Then, we presented restrictions of the chains, which were followed by the ranking of the remaining chains. Those steps were also implemented with rules. The last step was the quality assurance extension of the Technology Chain by use of verification technologies.

8 Process Configuration in SPEM

The purpose of this chapter is configuring the process in SPEM based on the Technology Chains. The prerequisite of this step is the selection of the most appropriate Technology Chain, which is the output of the previous Chapter 7.

The main findings of this chapter are:

- *Description of the Process Component Diagram and needed extensions*
- *Instantiation of the Process Component Diagram with technologies*
- *Explanation of the Transformation of Technology Chain to process patterns*

The first section of this chapter gives an overview of how the process configuration takes place and which transformations and other activities are needed for this. Most of the things mentioned in this section will be further refined later.

After the process, the next two sections deal with the Process Component Diagram of SPEM. First, there is developed a general one for the life-cycle of Section 2.2.1. This is then followed by the explanation of possible extensions needed later on.

The fourth section uses the Process Component Diagram and instantiates the components by the respective technologies of the chain.

The last section contains details of how the process patterns for the specific process are provided. At this point there is also some user involvement because the user has then the possibility to configure her process based on the provided patterns.

8.1 General Configuration Process

The process of configuring the development process of a software project is the last step needed for the overall framework. This is built upon the other steps, because it needs a Technology Chain as input.

Similar to the previous chapters we use the following steps for configuring the process in SPEM:

1. Using the generic *Process Component Diagram (PCD)*
2. Extending the *PCD* by new components
3. Instantiating the *PCD* with a *Technology Chain*
4. Defining the corresponding *process patterns*

The first two steps of this configuration are generic ones which are almost similar in each project and do not dependent on anything specific project. It is the generic *Process Component Diagram* [OMG08] (cf. Section 8.2). This models the software engineering life-cycle, introduced in Section 2.2.1, and the document flow of the different phases. This is shown by using ports which provide the Input and Output documents. In some cases this generic diagram needs to be extended by other components to fulfill the requirements for the later steps. These extensions are a transformation between technologies or the extension of the chain by QA.

Then the PCD is instantiated by the Technology Chain, no matter which kind of chain. Within this step the technologies of the chain are used for instantiating the respective process components.

The last step of the process configuration provides the *process patterns*. The details of providing those patterns and not the whole process will be explained in Section 8.5. A process pattern for each technology will be offered. Those can then be plugged together to build an individual process for each software project.

8.2 Process Component Diagram

A part of SPEM is the process components which are mainly used in the Process Component Diagram (PCD) [OMG08]. This diagram contains the different process components with their ports for input and output. A process is described by plugging the ports together and building a kind of product flow from the start to the end via different components.

In our research we use the PCD for modeling the generic software life-cycle explained in Section 2.2.1. The sequence of the different components the diagram is predefined because one phase mainly needs outputs of other phases as input. This is independent of the different development process, e.g., iterative process model [FM99]. How the framework will cover the problem of different development processes will be explained in a Section 8.5.

Sometimes it is the case that some phases use more than one outcome of the predecessor. This is covered by the PCD, although it is not visible in Figure 16. We assume that all outputs of the previous phases are available for later ones. For example the design sometimes needs the requirements as input which is provided by the specification phase.

In general the specification phase uses no input, because it is the starting phase and if there is already an object provided this is covered by the technology combination. For the other phases there is the regular product flow as it has already been described in Section 2.2.1 and illustrated in Figure 16.

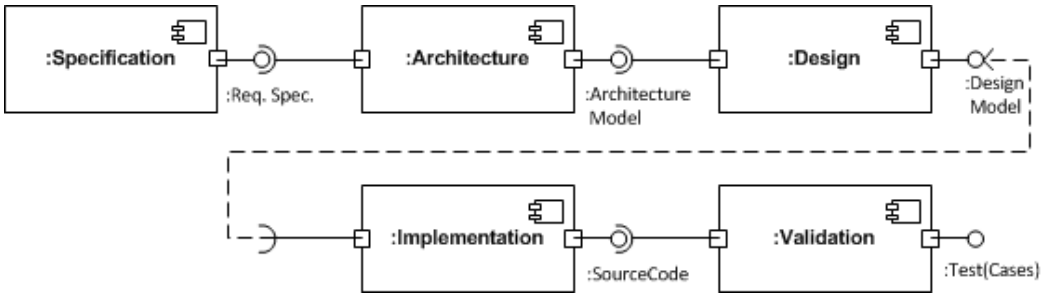

Figure 16. Generic Process Component Diagram

As described later in detail the generic process of Figure 16 is sometimes extended with other components (cf. Section 8.3) to capture the whole life-cycle process. Those extensions are components describing the transformation between other components because of their interface mismatch. In addition there is also the possibility of extending the chain by verification technologies. These extensions will both be treated in the next section.

8.3 Process Component Diagram Extension

As already mentioned in the previous section the PCD sometimes needs to be extended. There are two different cases. First a transformation between two phases is needed because of an interface mismatch. The Technology Chain induces the second case (cf. Section 7.2), because the extended chains differ from the standard ones only in the intermediary verification technologies.

Figure 17. Transformation Extensions of the PCD

The case of a mismatch of inputs and outputs of two different components induces the transformation extension. This results from the interface matching of Section 7.4.1.1. There Rule 11 is responsible for inserting a transformation (cf. Figure 17) into the Technology Chain (Rule 11, `line 10: $tc.addTranformation($i,$o)`). If there are such transformations in the chain we also need to model them in the process component diagram of SPEM [OMG08].

In contrast to the transformation, the second extension results from the extended Technology Chain. Because the standard chain can be extended by verification, this extension must also be supported in the PCD. Within this chain extension verification technologies are plugged in between two technologies.

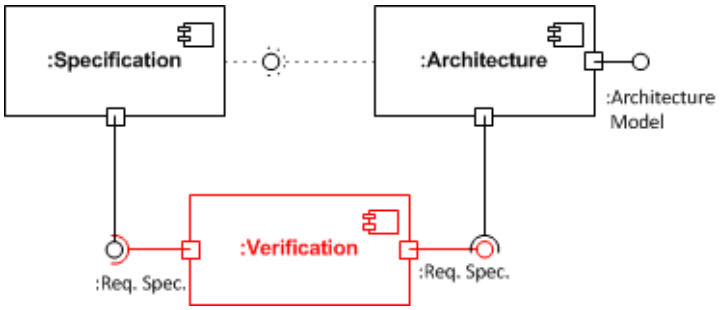

Figure 18. Verification Extensions of the PCD

Similar to Figure 17 the verification extension also needs an extra intermediate component (Figure 18). In contrast to the transformation component the verification technology has exactly the same input and output. Therefore, it is possible to plug it into the diagram without an additional transformation. For the following step we need to be aware of these extensions.

8.4 Instantiation of the Process Components by Technologies

This part of the process configuration uses the PCD with extensions as well as the Technology Chain which was the output of Chapter 7. This step needs both inputs for creating the process pattern as output.

The instantiation illustrated in Figure 19 has two areas. First the Technology Chain is shown on the left part of Figure 19. The other area shows the PCD with one exemplary extension (Figure 19, right). Although the example is a transformation problem (cf. Figure 19, dashed arrow) it works similar to the verification extension. For simplicity reason, we are using one example which covers all possibilities once.

The next step of the process configuration is the instantiation of process components describe the static phases of the SE life-cycle. Therefore, we need to take the chains' technology and instantiate the respective process component. For example if technology A is chosen in the specification phase this technology instantiate the specification component.

In addition to the instantiation of phases, there is one special case. This occurs if one of the technologies has alternatives, then the component is instantiated for each alternative. This is exemplarily illustrated in the implementation phase in Figure 19 (with one alternative).

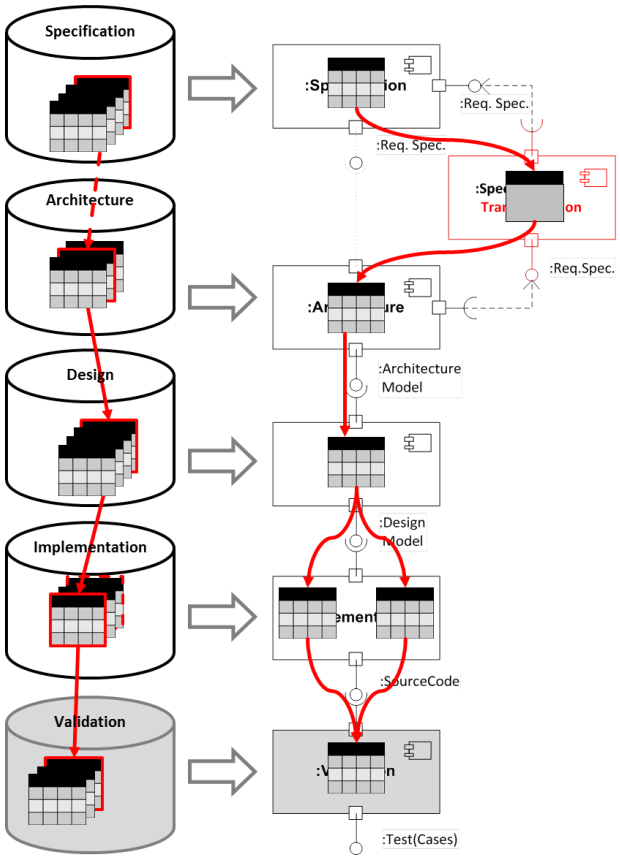

Figure 19. Process Component Instantiation

In addition to the instantiation of the standard chain the transformation or verification components also need to be instantiated. The instantiation of verification extensions uses the description of the technology as it is represented in the technology template (cf. Table 3). Instead the transformations are only used and specified in case of a mismatch. They are not stored in the technology repository and therefore must be specified separately.

8.5 Creation of the Process Patterns

This section deals with the creation of the *process patterns* of SPEM [OMG08] which are sometimes also called *Capability Pattern* (cf. [Ecl12]). Such a pattern is a "special process that describes a reusable cluster" [OMG08] of activities. This concept is important for reuse, because it can be used in different deliverable processes. SPEM process patterns should have the scope of one discipline providing a breakdown structure of reusable complex artifacts such as activities. An advice of the SPEM specification explains to "design process pattern to produce one or more work products. This offers process engineers the possibility of selecting patterns by deciding which products are required as in- and output." [OMG08] This explanation exactly fits to the idea with the inputs and outputs of the technologies (cf. Section 7.4.1.1). In addition the idea to serve as building block for the deliverable process also applies to our suggestions.

Those process patterns do not relate to any specific SPEM phase or iteration. They should be designed in a way that it is applicable anywhere in the deliverable process, which enables flexibility. This is

needed in SPEM because it is based on the Rational Unified Process [EM03] and the idea of using the pattern in the (deliverable) process (cf. Figure 21).

After the introduction of the process pattern the remaining part of this process configuration section deals with their usage and refinement. The usage ranges from the work with the Technology Chain and PCD to the configuration by the user. Instead the refinement is about the different abstraction levels in SPEM and how those patterns are modeled.

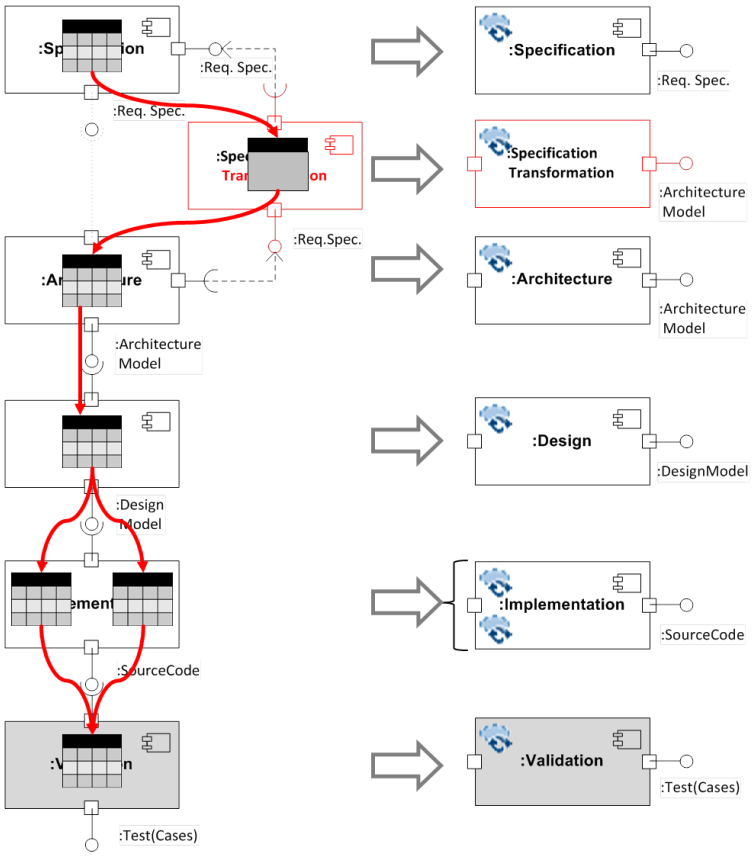

Figure 20. Process pattern Creation

The main part of the usage is shown in Figure 20. This part does not have a user involvement because it is working on the technologies of the chain and the PCD. The left part of the figure depicts the instantiated PCD which is then transformed into the respective process pattern. This is done based on the technology representation of the chain. Each of them has the required *SPEM Model* attribute, containing the patterns for the specific technology. There are all the needed attributes specified, such as *roles*, *activities*, *tasks* and *work documents*. Detailed Information about those SPEM attribute can be found in [OMG08] with an explanation. Those *SPEM Models* representing the process pattern of the single technologies (Figure 20, right) are then used for the process configuration.

Figure 20 also provides an alternative for the technology in the implementation phase. The transformation of these alternatives is done by providing the process pattern for each of the alternative technologies.

In the transition to the next part where the user is involved, the user needs to instantiate the iterations with the process pattern for the deliverable process. In contrast to the first part of the process configuration this one is the user centric part. Figure 21 illustrated this part of the usage. This is the case because we do not know the exact development process of the user or company and do not want to force to use a specific. Instead we are providing the set of process pattern which can be used by the company to easily build their process. There is also tool support for this process instantiation by using the EPF[10] composer [Ecl12].

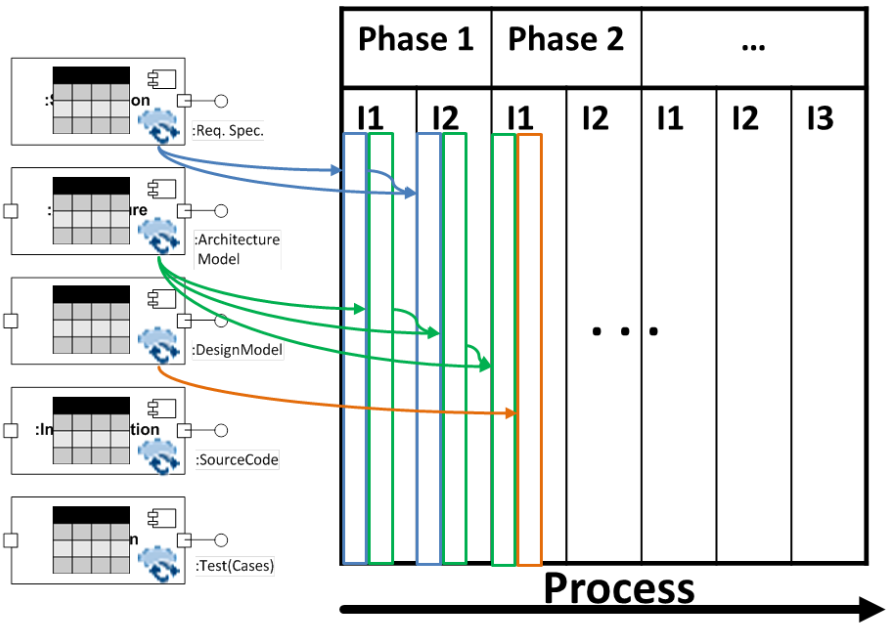

Figure 21. Process Configuration by using process patterns

The refinement of the process pattern is not explained in that much detail because for further details the SPEM specification [OMG08] can be used. Besides the general description aspects of the process pattern the (work) breakdown structure is the most important one for the framework. Within this hierarchical structure the patterns are broken down to the task level. This refinement can directly use tasks or a hierarchy of other patterns and/or activities. Nonetheless the process patterns contain more method content aspects then the task, e.g. work documents, roles, activities and tasks. This is the case because the tasks are related to *roles*, *work documents* and *guidance*. Because they are strong interconnected with elements of the breakdown structure of the patterns they also need to be specified. The whole breakdown structure of a pattern and its details, e.g., different tasks and roles, are defined in the *SPEM Model* attribute of the technology model (cf. Table 3). For further details of the different SPEM object types and how they are modeled, see [OMG08].

8.6 Chapter Summary

The purpose of this chapter was to describe the configuration of the process by using the most appropriate Technology Chain. This was done using the SPEM [OMG08] notation. Throughout this chapter, we first introduced the needed concept of SPEM, the Process Component Diagram. After the extension of this diagram we presented the instantiation of the process components with the technologies of the chain. Then, in the last step the process patterns for the technologies were provided.

[10] Eclipse Processing Framework

Part III: APPLICATION & EVALUATION

The purpose of the previous Part II: RESOLUTION was to provide solutions to the problem identified in Part I: INTRODUCTION INTO RESEARCH of this thesis. Different methods for ranking, selecting and combining of technologies were developed. All of them were built upon a theoretical technology model (gained from [Jed09] and the state-of-the-art survey). The knowledge and output of the different models were further used in the next stages to solve the overall problem depicted in Chapter 4. Finally after the different stage of working with the technology models we provide a possibility to model the Technology Chains as a process based on SPEM.

The purpose of Part III is the presentation of the overall framework as well as its evaluation. The first one is done by composing the different stage of Part II. In addition Part III also includes an evaluation of the framework. The workflow of this part is shown in Figure 22.

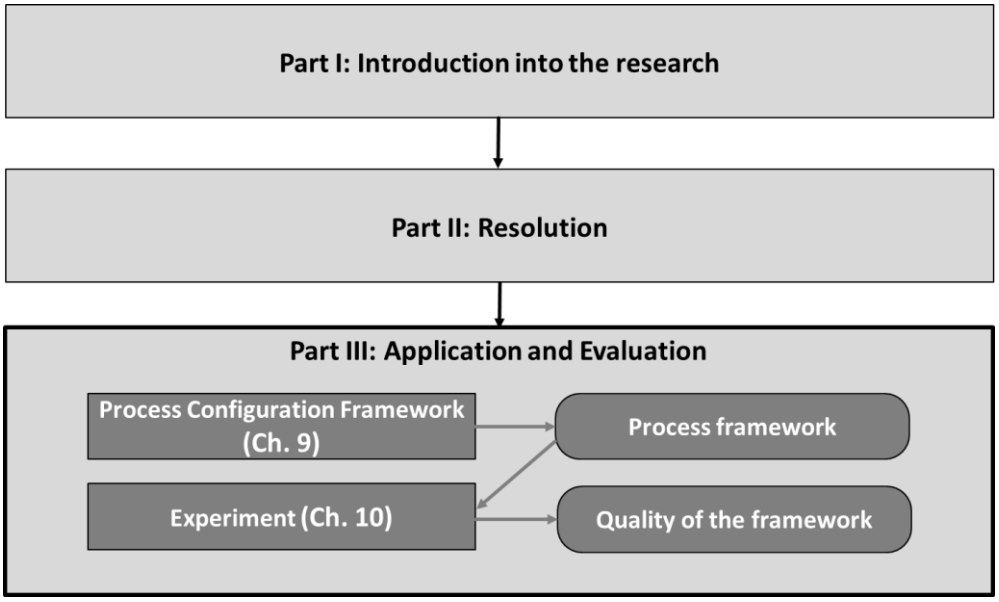

Figure 22. Workflow of Part III: Application & Evaluation

The first chapter of this part (cf. Chapter 9) is about the Process Configuration Framework which is built upon the different stages described in the previous part. Within the consolidation of the four stages, (1) adaption of the technology model, (2) selection and (3) combination of the technologies and (4) process modeling, we describe two different approaches of the framework: the *static* and the *dynamic approach*.

In the second chapter (cf. Chapter 10) we describe the evaluation of the Process Configuration Framework which needed to take place to show the quality of the framework performance. This is evaluated by performing a case study.

9 Process Configuration Framework

The purpose of this chapter is the consolidation of the different stage described in Part II - RESOLUTION. Only the combination of all these approaches solves the problem described in the Chapter 4. The conclusions of this chapter are the two different approaches of the Process Configuration Framework (PCF) build upon the described methods. However, the static phase-based approach is the main focus of the thesis and therefore the dynamic approach is briefly introduced.

The main findings of this chapter are:

- *Combination of different approaches (Chapter 5 - 8)*
- *Detailed Description of the static phase-based approach*
- *Brief Description of the dynamic approach.*

The first section of this chapter describes the prerequisites for both different approaches of the PCF (cf. Section 9.1). These prerequisites are a kind of technology database, explained in detail.

The remaining two sections explain the two different approaches of the framework. With the framework there are more options. First on the highest abstraction level the static as well as the dynamic approach can be used. In the case of the static one there is also the possibility of refining each phase by applying one of the approaches.

In the second section the static or phase-based approach (cf. Section 9.2) is described in detail. It describes the work of the framework in the static way, based on the life-cycle phases of Section 2.2.1. This approach uses all of the different previous explained methods from Chapter 6 - 8.

The last section then provides some information about the dynamic approach (cf. Section 9.3) of the framework, but this is not the main focus of this thesis. Therefore, it is briefly introduced and moved to some future work.

9.1 Prerequisites

The prerequisites for the static and the dynamic approaches of the framework are equal and therefore only presented once. The phase-based as well as the dynamic approach both need some kind of database [RG00] containing a number of technologies described in the uniform technology schema of Section 5.1.

As described, the framework needs a kind of database [RG00]. This database must include the specified technologies, because the framework used the data of these technologies to search, select and combine them. The minimal number of technologies needed to work with the static approach of the framework would be one technology for each of the SE life-cycle phases (cf. Section 2.2.1). However, this case has no realistic relevance because the subsequent stages of selection and combination of the best matching technologies would be irrelevant.

In addition all the technologies in the database need to be specified in a specific uniform way. For this we adapted the models of [Jed09] as it is defined in Section 5.1. Why and how the three models of Jedlitschka are adapted was explained in the Categorization Model section (cf. Section 5.1).

9.2 Phase-based Approach

This section describes the *phase-based approach* of the PCF, by using the methods defined in Chapter 6 - 8. It is also sometimes called *static approach* because we are using the fixed phases for combing the technologies. In the course of our research we recognized that also a dynamic solution might be possible. The idea of the dynamic approach of the PCF as well as its advantages and disadvantages are briefly introduced in Section 9.3.

This approach is static because we use the five static phases of the SE life-cycle described in Section 2.2.1. And based on this assumption we start the different stages of the framework. This has advantages as well as disadvantages. The major advantage is the reduced complexity, because of a fixed number of phases. However, this approach is not that much flexible because of the static SE phases.

After the initial stage with prerequisites of Section 9.1 the stages of the static approach differ from the dynamic one. The phase-based framework depicted in Figure 23 consists out of three stages, whereas all of them have already been described on their own in previous chapters (Chapter 6 - 8). Therefore, the important part of this section is a detailed description of the combination of them.

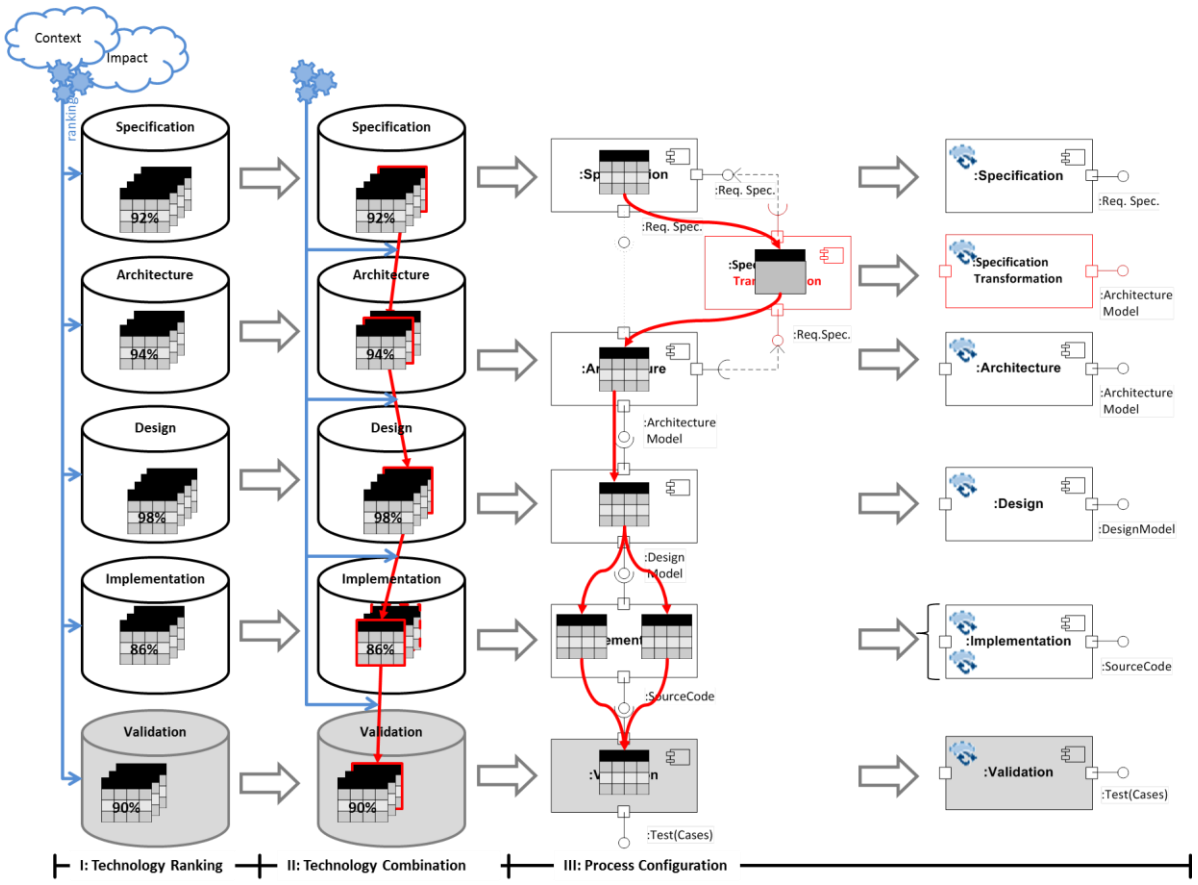

Figure 23. Static Process Configuration Framework

The phase-based approach of the Process Configuration Framework (cf. Figure 23) consists of:

I. Stage – Technology Ranking (Figure 23, left)
II. Stage – Technology Combination (Figure 23, middle)
III. Stage – Process Configuration (Figure 23, right)

Stage I – Technology Ranking

The first stage is based on two different inputs. One is the repository containing the technology descriptions explained in Section 5.1. The other one is the project-specific data of the user/company stored in the input model (cf. Section 0). This includes the different context and impact characteristics.

The process of this stage contains two tasks. The easier task is the *partitioning into the different (logical) repositories* for each of the SE life-cycle phases. It is more difficult to *rank the technologies* based on the matching of the input model (cf. Table 6) with the technology descriptions. Based on this the ranking value is assigned. This is done using rules described in Section 6.2. The tasks in this stage can be performed independently from each other.

The outputs of this stage are (logical) separated repositories (e.g. lists) for each SE phase containing the ranked technologies of the respective phases (Table 3, *SE phase* attribute). As already mentioned those technologies are ranked by a numeric value, which specifies the quality of the matching with the project context and impact.

Stage II – Technology Combination

The process of combining the technologies uses the output of Stage I as input. Therefore, Stage II takes the ranked technologies of the logical repositories of the SE phases.

This stage combines the technologies to find the best Technology Chain which is then provided as output. First the *Chaining Step* creates all possible Technology Chains. Then, these chains run through the *Restriction* and *Ranking Step*. This stage starts by creating all possible Technology Chains. Because this can result in a large number, those chains are then restricted based on some attributes, e.g., technology interdependencies. After this the remaining chains are ranked using the rules of Section 7.4. Within this stage the standard Technology Chain can also be extended by verification technologies.

One Technology Chain, the combination of one technology per life-cycle phase, is the output of the Technology Combination stage. For the following process it does not matter whether it is a standard or extended one.

Stage III – Process Configuration

This stage includes two different inputs. The first and dynamic one is the Technology Chain of the previous stage. In addition there is also a static input, the SPEM Process Component Diagram modeling a generic SE life-cycle process (cf. Figure 16).

The transformation taking place in this stage of the framework is separated into two subsequent tasks. First the two inputs are used to *instantiate the Process Component Diagram by the Technology Chain*. This is done by using the technologies of the chain and instantiating the process components with the respective technologies. Based on those technologies the *process patterns are modeled*. This is done by using their breakdown structure and SPEM elements. This refinement with different SPEM possibilities is explained in Section 8.5.

In contrast to the previous stage the process configuration provides a set of outputs which are the SPEM process patterns. A pattern is provided for each technology of the chain. The format of these patterns can be graphically or XML-based, dependent on how the user needs them. With this output it is then possible to configure the project specific process in the way the user needs (cf. Figure 21).

9.3 Dynamic Approach

As an alternative to the approach described in the previous section (cf. Section 9.2), one could think of a dynamic configuration approach. The *dynamic approach* differs from the static approach especially in Stage I and Stage II of the static one. In particular the ranking would be implemented in a different way in the combination stage. In contrast to the static approach, the dynamic approach is independent from the different static SE phases (cf. Section 2.2.1), i.e., Specification. The dynamic approach identifies paths from a set of possible input objects to all needed output objects (cf. Figure 24). Concepts from graph theory (e.g., [Wes01]) can be used to implement the dynamic concepts of this approach.

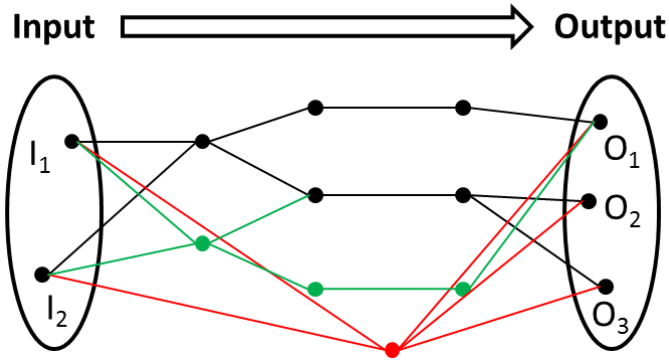

Figure 24. Dynamic Process Configuration Framework

The prerequisites described in Section 9.1 are also applicable to this approach. The dynamic approach of the Process Configuration Framework consists of the following stages:

I. Stage – Technology Combination
II. Stage – Process Configuration

Stage I – Technology Combination

Within this approach the first part of this stage has two inputs. The first one is the definition of the input and output SE objects of the framework. Those are needed because the combination is only based on the interfaces and the products (cf. Figure 24). The second input are most attributes of the input model of Section 0. Not all of those attributes are needed e.g., the *Existing SE Objects* are already in the other first part.

Now the inputs are used for finding for all possible paths that start at one or more input objects and reach all output objects. This is a very complex problem that belongs to graph theory. This is the case, because we need to find all graphs from all possible subsets of the input set $I_1 \ldots I_n$ to all objects of the output set $O_1 \ldots O_m$ (cf. Figure 24). Those graphs then contain a set of technologies similar to the Technology Chains of Section 7.2. Although they do not exactly fit to the chain explanation of Section 7.2 we treat them similar in the next steps.

After we got all possible graphs covering all output objects those Technology Chains are ranked. This ranking takes place in the same way as it is described in Section 7.4 based on parts of the input model (cf. Table 6). Parallel to the combination value the Technology Chain value can be calculated. This is a bit more difficult in the dynamic, because the number of technologies ranges more.

When all the *TechnologyChainValues* (including combination and ranking values) are calculated, the one with the highest value is selected as the output of this stage. This means it is a similar output to Stage II of the phase-based approach. Only the Technology Chain differs in the number and kind of technologies.

Stage II – Process Configuration

The last stage of this dynamic approach uses the selected Technology Chain as input. It seems as if this stage works similar to the Process Configuration stage of the static approach. Unfortunately this is not correct because of the PCD is used in the other approach. This generic diagram explained in Section 8.2 and 8.3 cannot be used because of its inflexibility. The only way to create a workaround is a new individual PCD based on the phases specified in the *SE phase* attribute of the technologies. Last, the instantiation of Section 8.4 takes place in a similar way as it was described in the static approach.

The dynamic approach was not further investigated in the course of this work and is part of future work aspects.

10 Evaluation

The purpose of this chapter is the evaluation of the static approach of the Process Configuration Framework presented in the previous chapter. This evaluation is performed as a case study and will be explained in detail in the upcoming sections.

The main findings of this chapter are:

- *Description of the design of the case study*
- *Providing the results of the case study*
- *Discussion of the evaluation results*

The first section of the evaluation explains the research question and hypothesis, i.e., we explained our expectations with regard to the "performance" of the PCF in an informal way, so that it is possible to discuss them in Section 10.4.

The next section provides all the information about the design of the evaluation (cf. Section 10.2). As required for the description of a case study, this section offers details about the context, the case and the units of analysis. This information is provided at a level of detail so the evaluation can be repeated.

The third section includes the results of the performed case study (cf. Section 10.3). This means it provides the objective presentation of the data and findings of the study. In addition to the different chains, we also provide the configuration data of the experts as well as the positions of the manual chains in the PCF ranking.

The conclusions drawn from the results are explained in the last section (cf. Section 10.4) of this chapter. Within this section the answer to the research question is presented. In addition to this we also discuss all the ranked chains of the PCF as well as the most important experts' comments.

10.1 Hypothesis

This subsection of the evaluation chapter deals with the hypothesis of the evaluation. In contrast to most of the evaluation hypotheses we will provide an informal description of them, because there is no need of a hypothesis for a case study we are performing [RH09].

Similar to the definition section of [RH09] where the overall goal of the evaluation is refined in a set of research questions we are also using a set of hypotheses, whereas "a research question may be related to a hypothesis" [RH09]. This refining hierarchy works similar to the GQM approach of Rombach [BCR01b].

The main research question of our work presented in the Problem Description (cf. Chapter 4) is that we want to provide a framework that configures a process for a specific software project based on a set of technologies. This process should support companies, especially without domain experts, in getting a process initially for a project. Based on this research question we derive an evaluation question. This includes the subject we want to evaluation, in our case that are the different Technology Chains. The evaluation question is then further refined to hypotheses presented in the next paragraphs.

The first hypothesis provides an assumption about the experts. We assume that **all the experts come to a similar Technology Chain, if they have a similar background knowledge and experience in the domain**.

In contrast to the first hypothesis, the next one is more important because it compares the Technology Chains of the experts and the PCF, developed in this thesis. Here we assume that **the Technology Chain manually selected by the expert is similar to the Technology Chain provided by the Process Configuration Framework after it was configured by the same expert**.

For both hypotheses we need to define the similarity of two Technology Chains. This similarity is defined in two ways, whereas only one of them can be used as criteria for the first hypothesis. First, we compare the number of same technologies in both chains, e.g., two out of three are the same. Therefore, we do not consider any similarity between technologies and check whether they are exactly the same or not. Second we observe the position of the Technology Chain in the ranking of the PCF. This way of characterizing the similarity can only be used for the hypothesis comparing the manual with the automatic one because without the working of the framework the position cannot be determined.

10.2 Design

Within this part of the evaluation chapter we describe the design for evaluating the PCF. Before the details of the participants, the used technologies, the scenario and the process are explained in the subsections, a general design is provided.

Because our primary research methodology objective is *exploratory* and partly *descriptive* based on Robson‘s [Rob02] classification we selected the *case study* as methodology. Therefore, the design is very flexible and can be adapted to our needs. The design of our case study includes only one *single-case* [BGM87], the scenario described in a separate section. In addition to the single case we use an *embedded case study* following the definition of [Yin03] to study *multiple units of analysis* with this single case. As suggested by [BGM87] a case study is an instrument to gather information from few entities.

The description of the context, the case and the units of analysis of this case study will be provided in separate sections. The general research question we are focusing on in this evaluation is the working of the PCF as it was expected. Of course this overall and very generic goal is refined with GQM [BCR01b], to come to smaller questions or hypotheses (cf. Section 10.1). However, the metrical aspect of GQM is not that important in our study because the data of a case study is mainly qualitative and the number of units of analysis is too low to perform a statistical test. The methods we used for collecting the data were *direct* and *indirect* methods as categorized by Lethbridge [LSS05]. The direct ones are the data collection of the experts by using *interviews* and *think-aloud* method [SBS94]. The indirect one is the instrumentation with the PCF. The units of analysis are the experts and their provided data.

In addition to the described design of the evaluation or case study based on [Rob02] we are providing a graphical representation of the design (cf. Figure 25) which includes additional information about the process. Figure 25 shows, that the experts and the PCF get the same context (i.e., technology repository as well as the same case, i.e., project scenario). Then the experts create their Technology Chain manually based on the scenario and the given technologies. During this creation the *think-aloud* method [SBS94] was used to collect data. We were interested in the reasons why the experts selected a certain technology for the chain, e.g., which criteria were used, which information was considered, and which assumptions were made. When the chain was created we asked the experts to configure the PCF

using the predefined template (cf. Table 23 and Table 24). With this configuration the framework creates the Technology Chains based on the same scenario and technologies. The PCF uses the standard configuration (cf. Table 7 and Table 10) of the framework as reference value. In addition to the chain with the standard configuration the framework runs with configuration of each expert to compare the manually and automatically created chains.

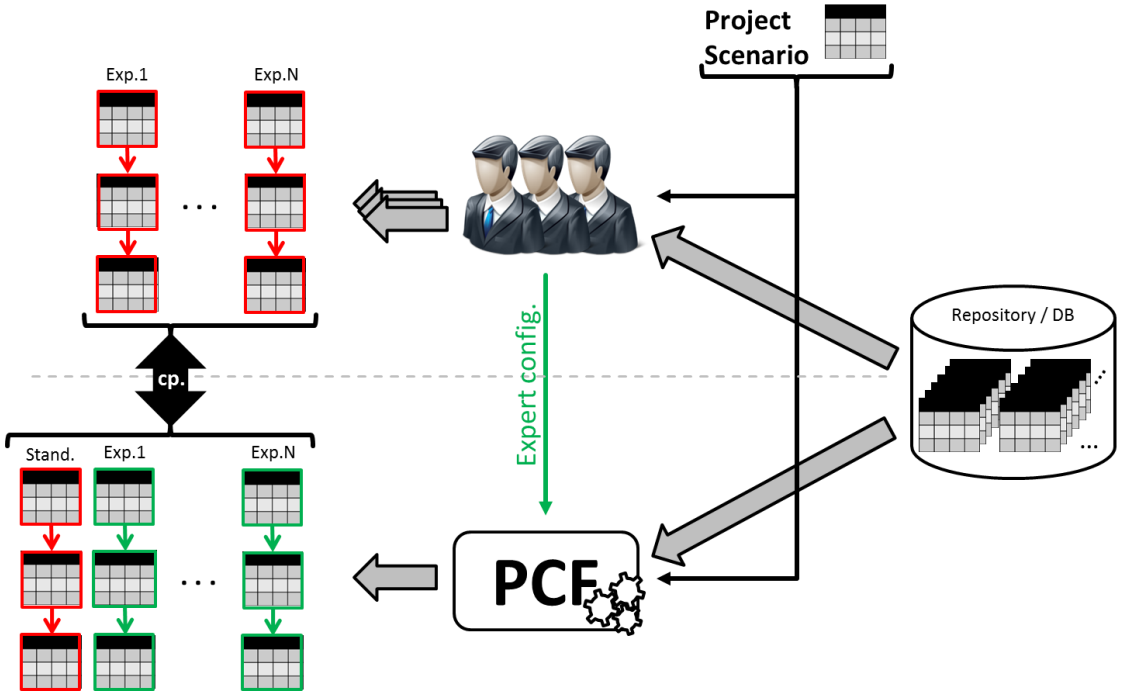

Figure 25. Evaluation Design

The comparison (Figure 25, left) between these Technology Chains is then performed in the results (cf. Section 10.3) and discussion section (cf. Section 10.4).

10.2.1 Unit of Analysis

The units of analysis in our case study are the participants, especially the Technology Chains created by them. This included the manually created one as well as the automatically by the PCF configured with the configurations of the participants.

The population in case of the PCF are managers, for example project managers or SPI[11] managers. This is the case because those managers are the target user group of the framework because they decide about the development process and technologies used in software projects.

Based on this population we selected our sampling. This was a convenient sample trying to get participants which are easy achievable. In this case study the sample consists of four experts of the Fraunhofer IESE. The reason for selecting those IESE employees as experts is based on their domain expertise and project experience. Their domain experience results from the working in the Embedded Systems division of the Fraunhofer, mainly in the department of Embedded Systems Development. In addition to this they are also involved in industrial software projects as well as some of them are working on the SAVE[12] tool. In our opinion this is comparable to the population presented in the previous paragraph. In addition to this they are also involved in supporting company improvements of their processes and in introducing new technologies.

[11] Software Process Improvement
[12] Software Architecture Visualization and Evaluation

For getting more details about the participants of the case study we will provide more information about them in this paragraph, by use of their graduation and role as well as the time they are working in the software / Embedded Systems domain. Two of the four experts have a PhD in computer science and the other two have a diploma[13] in computer science. Most of those participants are involved a longer time in this domain because they are working for at least four years for the Fraunhofer (four, seven and ten years). One of those experts is Head of the Embedded Systems Division of the IESE and the others are working in the Embedded Systems Department. All of them are working in the Embedded Systems for at least the time they are employees of the IESE. Because the participant with the least time at the IESE worked in industry before, all of them are involved in this domain for at least five years. In addition one of them is also the professional management of the studies program "Software Engineering for Embedded Systems" provided by the Fraunhofer.

For practical reasons (e.g., availability of the experts and technology descriptions), we decided to limit the number of life-cycle phases to *Specification*, *Architecture* and *Design*. Further we restricted the number of technologies to *three technologies for each phase*.

10.2.2 Context

The context of the evaluation is similar to the thesis context, the domain of Embedded Systems especially the ARAMiS project.

As already mentioned in the explanation of Figure 25 the PCF uses a technology repository. This is also necessary for the creation of the chains of the experts because without it they could select the chains based on their knowledge and probably each of the four experts knows different technologies. This would lead to non-feasible results. Therefore, we created a set of nine technologies used in this evaluation, three for each of the three phases.

Because the whole research of the thesis came up in the context of the ARAMiS project [KIT11], we focused the specified technologies on the data provided by the project partners. The Fraunhofer IESE and the University of Kaiserslautern collected technologies and their contexts from the industry partners. If technologies were described incompletely, we first searched for more information, e.g., by using the Internet, if we did not succeed, we completed the models with fictive data, e.g. context data.

The technologies[14] used for the evaluation are listed below (abbreviations in parentheses):

Specification:
- Requirements Analysis (RA)
- Requirements Based Engineering (RBE)
- Use Case Specification (UCS)

Architecture:
- Enterprise Architecture (EA)
- Model-Driven Architecture (MDA)
- Unified Modeling Language Method (UMLM)

Design:
- Architecture Design (AD)
- Parallel Programming Model (PPM)
- UML Design (UMLD)

[13] German university degree comparable to the Master of Science (Computer Science)

[14] The detailed specification of all the technologies can be found in the digital appendix of this thesis. We decided not to include them in the appendix because it would result in almost 20 pages of technology, context and impact models.

10.2.3 Case: Scenario – Software Project

In a case study the case itself is one of the most important aspects because it describes the specific situation in which the study takes place. In our case it is an example software project specified as a scenario with the input model (cf. Section 0). In addition to this scenario the setting of the framework for this evaluation is also important and will be discussed in this section.

The scenario for the study must be specified with the input model (cf. Table 6). This was done for all attributes of this model. We excluded the schedule and cost because of not feasible data. The table with the data can be found in the evaluation appendix D.1 (cf. Table 22). We specified a project with the size of *25 person months* which is an *embedded system*, because we thought of a *motor control unit*. This results in the *automotive domain*. In addition to this more general information we also specified the environment data, e.g., tools such as Doors, and fulfilled requirements, e.g., specific certifications such as the ISEB[15] Certificate. The given example project does also contains quality data, e.g. ISO 9126 characteristics (cf. Appendix D.1).

The configuration of the PCF needs to be specified to make the study repeatable. The framework has two modifiable part, the similarity functions of the rules and the configuration used by the PCF. Because the configuration is used in the different units of analysis for the knowledge of the different experts we are now focusing on the similarity functions. In the rules given in Section 6.2 there are a lot of attributes using similarity functions. Those attributes are the *Industry Sector*, *Project Size*, *Qualification*, *Experience*, *ISO 9126*, *Costs* and *Schedule*. Because the majority of the similarity functions are given in the explanations of the rules they do not need to be addressed in detail here. Therefore, we need to explain the similarity functions of the *Qualification* and *Industrial Sector* we are using in the evaluation.

Because of limited information provided in the technology templates, we used simplified functions for modeling the similarity. This would mean that in later evaluations with improved functions the quality of the result would increase. The *Qualification* attribute uses a Boolean function that checks whether the needed and required qualifications are the same or not. The similarity function of the other attributes work similar. Because we do not want to use an incorrect simplicity function, we use the one that increases in case of a correct sector. At least this attribute and its similarity are also included in the future work with an idea how to compare similarities of domains because the development of this would exceed the scope of this thesis.

10.3 Results

Within this section we provide the results from the case study, so they can be interpreted and discussed in Section 10.4. The results are described following the steps of the case study process, described in Figure 25: (1) First the Technology Chains manually created by the experts (cf. Table 12). (2) Then the configurations of the experts (cf. Table 13), which are used to (3) create the chains with the PCF (cf. Table 14).

Table 12. Technology Chains manually created by the Experts

Participant	Specification	Architecture	Design
Expert 1	RBE	UML	UMLD
Expert 2	UCS	MDA	UMLD
Expert 3	RA	UML	UMLD
Expert 4	RBE	UML	UMLD

[15] Information Systems Examination Board (part of the British Computer Society)

Table 12 shows the Technology Chains manually created by the four experts with the abbreviations of the technologies described in Section 10.2.2. From this table, we see that the first and the fourth expert chose the same technologies for their chains, yielding identical chains. Another interesting result is that all experts selected UMLD as the technology for the design phase. In the architecture phase only Expert 2 used another technology than the others. The specification is the phase with the most different results, but still two experts chose the same technology.

Table 13. Different Configurations of the Process Configuration Framework

	Exp.1	Exp.2	Exp.3	Exp.4	Attributes	Exp.1	Exp.2	Exp.3	Exp.4
Comb. Value	30%	50%	20%	70%	Interface	15,0%	25,0%	20,0%	50,0%
					Complements	30,0%	25,0%	50,0%	50,0%
					Paradigm	15,0%	25,0%	20,0%	0,0%
					Dev. Process	40,0%	25,0%	10,0%	0,0%
Ranking Value	70%	50%	80%	30%	Industrial Sector	6,5%	2,0%	28,5%	44,1%
					Size	3,9%	7,5%	4,3%	2,5%
					Kind	9,1%	2,5%	10,0%	2,5%
					IT Environment	6,8%	1,0%	12,8%	22,1%
					Dev. Env.	6,8%	2,8%	12,8%	0,0%
					Dev. Process	6,8%	0,0%	1,4%	0,0%
					Paradigm	2,3%	0,2%	1,4%	22,1%
					Qualification	11,4%	0,4%	5,9%	2,5%
					Training	4,6%	0,4%	5,9%	0,0%
					Experience	6,8%	3,2%	11,9%	2,5%
					ISO 9126	21,0%	26,7%	1,7%	1,0%
					Total Cost	7,0%	26,7%	1,7%	0,1%
					Time to Market	7,0%	26,7%	1,7%	0,9%

Before it is possible to create Technology Chains by the framework the configurations of the experts (cf. Table 13) are needed. From Table 13 we can see that the configurations are very different. Some participants consider the ranking value of the technologies is more important (e.g. 80%) and others consider the opposite (e.g. 30%). Possible reasons and explanations for these differences will be discussed in the next section based on the interview and think-aloud data.

Table 14. Technology Chains automatically configured using different configurations

Config.	Specification	Architecture	Design
Standard	UCS	UML	UMLD
Expert 1	RBE	UML	UMLD
Expert 2	RBE	UML	UMLD
Expert 3	UCS	UML	UMLD
Expert 4	UCS	UML	UMLD

In contrast to the manually created results shown in Table 12 the Technology Chains in Table 14 are configured by the PCF using the standard and experts' configurations (cf. Table 14). It is very interesting, that the results of those five different configurations yield two most appropriate chains (UCS→UML→UMLD or RBE→UML→UMLD). The reasons for this will be explained in the following discussion section.

The technologies marked in Table 14 show the technologies differing from the technologies of the manually created ones (cf. Table 12). Most of the differences are in the specification phase. Except for Expert 2, experts' PCF configurations yielded the same technology for the architecture phase as they chose manually. There are no differences with regard to the selection of technologies for the design phase.

Table 15. Position of the three manually selected chains in the different configurations

Technology Chains	Stand.	Exp.1	Exp.2	Exp.3	Exp.4
RBE→UML →UMLD	2	**1**	1	2	**2**
UCS→MDA →UMLD	3	3	**3**	3	4
RA →UML →UMLD	5	5	8	**4**	3

The complete list of all 27 Technology Chains is given in Table 26 in the appendix. Table 15 provides the excerpt with regard to the Technology Chains that were manually created. Each cell of this table contains the position of this chain within the specific configuration (Table 26, different columns). The bold numbers in Table 15 are the chains manually selected by the respective expert. For example Expert 2 has selected the Chain UCS→MDA→UMLD which is the third best chain in his configuration.

10.4 Discussion

In this section the results of the previous chapter are discussed and we provide some analysis about the presented hypothesis and reasons for the results. After analyzing the resulting data in terms of the hypothesis we provide further material resulting from the evaluation. We further provide some suggestions for improving the study, especially with regard to a larger evaluation with industrial partners. We close this chapter by concluding with all the worked out findings.

10.4.1 Hypothesis

First we want to discuss the data presented in Section 10.3 with regard to the first hypothesis mentioned in Section 10.1. There we assumed that the Technology Chains manually created by the experts are similar to each other.

For the following discussion, we use the results presented in Table 12. Of the four Technology Chains manually chosen by the four experts, two are the same (Expert 1 and 4).Therefore, we use this chain (RBE→UML→UMLD) as a reference for discussing the similarity of the manually selected chains by all experts.

Before discussing the similarity of the chains, we want to provide explanations why the two experts choose this one. For the specification technology there were two possible technologies, RBE and UCS, because of their higher formality. The third technology was excluded because of the plaintext output. The RBE technology was selected because of appropriate qualifications. It also seemed to be more suitable for the scenario of a motor control unit because of certification reasons. The UML technology for the architecture was selected based on elimination of the others. EA was excluded because of its

business orientation and MDA would be overkill for the given scenario. The only possible usage of MDA could be a combination with UML for this phase. The selection of UMLD was also based on elimination of PPL and AD. In AD security is irrelevant and PPL has also been excluded because of its focus on parallelism.

Now, the similarity of the other two chains with this reference chain is discussed. All chains have the same design technology. In contrast to the majority one expert selected a different architecture technology. The reason for this was, he believed two very similar technologies in architecture and design is similar to "comparing apples and oranges". The specification phase seems to be most difficult phase allowing much variation; each of the available technologies was selected at least once. There are several possible reasons for this variety. One reason could be the interpretation of the technologies, for example Expert 3 thought that UCS is included in RA although the interfaces show the opposite. Expert 2 selected the UCS specification because of his prior knowledge. He was more focused on his context and his knowledge then the provided context of the technologies in the models. A further explanation might be based on experts' experience because all of them are more focused on architecture and design and less on requirements engineering.

In addition to the previous discussion with subjective statements of the experts we also tried to verify our hypothesis by an objective measure. One of the few possibilities would have been the Fleiss kappa [Fle81] but the low sample size permitted this.

Concluding for this hypothesis, we observed that the experts selected very similar chains, even if they used different assumptions and had different experience and knowledge. These results indicate that the first hypothesis seems to be right.

Now we analyze the second hypothesis, which is about the similarity of the manually created Technology Chains by the experts compared to the automatically created ones by the PCF with the configuration of the same experts. For this comparison, we use two aspects. One is the similarity of the chains by checking how often they used the same technologies. The other aspect is a Top 5 position of the manually created chain in the ranking of all chains by the framework (cf. Table 15 or Table 26).

Table 16. Technology Chain Similarity

		PCF-Ranking Position (Expert Config.)				
		1	**2**	**3**	**4**	**> 4**
# same Tech.	**3 of 3**	Exp. 1				
	2 of 3		Exp. 4		Exp. 3	
	1 of 3			Exp. 2		

Table 16 shows the similarity of the manually and the automatically created Technology Chains by comparing the number of same Technologies as well as the position of the manually created chain in the ranked list of Table 15. First of all, the chains, which were created manually, are in the Top 4 of the ranking resulting from the PCF using experts' configuration (Table 16, column 1 - 4). In addition three of the four manually created chains use at least two same technologies.

For Expert 1 the manual selection process and the one performed by the PCF using the experts' configuration yielded the identical technology chain. The selected chain was on the first position of the PCF and therefore had the same three technologies as the configured chain. This exact match seems to be the case because this expert was the one who considered the given context of the technologies the

most during the manual selection process. For selecting the specification the output was important because he wanted a kind of formal model, which excluded the RA technology. From the remaining ones he selected RBA because both other technologies were not fitting in the given qualification and experience context. For the architecture phase UML was selected because the other technologies were excluded. EA has the wrong focus because of the business orientation and MDA is too much for the motor control unit scenario. For the design phase UMLD was selected because of the good combination of UML, e.g., the complementary value. For a real project the design would need some additional programming, so that UMLD could be used as documentation.

The chain manually created by Expert 2 is on the third position of the list of chains resulting from the PCF using his configuration, even if there is only one technology similar to the automatically created one (UMLD). The reason for this is that the expert explicitly disregarded the provided contexts and used experience instead. Therefore, it is even more interesting, that the ranking of the manual Technology Chain is that high, because he selected most of the technologies based on his experience in the software domain. After choosing UMLD as technology, the expert selected MDA instead of UML for the architecture because UML and UMLD would be too similar. Although the chains are not that similar based on the number of same technologies, we know from the results that the framework is performing well even if the expert uses his own experience and does not consider the provided context.

Expert 3 is the one with his value on the most right of Table 16, which means that the ranking position was the lowest of all the experts. Two of the three technologies of the chains are the same. This leads to the conclusion that there must be larger disagreement with regard to the technology that does not match, the specification technology. He chose the RA technology, because in his opinion UCS and RA are most often used together. So he had to decide between those both and did this without another criterion, which means probably that he selected RA based on experiences. His experience is that a transformation between SE objects of the phases is not a problem and performed very often. Similar to the first expert, Expert 3 selected the architecture and design technologies based on exclusion criteria. For example he considered that the EA technology is more a business architecture. Overall the automatically and manually created chains are very similar, because the number of same technologies is high and the position in the ranking is only four, but the percentage values of the ranking between two and four (cf. Table 26) are very small.

Besides the first expert, Expert 4 has the most similar result (cf. Table 16). This is the case because both chains have the same architecture and design technology and the manually created chain is also on the second position of the PCF ranking. This participant manually created the same chain as Expert 1 but he configured (cf. Table 13) the framework in a different way so that the created chain differs in the specification phase. The selection criterion used by this expert was a very different when compared to the other experts. For example, he considered RBE to be the best for the scenario because it uses the tool Doors which can be used for specific certifications. For the architecture phase UML was selected because of similar reasons to the other experts. UML itself would be not enough for him. He would extend this technology by UML profiles, and other specific safety modeling techniques, e.g. ESTARELL. Selecting the design technology was the hardest part for this participant. At the end he did not want to select one of the three proposed technologies but two were strictly excluded. PPL because parallel programming with threat distribution would not be needed in the given scenario and the other technology was excluded based on the business context.

The results from all those four experts indicate the correctness of the second hypothesis based on the analyzed data of the case study. This means that the results provided by the PCF can be used as a first proposal for a Technology Chain as well as a process, especially when no expert is available.

10.4.2 Further Findings

Within this section, we discuss further data from the result section. We first compare the configurations of the different experts and also analyze the ranking data (cf. Table 26). In the second part the improvement suggestions for the study provided by the participants during the interview and case study will be discussed.

Starting with the different configurations of the experts (cf. Table 13) we first discuss the distribution of the percentage between the combination and ranking value. From the configuration of the four experts in Table 13 it can be seen that the values differ very much (from 20% combination and 80% ranking value to 70% and 30%). This results from the different preferences of the experts, because the two experts who focus on the ranking value consider that the fitting of the technology to the given context is more important than the combination. This results from the knowledge that it is most often possible to find a good transformation between non-matching technologies. This transformation is also part of our framework introduced in Section 8.3. In contrast the expert focusing on the combination values believes that a continuous tooling without a manual break would be the most important aspect. Based on those findings the four attributes for the *combination value* also differ. Expert 4 focuses only on the interface and the complement, whereas he assumes that interface attribute included the tooling interface and the complement includes identical tooling. Another example is that the first expert ranks the combined development process very high, similar to Expert 4 who believes that the process is fixed and cannot be changed. In contrast to that Expert 3 means that the process in industry is not that important because most of the people deviate from the process and work in their own way. Within the *ranking value* of the same results it can be seen that the experts distributed their values differently. The distribution of the percentage between impact and context ranges very wide. For example, Expert 2 does not consider the context that much, he ranked the impact higher. The other three experts rank the importance of the context higher, especially Expert 3 and 4 (>95%) because they consider that the impact is implicitly high because it depends on the context. In the impact section, Expert 1 ranks the quality much higher compared to the other experts, who use a uniform distribution. The differences in the context attributes are not that high. We discuss the most important differences. Only two of the experts believe that the Industry Sector/Application Domain is one of the most important attributes. The other attributes of the context do not differ that much.

This paragraph deals with the results of the PCF with the different configurations, presented in Table 26. Here we observed the three/five chains with the highest and the lowest *TechnologyChainValue* from all the configurations of the PCF (cf. Table 26). With these chains we want to check, how similar the different configurations work and how the top and the worst chains differ. The observation of the Top 3 chains of the different configurations results in only four chains and the Top 5 in seven chains. A similar result is shown within the lowest chains. There seven chains are under the Worst 5 and six under the Worst 3. The agreement on the top chains is better than on the worst ones. It is interesting to see that all the different configurations result in such similar chains and positions. One possible reason for that is, that the configurations do not matter that much because the technologies are specified in a way that only some combinations are appropriate and most are not. A further explanation for this is the number of technologies, which might have been too low to get more precise results. In addition to the number of top chains in all configurations the range of the Technology Chain value and possible varies very much. The complete range of the Experts 1, 2 and 3 is very similar (between 30% and 42%). In contrast to those three, the last expert has a range of 67%, which is almost twice as high compared to the others. The main reason is the configuration of Expert 4 (cf. Table 13, last column); he rated several attributes with 0%, so that these attributes do not have an influence, which also results in the very high value of the best technology (97%). In none of the other configurations there is such a big gap between two chains like between the first and second chain of the configuration of Expert 4.

Improvements

The improvement part includes information provided by the experts to improve the scenario, the templates and the technologies.

During the development of the material, the challenge was to find enough technologies that would fit together in a given context. Thus we made some assumptions for our example project that might not reflect the current state of the practice. Here our experts mentioned that the use of C or ADA would be more suitable, because they are most used in the automotive domain. In addition to this the V-Model XT development process specified in the scenario is only used in military embedded systems. In the other cases normally (Automotive) SPICE [ISO93] is used. Some additional improvements on the scenario, not directly on the attributes, are only the coverage of the software aspects and not hardware aspects as well as any multithreading. Whereas the last mentioned aspect is not that important because two experts assumed this from the scenario specification as a motor control unit. However, for the purpose of the study, this does not mean to be a threat, because the setting was a controlled rather than a realistic one.

The possible improvements on some attributes of the models (cf. Chapter 5) result from obscurities of one or more experts. The two attributes *size* and *kind* of the *project* element produce some questions. First the project size given in the context and input model are always calculated based on a single person. This means with more persons the project could be speeded up. In contrast the project kind was not enough detailed for most of the experts. They would like to have a more precise kind then the differentiation of Information and Embedded System, e.g., the key functionality such as motor control unit. The next understanding problems appeared in the *Static Context* element. The *experience* given in the template for the technologies only provides information about a recommendation. Another problem was the differentiation of the *qualification* and *training* attribute, because one expert had the opinion that a performed training results in a qualification. This could be a very important point for a new and improved version of the technology model. The last mentioned aspect is about the granularity of the *costs* and *schedule* in the impact model, because the granularity of the template with training, introduction and so on is too detailed and no industrial company would provide this data. They consider that only realistic granularity would be the *Time to Market* or *Total Costs*, which is already used in the input model. In addition to all those specific aspects there is one generic aspect mentioned by one expert. He thinks that such a generic template, for all SE phases, would decrease the quality of the data. Therefore, he would recommend a template with hard and soft criteria, where the hard ones are the same for all the technologies and the soft ones differ in the different phase.

In general, the selection of the technologies for the case study was one of the biggest problems. We tried to get data from ARAMiS project partners but their data was not that much and only partly useful. For this reason we tried to stock up the number of technologies by getting well-known technologies. All this results in the problem, that we mixed technologies that are more a kind of notation (e.g. UML), paradigm (e.g. MDA) or others. This means we did not provide orthogonal technologies in the single phase. Because we are planning to perform a larger evaluation with industry partners later on, we also collected data about possible better technologies as alternatives. Examples for this are Task-oriented Requirements Engineering (TORE) [ADE09] for the specification or ASKET and SIMULINK for the design.

10.4.3 Threats to Validity

All threats that might have an impact on the validity of the results need to be discussed. This includes threats to construct validity, threats to internal validity, threats to external validity and threats to conclusion validity as far as they are an issue. Ignoring these threats can lead to wrong conclusions regarding the validity of the results.

Construct validity refers to the degree to which the operationalization of the measures in a study actually represents the constructs in the real world. Within our case study this refers to the selection of the different technologies, selection of the scenario as well as influence of the researcher on the different experts.

The problem with the nine selected technologies for the experts was that they are not state-of-the-practice. However, we selected them because some were mentioned by ARAMiS project partners from industry and we wanted to focus more on state-of-the-art technologies. The same appeared with the specified scenario, e.g., with the programming language C++. We also assume that those aspects will change in Praxis to become the state-of-the-art.

The different influences to the participants are minimized by giving the same introduction and the same tasks (within a PowerPoint-Presentation) to all of them.

The external validity in our evaluation refers to the degree to which the findings of the study can be generalized to other participant populations or settings. External validity can often be a problem for controlled experiments in artificial environments where the same conditions might not hold in the real world.

The case study we performed for the evaluation was in the Embedded Systems domain, especially in the automotive industrial sector. Because we created the case exactly for this domain the results of this study are restricted to this domain. The results are not only generalizable for this domain, if we assume a larger repository with other technologies and a scenario within another domain.

Another threat of the external validity is the selection of the participants of the study itself. We selected them based on the similarity of a PCF user as well as on their expert knowledge to select the chains manually. In our case this issue was reduced by selecting these experts based on their industrial involvement as well as their management and consulting experience. For further details about the experts see Section 10.2.1.

10.4.4 Conclusion

With this part we are concluding this evaluation chapter by summarizing some important aspects of the previous sections and also providing comments of the experts. The conclusion of the evaluation described in this chapter is that the PCF in general provides a Technology Chain with an appropriate quality. Further, we showed that the framework can be configured for the own purpose. The configuration requires sufficient experience. The final decision needed to be made by experts.

Another important observation derived from the evaluation is, that for some participants it is very important to provide the possibility of transforming SE objects to another format because of a mismatch. This is supported in the framework by extending the SPEM Process Component Diagram (cf. Figure 16) with a transformation component (cf. Section 8.3).

Another thing one of the experts mentioned was the area of operation of the framework. He thought that such a framework is an interesting idea, but would be used rarely because most of the industrial companies select all their technologies, tools, etc. once and then use this until the next technology period, e.g. Model-driven Development. This would mean the framework could be used only for each technology period and not for the micro iterations in it. In addition, if such a process were fixed by the industry there could also be variation points in this process, e.g., only an inspection is fixed but the technology used for the inspection could be chosen from Perspective-base, Checklist or Ad-Hoc Reading. However, he provided other ideas for using this or similar frameworks: For example the technology selection in a standard that provides a range of possible technologies to use. Examples for this are [IEC10] and [EUR99] that provide as set of technologies to use in the aerospace domain.

11 Conclusion and Future Work

The research problem addressed in the course of this thesis was that selecting the best technologies over the whole life-cycle for one specific software project is a complex task and most often it can be performed only by domain experts. To solve this problem, we developed a framework for configuring a process based on a project-specific Technology Chain. To arrive at this framework we first adapted a generic model for technologies, their contexts, and impacts to use these as a foundation. Based on this model, we developed strategies for rating the technology match to the project context as well as the technology combination. Finally the chain is transformed to SPEM process patterns to create a process. The framework provides support for the process configuration at the project beginning. It also provides an opportunity to transfer new research technologies faster and easier to industry.

The purpose of this chapter is to present our conclusions on the research problem (cf. Section 11.1) and state the contributions of the thesis (cf. Section 11.2). The limitations of this research are described (cf. Section 11.3), and opportunities for future research are presented (cf. Section 11.4). The final section is dedicated to some concluding remarks (cf. Section 11.5).

11.1 Conclusion on the Research Problem and Questions

One major assumption behind this thesis was that currently it is possible for only domain experts to have knowledge about existing technologies and the project context to select the best technologies and combine them to a process. We discussed that this threatens the success of software projects in industry companies and, in turn, threatens their business. Therefore, we provide a framework which supports companies in configuring their process by the use of the most suitable technologies.

The high-level research problem addressed by this thesis was:

Specific software projects need technologies that fit the context and the addressed goals. The problem is determining how to configure the best project specific process based on different technologies.

The focus of the investigation was narrowed down by studying research questions discussed in the following research questions. These questions were derived based on the GQM approach [BCR01b].

Research question 1: How can we specify all possible technologies in one generic schema?

To answer the first research question, we used the technology model as well as the dependent context and impact models from [Jed09]. Those three models were adapted because of the need of a new element. In addition, further elements needed to be refined or transferred from one model to another.

The main conclusion for this research question is that there is a generic model that covers all needed aspects of a technology, its impact and context. However, such a generic schema has the drawback that there are no soft attributes that differ for the different phases.

Research question 2: How can we build project-specific Technology Chains?

The second research question focuses on the creation of Technology Chains. This can be separated into two steps, first, the ranking of how well all the technologies match the given context, and second, the combination of all possible technologies. For these two steps, rules were implemented that perform this ranking and combination.

The conclusion for research question two is that it is possible to build a project-specific Technology Chain with a given configuration of the importance of the schema attributes.

Research question 3: How can we configure a software process based on a number of technologies?

To answer the last research question, we used the Software & System Process Engineering Meta-Model (SPEM). Within this model the Process Component Diagram was used to instantiate the different process components with the technologies of the chain. Then each of the technologies provides a small SPEM specification as process pattern, indicating how the technology works. With those patterns it is now possible to configure its own process by plugging the pattern in the needed phases or iterations, whatever development process the company is using.

The main conclusion for this research question is that SPEM provides a good possibility for the process modeling we are interested in. This is because it is built upon the Rational-Unified Process [EM03] which exactly meets our idea of providing process patterns so that the user is free to configure the process by using these patterns.

11.2 Contributions

The main contribution presented in this thesis is the support of software processes for an individual software project by selecting and combining the most appropriate technologies. In addition, we contributed to theory, methodology, and practice.

The **theoretical contribution** made in this thesis:

An adaption of the technology schema of [Jed09]. Based on the work of Jedlitschka, we defined a technology schema for applying it in this thesis. In doing this we used the models proposed in [Jed09]. Two of them were adapted to fit our needs exactly. This adaption was a new attribute in the technology model and a transfer of an element from the context model to the technology model. In addition we refined the explained attributes. The schema was developed throughout Section 5.1.

The most important **methodological contributions** made in this thesis:

An XML schema for the three models of the adapted technology schema. These schemas are an implementation and refinement of the adapted technology model of the theoretical contributions. The model refinement provided the possibility of specifying new technologies directly in an XML document. It was needed to implement the rules of the practical part. The whole schemas are illustrated in Appendix B.

Combination of SPEM process patterns with the SPEM Process Component Diagram. For the last step of the developed framework we needed a possibility for transforming the Technology Chain in a process. Therefore, we used the SPEM notation, which includes the Process Component Diagram as well as the concept of process patterns. With these concepts, the transformation was realized and was described in Chapter 8.

The most important **practical contributions** made in this thesis:

A systematic approach to configure a process for a specific software project using the framework. Another element is the implementation of a framework in support of configuring a process based on context specific technologies. It consists of using the previously mentioned technology, context, and impact model; ranking the different technologies based on their matching with the specific project and ranking of all possible Technology Chains. The framework assists a user during the initial creation of a process for a software project. The different approaches were described in Chapter 9.

Rules for the ranking, restriction, and combination of the technologies. For the implementation of this Process Configuration Framework, especially its ranking, restriction, and combinational parts, rules were needed. The rules were used to perform the ranking and combination steps and were described in Section 6.2, 7.3 and 7.4.

In addition, we evaluated our approach by conducting a case study with four Fraunhofer IESE experts, who, because of their industrial projects, have experience in the used domain. The results of the evaluation showed that the Technology Chains automatically created by the PCF are qualitatively similar to the chains manually selected by the domain experts.

11.3 Limitations

The whole PCF and its different stages developed in this thesis were based on the repository of the technologies. This was the first limitation since this repository did not exist to the extent that it could be used for our purpose. This resulted in several problems and limitations which are described in here.

In your research, the framework was mainly developed on a conceptual level and some important parts were implemented. The most important parts of the implementation were the ranking, restriction, and combination rules using the rule engine Drools Expert [JBo12]. It would have been impossible to implement the rules without the implemented technology model section. We implemented the models in Java based on the XML schema representations. Additionally we implemented the PCF in Excel to verify the implemented rules in the context of a case study. To combine all the implementation parts we developed a technical realization concept, which can be seen in Appendix C.

Other limitations presented themselves at the beginning/planning of the evaluation. First, there was only the Excel implementation of the framework, and secondly, the technology repository was nonexistent. To solve the first problem we used a paper and pencil evaluation to perform the framework. To address the missing repository, a limited technology repository was created in the context of the case study. Most of the created technologies are based on ideas from industry partners of the ARAMiS project. However, it was hard to get all the data needed for the technology schema. This limitation prevented us from using a statistical test for the verification of the hypothesis.

11.4 Future Work

The research results presented in this thesis along with its contributions and limitations provide a basis for further research areas of software engineering, especially the technology selection and technology combination, as well as the process configuration.

We have already indicated that companies can benefit from the **exchange of new research technologies** and the **support in configuring the process**. Also, research can benefit from **faster feedback and improvements of new technologies**. Figure 26 shows a context in which the framework could be used. To work with this framework more efficiently, organization specific improvement activities need to be connected with external sources of information [JP04], which should be resolved by the framework.

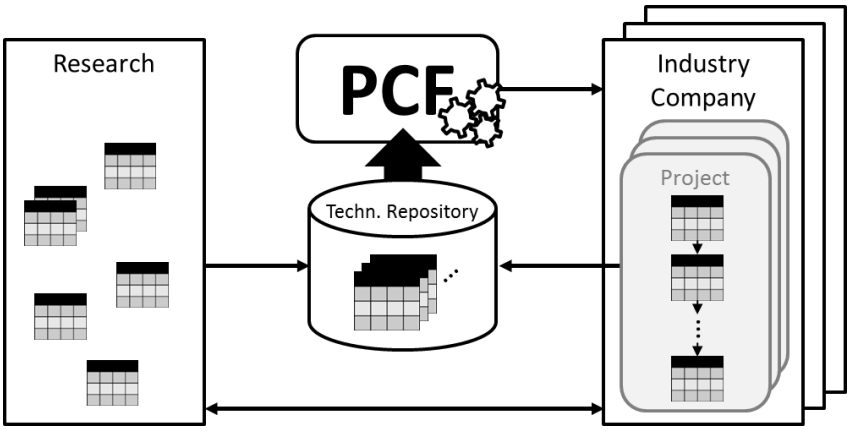

Figure 26. Process Configuration Framework in Use

The PCF and its usage are illustrated in Figure 26. The bold arrow from the technology repository to the PCF represents the technologies specified in the schema of Chapter 5 that are used by the PCF to configure the process. It is a very important part for the context of the usage of the framework. It is the place where research places its new technologies. It is integrated in the framework-industry-circle. This includes the usage of the framework by the industry, where the companies get a number of technologies to use them as a process. And after they finished a project they could then provide improvements to the repository, such as, new contexts in which the technologies work.

As it was mentioned in the limitation section, some of the biggest problems we faced were the lack of technologies in our template and a technology repository containing them. Therefore, this **technology repository using the introduced technology schema** could be the next and most important aspect for future work. An example of this could be the development of a database based on XML, since the technology schema has already been implemented as XML schema during the research for this thesis.

The development of this thesis covers the static approach in detail and only briefly introduces the idea of the **dynamic approach of the Process Configuration Framework** (cf. Section 9.3). The dynamic approach is more flexible and adaptable because of the independence of its phases. It was not the focus of this thesis and is, therefore, mentioned as an alternative or detailed approach. Because this is a very complex approach, for example, by using graph theory, it could be a thesis on its own. In contrast to the following future work ideas, more research effort on a conceptual level is needed in here.

Many rules require a **similarity function to describe the similarity between two inputs**. Some of these inputs are already described entirely because e.g., they use numeric values, and the similarity is covered by a mathematical function. There are other attributes that cannot be described as a numeric value. For example, the application domain attribute describing the domain of the software project could be something like Automotive, Avionics, Railway, Telecommunication, or Health care. In the evaluation such attributes were assigned by a similarity value of 100% or 0%, because there is no categorization for it. One possibility for an application domain attribute could be the inspection of how many quality attributes important for each domain are the same. Because this is only one of probably many possibilities, this would be a good area for further research.

Another interesting topic for future work would be **coding the technical binding to achieve a complete implementation of the Process Configuration Framework**. Details of the technical realization of the implementation are in Appendix C. In addition to the structure of the implementation this appendix chapter includes some further aspects of the implementation which could be developed in the future, e.g., the implementation of a web service.

In this thesis we evaluated the framework by the use of a case study with four experts and a limited number of technologies. Because of this and other limitations and opportunities for future research, we are planning to **perform a bigger evaluation with the implemented framework, many more technologies from the ARAMiS project, and including industrial experts as participants**. This evaluation should then provide more data that could also be used for statistical testing.

The last of our future prospects involves using the results from the evaluation performed in our research. Our thought is to **include the possible improvements mentioned by the participants of the case study**. This will result in an improved evaluation with a more realistic scenario or better technologies. All the improvements suggested by the experts are discussed in Section 10.4.2.

11.5 Concluding Remarks

The work presented throughout this thesis showed that it is possible to create a framework for configuring a process by providing SPEM process pattern for the Technology Chain including the most appropriate context matching and combined technologies. Using this framework in an evaluation, we showed that the frameworks result, given a specific configuration, was similar to one that was manually selected by experts, which was an excellent result. Once the framework is configured by an expert for a specific kind of project, its results can be used by non-experts as a proposal for starting a project.

One major objective of this work was to find means for supporting the creation/configuration of a software process at the beginning of a project. The idea was that the developed framework provides the process pattern for each technology of the Technology Chain so that the only thing to do is to use those patterns in the specific phases or iterations, as the RUP [EM03] introduced this concept. We hypothesized that in industry often the same technologies are reused and adapted although there are many more context specific and better fitting ones. To improve this situation, we proposed an approach that uses all available technologies, including newly researched ones, to build a Technology Chain and process from them.

Some of the problems mentioned in this thesis, one being that most companies' change their technologies only when there is a significant change in technology, will remain, because our approach will have no influence on this. However, we assert that our approach is one step towards an automatic selection of project specific technologies and automatic creation of a project specific process.

Other aspects of process configuration of technology selection and combination were considered to be beyond the scope of this work, although we did address them in the background.

References

[Aal11] van der Aalst, W. M.: Business Process Configuration on the Cloud: How to Support and Analyzse Multi-tenant Processes?. In Proc. of the *9th IEEE European Conference on Web Services*, IEEE, Lugano, Switzerland, 2011, P. 3-10.

[ADE09] Adam, S.; Doerr, J.; Eisenbarth, M.; Gross, A.: Using Task-oriented Requirements Engineering in Different Domains – Experiences with Application in Research and Industry. In Proc. of the *17th IEEE International Requirements Engineering Conference*, Atlanta, GA, USA, 2009, P. 262-272.

[AHW03] van der Aalst, M. P.; ter Hofstede, H. M.; Weske, M.: Business Process Management: A Survey. In Proc. of the *1st International Conference on Business Process Management*, Springer, Eindhoven, Netherlands, 2003, P. 1-12.

[Apa03] Apache Software Foundation (ASF): *Java XMLBeans*. Apache Software Foundation, 2003.

[Apa08] Apache Software Foundation: *XMLBeans User Documentation*. 2008.

[BB96] Barron, F. H.; Barrett, B. E.: The efficacy of SMARTER - Simple Multi-Attribute Rating Technique Extended to Ranking. In *Acta Psychologica*, 93, Elsevier Science, Amsterdam, Netherlands, 1996, P. 23-36.

[BCR01a] Basili, V. R.; Caldiera, G.; Rombach, H.: Experience Factory. In Marciniak J. J. *Encyclopedia of Software Engineering*; John Wiles & Sons, Inc., New York, NY, USA, 2001, P. 511-519.

[BCR01b] Basili, V. R.; Caldiera, G.; Rombach, H.: Goal Question Metric Paradigm. In Marciniak J. J. *Encyclopedia of Software Engineering*; John Wiley & Sons, Inc., New York, NY, USA, 2001, P. 528-532.

[BGM87] Benbasat, I.; Goldstein, D. K.; Mead, M.: The Case Research Strategy in Studies of Information Systems. In *MIS Quarterly*, 11, Management Information Systems Research Center, University of Minnesota, MN, USA, 1987, P. 369-386.

[Bir] Birk, A.: *A Knowledge Management Infrastructure for Systematic Improvement in Software Engineering*; Fraunhofer IRB Verlag, Stuttgart, Germany, 2000.

[Bir97] Birk, A.: Modelling the application domains of software engineering technologies. In Proc. of the *12th IEEE International Conference on Automotive Software Engineering*, IEEE, Incline Village, NV, USA, 1997, P. 291-292.

[BPS06] Bray, T.; Paoli, J.; Sperberg-McQeen, C. M.; Maler, E.; Yergeau, F.; Cowan, J.: *Extensible Markup Language (XML)*; W3C, MA, USA, 2006.

[BR06] Bottani, E.; Rizzi, A.: A fuzzy TOPSIS methodology to support outsourcing of logistics services. In *Supply Chain Management: An International Journal*, 11, Emerald Group Publishing Limited, Bingley, England, 2006, P. 294 - 308.

[BR88] Basili, V.; Rombach, H.-D.: Towards A Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment. In Proc. of the *19th Annual Software Engineering Workshop*, Maryland, USA, 1988.

[BR91] Basili, V.; Rombach, H.-D.: Support for Comprehensive Reuse. In *Software Engineering Journal*, 6, IEE British Computer Society, New York, NY, USA, 1991, P. 303-316.

[CKO92] Curtis, B.; Kellner, M. I.; Over, J.: Process modeling. In *Communications of the ACM - Special issue on analysis and modeling in software development*, 35, ACM, New York, USa, 1992, P. 75-90.

[CKS11] Chrissis, M. B.; Konrad, M. D.; Shrum, S.: *CMMI for Development: Guidelines for Process Integration and Product Improvement, Third Edition*; Addison-Wesley Professional, New York, NY, USA, 2011.

[DH01] Dumas, M.; ter Hofstede, A. H.: UML Activity Diagrams as a Workflow Specification Language. In Gogolla M.; Kobryn C. *The Unified Modeling Language. Modeling Languages, Concepts, and Tools*; Springer, Toronto, Canada, 2001, P. 76-90.

[Din02] Dingsøyr, T.: Knowledge Management in Medium-Sized Software Consulting Companies. In *Empirical Software Engineering*, 7, Kluwer Academic Publishers, Trondheim, Netherlands, 2002, P. 383-386.

[Doy02] Doyle, D. P.: Knowledge-based decision making. In *The School Administrator*, American Association of School Administrators, Alexandria, VA, USA, 2002, P. 30-34.

[Ecl12] Eclipse Foundation: Eclipse Process Framework Project (EPF). Ottawa, Canada, 2012.

[EM03] Essigkrug, A.; Mey, T.: *Rational Unified Process kompakt*; Spektrum Akademischer Verlag, Heidelberg, Germany, 2003.

[EM07] Einhorn, H. J.; McCoach, W.: A simple multiattribute utility procedure for evaluation. In *Systems Research and Behavioral Science*, 22, Wiley, New York, NY, USA, 1977, P. 270–282.

[EUR99] European Organisation for Civil Aviation Equipment (EUROCAE): *DO-178B*; Paris, France, 1999.

[FH93] Feiler, P.; Humphrey, W.: Software process development and enactment: Concepts and definitions. In Proc. of the *2nd International Conference on the Continuous Software Process Improvement*, IEEE, Berlin, Germany, 1993, P. 28-40.

[FKV91] Fraser, M. D.; Kumar, K.; Vaishnavi, V. K.: Informal and Formal Requirements Specification Languages: Bridging the Gap. In *IEEE Transactions on Software Engineering*, 17, IEEE, New York, NY, USA, 1991, P. 454-466.

[Fle81] Fleiss, J. L.: *Statistical methods for rates and proportions*; John Wiley & Sons, New York, NY, USA, 1981.

[FM08] Fraser, S.; Mancl, D.: No Silver Bullet: Software Engineering Reloaded. In *IEEE Software*, Vol. 25, IEEE, New York, NY, USA, 2008, P. 91-94.

[FM99] Fichman, R. G.; Moses, S. A.: An Incremental process for Software Implementation. In *Sloan Management Review*, 40, Massachusetts Institute of Technology, MA, USA, 1999, P. 39-52.

[FW06] Fortune, J.; White, D.: Framing of project critical successfactors by a systems model. In *International Journal of Project Management*, 24, Elsevier, Netherlands, 2006, P. 53-65.

[GG94] Gilb, T.; Graham, D.: *Software Inspection*; Addison-Wesley Longman, Amsterdam, Netherlands, 1994.

[GWJ91] Gottschalk, F.; Wagemakers, T. A.; Jansen-vullers, M. H.; Aalst, W. M.; Rosa, M.: Configurable Process Models: Experiences from a Municipality Case Study. In Proc. of the *21st International Conference on Advanced Information Systems Engineering (CAiSE)*, Springer-Verlag, Amsterdam, Netherlands, 2009, P. 486-500.

[Hay85] Hayes-Roth, F.: Rule-based systems. In *Communications of the ACM*, Vol. ACM, New York, NY, USA, 1985, P. 921-932.

[Hei11] Heidrich, J.: *Process Modeling*; Lecture Slides, Kaiserslautern, 2011.

[Hen96] Henninger, S.: Accelerating the Successful Reuse of Problem Solving Knowledge Through the Domain Lifecycle. In Proc. of the *4th International Conference on Software Reuse*, IEEE, Washington D.C., USA, 1996, P. 124-133.

[HKS02] Hansch, M.; Kuhlins, S.; Schader, M.: XML-Schema. In *Informatik-Spektrum*, Springer, Berlin, Germany, 2002, P. 363-366.

[IEC10] International Electrotechnical Commission (IEC): *CEI IEC 61508-3 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*; Geneva, Switzerland, 2010.

[ISO01] International Organization for Standardization (ISO): *ISO/IEC 9126 - Software Engineering - Product Quality*; Geneva, Switzerland, 2001.

[ISO08] International Organization for Standardization (ISO): *ISO 12207 - Systems and Software engineering - software life cycle processes*; Geneva, Switzerland, 2008.

[ISO93] International Organization for Standardization (ISO): *ISO/IEC 15504 - Information technology - Process assessment*; Geneva, Switzerland, 1993.

[JAD01] Jedlitschka, A.; Althoff, K.-D.; Decker, B.; Hartkopf, S.; Nick, M.: Experience Management: The Fraunhofer IESE Experience Factory. In Proc. of the *4th International Conference on Case-Based Reasoning*, Vancouver, Canada, 2001.

[Jal08] Jalote, P.: *A Concise Introduction to Software Engineering*; Springer-Verlag, London, England, 2008.

[JBo12] JBoss Drools Team: *Drools Expert User Guide*; 2012.

[JCF07] Jedlitschka, A.; Ciolkowski, M.; Freimut, B.; Schlichting, A.: Relevant Information Sources for Successful Technology Transfer: A Survey Using Inspections as an Example. In Proc. of the *1st International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, Madrird, Spain, 2007, P. 31-40.

[Jed09] Jedlitschka, A.: *An Empirical Model of Software Managers' Information Needs for Software Engineering Technology Selection*; Fraunhofer IRB Verlag, Stuttgart, 2009.

[JHS05] Jedlitschka, A.; Hamann, D.; Schröder, A.: Adapting PROFES for Use in an Agile Process: An Industry Experience Report. In Proc. of the *6th International Conference on Profuct-Focused Software Process Improvement (PROFES '05)*, Oulu, Finnland, 2005, P. 227-304.

[JP04] Jedlitschka, A.; Pfahl, D.: Towards Comprehensive Experience-Based Decision Support. In Proc. of the *11th European Conference on Software Process Improvement (PROFES '04)*, Springer-Verlag, Trondheim, Norway, 2004, P. 34-45.

[KIT11] Karlsruher Institut für Technologie (KIT): *Vorhabensbeschreibung - ARAMiS (Automotive, Railway and Avionic Multicore Systems)*. Karlsruher Institut für Technologie, Karlsruhe, Germany, 2011.

[KB09] Kim, H.; Bond, R.: Multicore software technologies. In *IEEE Signal Processing Magazine*, 26, IEEE, New York, NY, USA, 2009, P. 80-89.

[Kit04] Kitchenham, B.: *Procedure for Performing Systematic Reviews*. Keele University, Software Engineering Group, Keele, UK, 2004.

[KLM97] Kiczales, G.; Lamping, J.; Mehdhekar, A.; Maeda, C.; Lopes, C. V.; Loingtier, J.; Irwin, J.: Aspect-oriented programming. In Proc. of the *11th European Conference on Object-Oriented Programming (ECOOP)*, Springer-Verlag, Jyväskylä, Finnalnd, 1997, P. 220-242.

[KM86] Kodratoff, Y.; Michalski, R. S.: Probabilistic Decision Trees. In Kodratoff Y.; Michalski R. *Machine Learning: An Artificial Intelligence Approach*; Morgen Kaufmann Publishers, San Mateo, CA, USA, 1986, P. 140-152.

[Kru95] Kruchten, P.: Architectural blueprints - the "4+1" View Model of Software Architecture. In *IEEE Software*, 12, IEEE, New York, NY, USA, 1995, P. 42-50.

[Lau02] Lauesen, S.: *Software Requirements: Styles and Techniques*; Pearson Education, London, England, 2002.

[Lon93] Lonchamp, J.: A Structured Conceptual and Terminological Framework for Software Process Engineering. In Proc. of the *2nd International Conference on Continuous Software Process Improvement*, IEEE, Berlin, Germany, 1993, P. 41-53.

[LSS05] Lethbridge, T. C.; Sim, S. E.; Singer, J.: Studying Software Engineers: Data Collection Techniques for Software Field Studies. In *Empirical Software Engineering*, 10, Springer, Berlin, Germany, 2005, P. 311–341.

[Mar89] Marsan, M. A.: Stochastic Petri nets: An elementary introduction. In Proc. of the *12th International Conference on Applications and Theory of Petri Nets*, Springer, Gjern, Denmark, 1989, P. 1-29.

[MRB82] Meadows, D.; Richardson, J.; Bruckmann, G.: *Grouping in the Dark*; John Wiley & Sons, New York, NY, USA, 1982.

[Nør10] Nørmark, K.: Overview of the four main programming paradigms. 2010.

[OMG08] Object Management Group (OMG): *Software & Systems Process Engineering Meta-Model Specification - SPEM*; Needhham, MA, USA, 2008.

[OMG11] Object Management Group (OMG): *Meta Object Facility*; Needham, MA, USA, 2011.

[Ost87] Osterweil, L. J.: Software processes are software too. In Proc. of the *9th International Conference on Software Engineering (ICSE '87)*, ACM, Los Alamitos, CA, USA, 1987, P. 2-13.

[Pru81] Prugovečki, E.: *Quantum mechanics in Hilbert space*; Academic Press, New York, NY, USA, 1981.

[Pyr99] Pyrczak, F.: *Evaluating Research in Academic Journals*; Pyrczak Publishing, Los Angeles, CA, USA, 1999.

[Rak01] Rakitin, S. R.: *Software Verification and Validation for Practitioners and Managers*; ACM, Nordwood, MA, USA, 2001.

[RG00] Ramakrishnan , R.; Gehrke, J.: *Database Management Systems*; ACM, Hill Berkeley, CA, USA, 2000.

[RH09] Runeson, P.; Höst, M.: Guidlines for conducting and reporting case study research in software engineering. In *Empirical Software Engineering*, 14, Springer, Berlin, Germany, 2009, P. 131-164.

[RII05] Raja, B. S.; Iqbal, M. A.; Ihsan, I.: Moving from Problem Space to Solution Space. In *World Academy of Science - Engineering and Technology*, 2005, P. 36-39.

[Rob02] Robson, C.: *Real World Research: A Resource for Social Scientists and Practitioner-researchers*; John Wiley & Sons; Blackwell Publichers, NJ, USA, 2002.

[Ros10] Ross, T. J.: *Fuzzy Logic with engineering applications*; John Wiley & Sons, Chichester, UK, 2010.

[RR85] Redwine, S. T.; Riddle, W. E.: Software Technology Maturation. In Proc. of the *8th International Conference on Software Engineering*, IEEE Computer Soc. Press, Los Alamitos, CA, USA, 1985, P. 189-200.

[Ruh03] Ruhe, G.: Software Engineering Decision Support - A New Paradigm for Learning Software Organizations. In Proc. of the *5th International Workshop on Advances in Learning Software Organizations*, Springer, Chicargo, IL, USA, 2003, P. 104-113.

[SBS94] van Someren, M. W.; Barnard, Y. F.; Sandberg, J. A.: *The think aloud method: A practical guide to modelling cognitive processes*; Academic Press, London, England, 1994.

[SEI97] Software Engineering Institute (SEI): *SEI C4 Software Technology Reference guide - A Prototype*; Pittsburgh, PA, USA, 1997.

[SSL07] Shih, H. S.; Shyur, H. J.; Lee, E. S.: An extension of TOPSIS for group decision making. In *Mathematical and Computer Modelling*, Elsevier, Amsterdam, Netherlands, 2007, P. 801–813.

[Sum07] Summerville, I.: *Software Engineering*; Addison-Wesley, Boston, MA, USA, 2007.

[TUM09] Technische Universität Münschen - Institut für Informatik: *Das V-Modell XT 1.3 Metamodell*; Institut für Informatik der Technischen Universität München, Munich, Germany, 2009.

[Tet90] Tetlow, W. L.: Selecting appropriate computing tools. In *New Directions for Institutional Research*, Wiley Periodicals, NJ, USA, 1990, P. 71-80.

[TF84] Tipton, C. M.; Fisher, J. M.: An outline of the essentials of journal reviewing for beginning investigators. In *Sports Medicine Bulletin*, 1984.

[Tro06] Trochim, W. M.: Descriptive Statistics. In Trochim W. M.; Donnelly J. P. *Research Methods Knowledge Base*; Atomic Dog Publishing, Mason, OH, USA, 2006.

[Veg02] Vegas, S.: *Characterisation Schema for Selecting Software Testing Techniques*; Technical University of Madrid, Madrid, Spain, 2002.

[Wes01] West, D. B.: *Introduction to graph theory*; Prentice Hall, NJ, USA, 2001.

[Yin03] Yin, R. K.: *Case Study Research: Design and Methods*; Sage Publications, London, England, 2003.

[Zah65] Zahed, L. A.: Fuzzy sets. In *Information and Control*, 8, Elsevier, Amsterdam, Netherlands, 1965, P. 338-353.

Appendix A – Literature Review

This chapter provides some additional data of the systematic literature review in Section 3.1. Because this review was a part of the state-of-the-art chapter we do not want to provide that much data in there. The tables and data of this appendix are provided to get further details if necessary.

As it was already mentioned in Section 3.1 the number of papers in literature decreases in each stage of the selection process (cf. Figure 6) so that at the end there were 1,81% of the initial papers. How the number of papers decreases is illustrated in Table 17:

Table 17. Number of Papers in the different stages in the databases

Libraries	**Stage 1**	**Stage 2**		**Stage 3**		**Stage 4**	
ACM	93	20	21,5%	9	45,0%	4	44,4%
IEEE	1006	63	6,3%	24	38,1%	11	45,8%
SpringerLink	113	29	25,7%	11	37,9%	5	45,5%
ScienceDirect	227	38	16,7%	10	26,3%	6	60,0%
Overall	**1439**	**150**	10,4%	**54**	36,0%	**26**	48,1%
(elim. of dupl.)		*146*		*51*		*23*	

In contrast to Figure 6 the Table 17 also provides data about the different libraries over the stages of the selection process. This is important to see the change during the stages of the different libraries. For example the IEEE Xplore library uses 6% (stage 2) of the previous 1006 papers, which is a big difference to the others.

In contrast to this analysis the following are more related to the number of papers in a specific context, e.g. the time, the topic or the domain.

Table 18. Number of Papers sorted by publication year

year	**number**
2000	0
2001	1
2002	0
2003	0
2004	2
2005	2
2006	1
2007	3
2008	4
2009	4
2010	3
2011	3

Table 18 provides the data for Figure 7 in the literature review. Therefore, we went through all the resulting papers of the last stage of the selection process and sorted them based on the *publication year*.

Table 19. Number of Papers sorted by conferences- and journal topics

Topics	Number
Information & Technology	4
Systems	4
Empirical SE	2
Power Engineering	2
Communications	2
Management of Engineering and Technology	2
Computers helping people	1
Software Quality	1
Environmental Modeling & Software	1
Software Process Improvement	1
Computational Science and Its Applications	1
Design, Implementation, and Use	1
Biomedical informatics	1

In contrast to the previous table, Table 19 deals with the *topics of the conferences* and *journals* the papers were published. This is important to get to know the most important conferences and journals for our search. This is not that easy because a big number of the method platforms were published not directly in the computer science or SE domain but in their application domain. Table 19 illustrated that there is a set of topics used more than only once.

Table 20. Number of Papers sorted by domains

UsedIn / UsedFor	number
SE	6
Testing	*3*
Ver.&Val.	*1*
Architecture	*1*
CS	3
BioInfomatic	*1*
MobileComerce	*1*
Manufacturing	3
Economy	2
Electrical Eng.	2
Health Care	2
Ecology	2
Human Resource	1
Telecommunication	1

The next part of the analysis deals with the *domains* the method platform is used in. Table 20 depicts the number of papers in the different domains or subdomains (cf. Table 20, grey rows). The most important information for us is that a big number is in the SE or computer science domain. In addition it is positive to see that at least some phases of the SE life-cycle are included.

Table 21. Number of Papers sorted by input, output and decision-making

Input	**number**
Technology	7
Context	6
Problem/Goal	5
Defects	1
DB	1
Boundary Condition	1
User Preferences	1
Req.	1
Output	**number**
Technology	12
Matrix	1
Selection	1
Scenario	1
Comb. of Technologies	1
Ranked Technology List	1
Tool	1
Decision-making	**number**
Fuzzy logic	8
(Multi-) Rule based	3
Probability	2
Porantim Selection Strat.	2
TOPSIS	2
DEA model	1
Knowledgebase	1
Pairwise comparison	1
Bound and free var.	1
User questions	1
Dependency relation	1
Integrity constrains	1
Neutral networks	1
CMOS logic / SRAM	1

The last part of the detailed analysis of this appendix deals with the different aspects of the method platform definition of Section 3.1. Therefore, we need to check which Input, Output and Decision-making is used in the publications. Those three aspects are all shown in the different parts of Table 21. In the input and output part there is an accumulation on a small number. Those possibilities are then already used in our conclusion (cf. Section 3.1.4) of the literature review. The more difficult part is the decision-making. This is the case because Table 21 shows that the publications do not use the separation of Section 2.1. For example rule based decision making is a possibility to implement other decision theories, e.g. fuzzy logic or knowledge based.

Appendix B – XML Technology Template

This part of the appendix contains the details of the new adapted categorization schema based on [Jed09]. The attributes and elements described in Chapter 5 will now be modeled in XML [BPS06] to get a more formal possibility to describe and refine them. This is done for all the three models of the Categorization chapter (cf. Chapter 5.1) in this thesis (technology, context and input model). Here we provide the specifications as XML schemas (DTD[16]s [BPS06] or XSD[17]s [HKS02]) as well as some digital examples.

All the defined technologies used within this framework need to be specified in XML files which need to be valid to the respective schemas (DTD and XSD). As the framework will probably be implemented in the programming language Java the respective technology for accessing XML by binding it to Java types (e.g. XMLBeans [Apa03]) needs to check if it is well-formed and valid.

For later implementation reasons (cf. Appendix C) of the PCF the XML schema definitions were used to create JARs[18] for each of the needed models. This needed for accessing XML by binding it to Java types and is done with [Apa08]. The JARs will only be attached to this thesis as digital files, because of simplicity reasons.

[16] Document Type Definition
[17] W3C XML Schema Definition – Advice for defining the structure of XML documents
[18] Java ARchive

B.1 Technology Model

This section provides an overview of the full XML schema of the technology model in two different ways. They are depicted in a graphical representation and a textual representation. Since the textual representation of the model is very complex and large it is only included in the digital attachment on the accompanying CD.

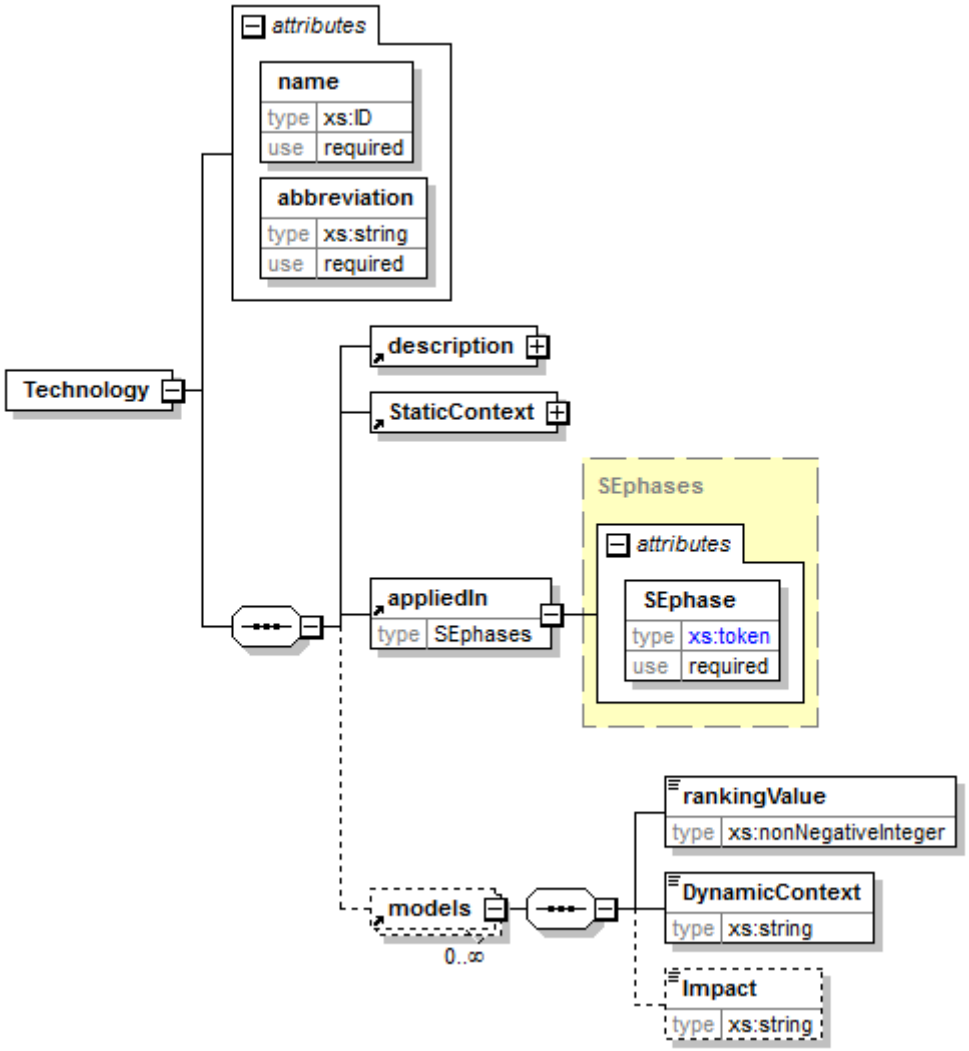

Figure 27. XSL Schema of the technology model (starting at the *Technology* attribute)

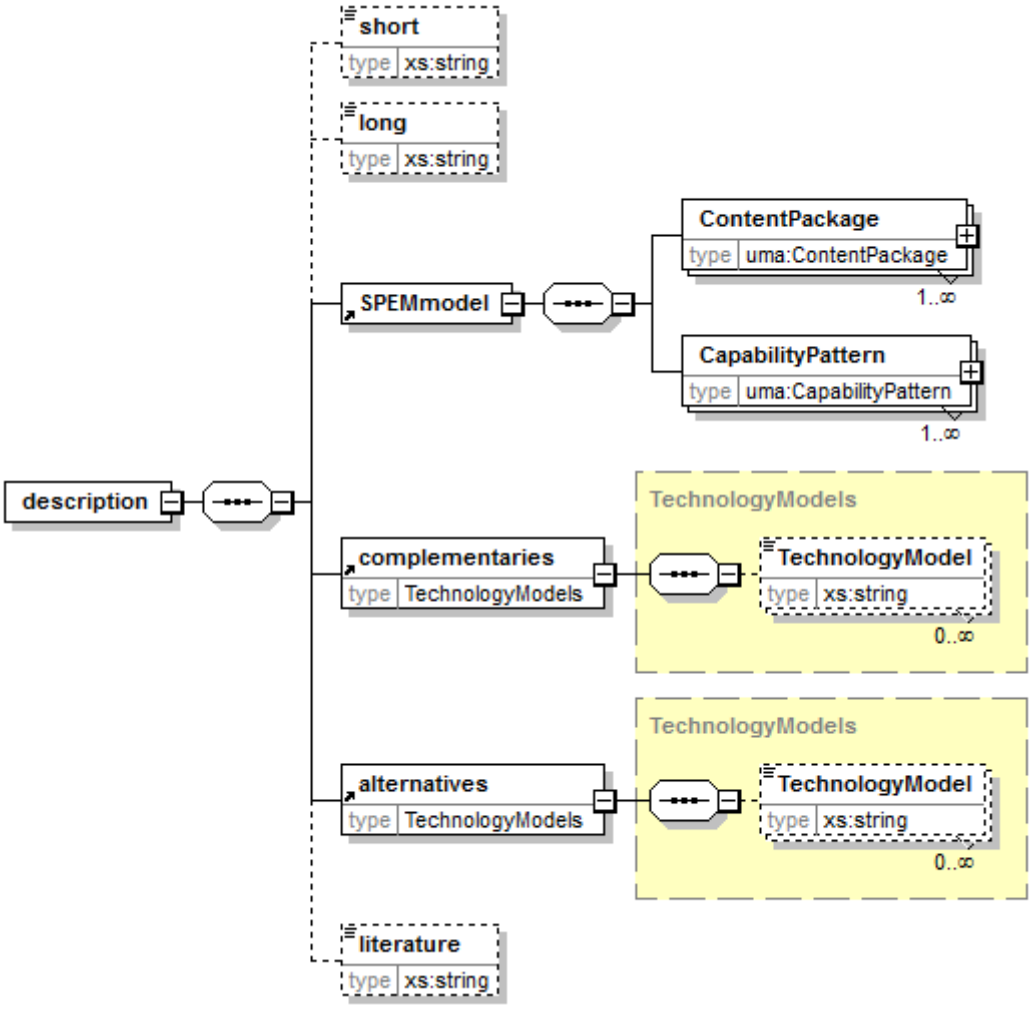

Figure 28. XSL Schema of the technology model (starting at the *Description* attribute)

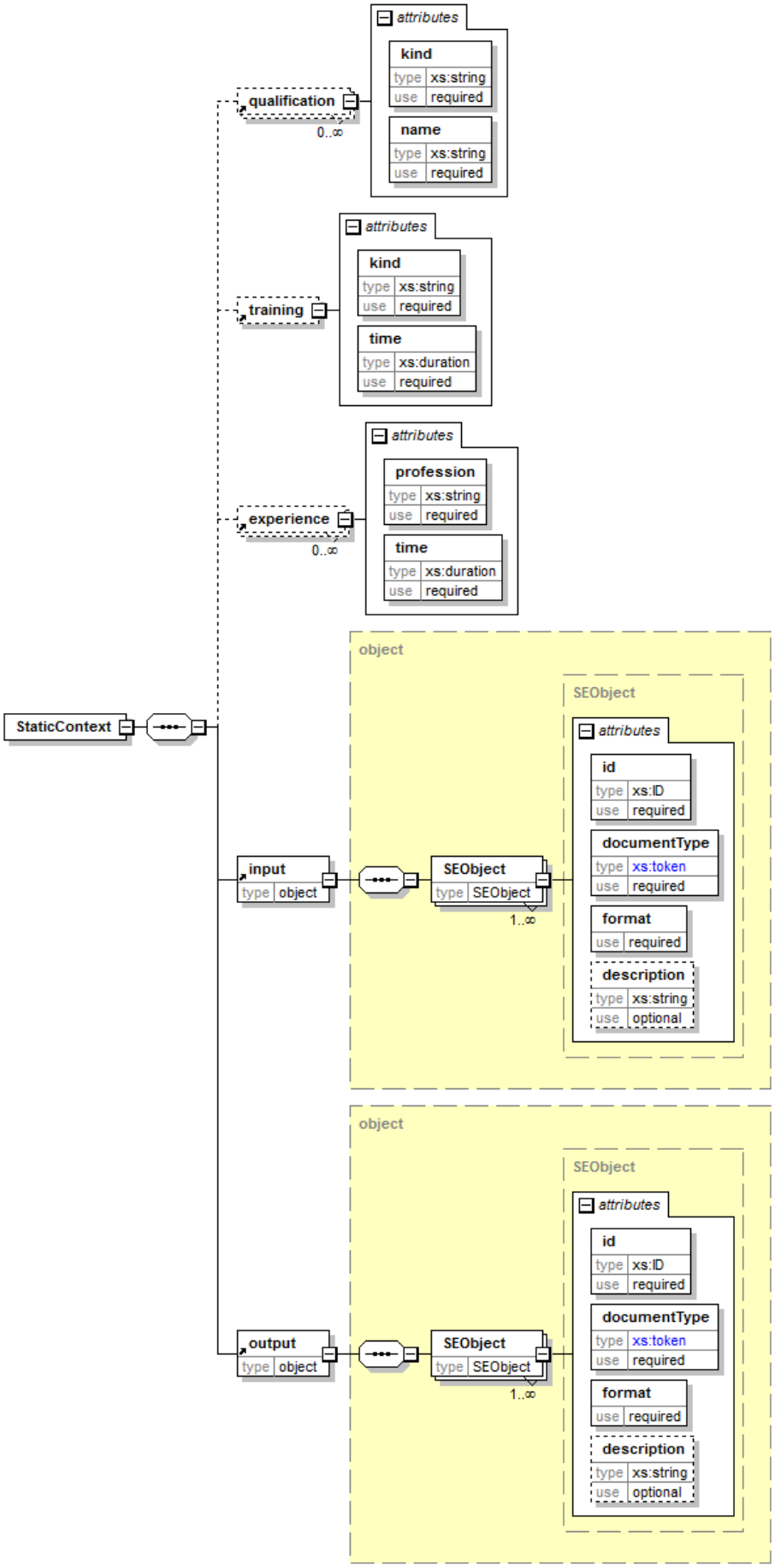

Figure 29. XSL Schema of the technology model (starting at the *Static Context* attribute)

B.2 Context Model

This section provides an overview of the full XML schema of the context model in two different ways. They are depicted in a graphical representation and a textual representation. Since the textual representation of the model is very complex and large it is only included in the digital attachment on the accompanying CD.

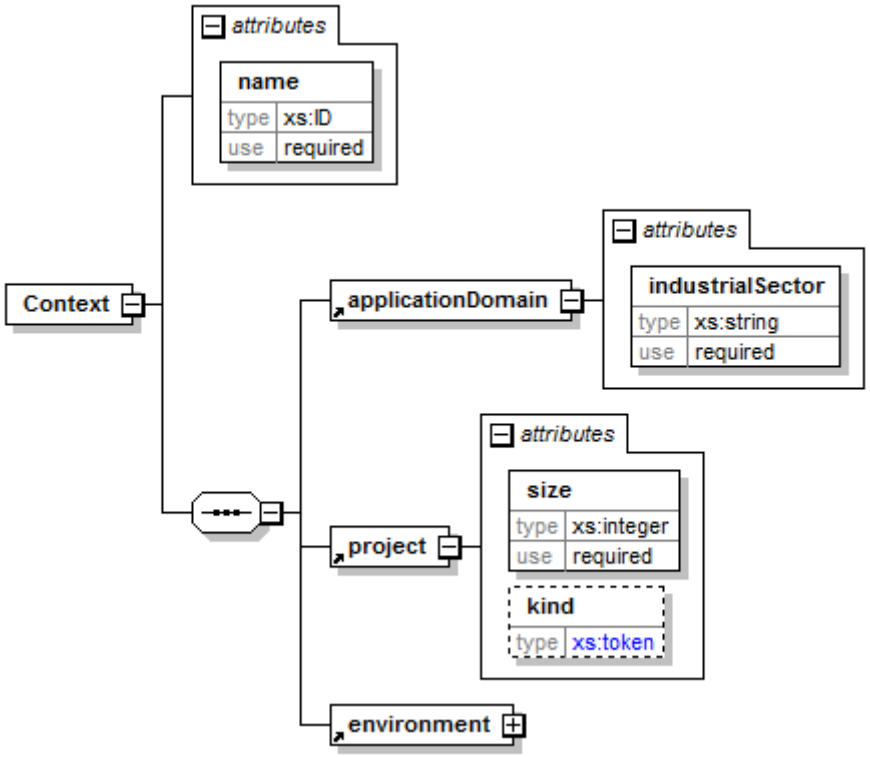

Figure 30. XSL Schema of the context model (starting at the *Context* attribute)

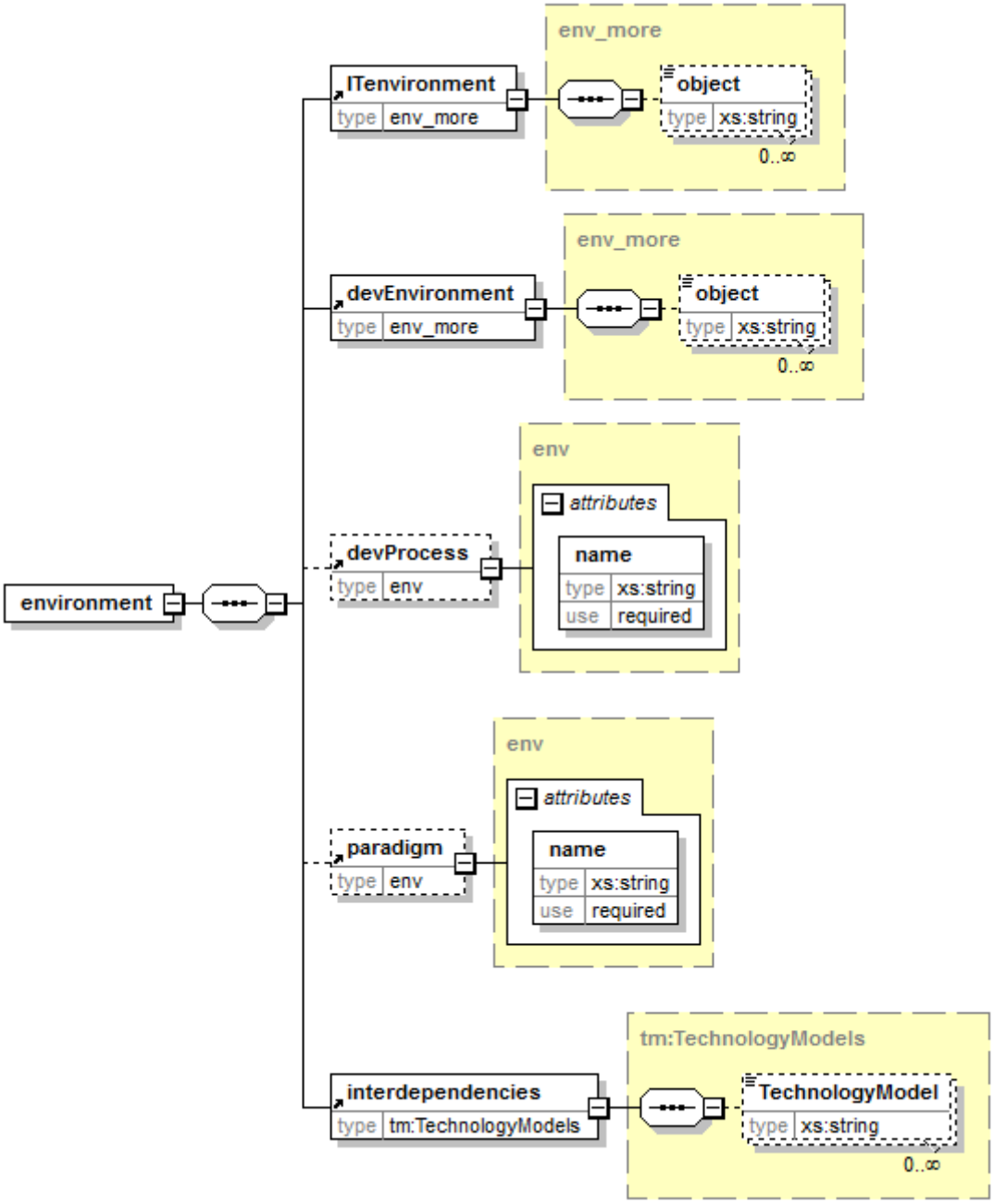

Figure 31. XSL Schema of the context model (starting at the *Environment* attribute)

B.3 Impact Model

This section provides an overview of the full XML schema of the impact model in two different ways. They are depicted in a graphical representation and a textual representation. Since the textual representation of the model is very complex and large it is only included in the digital attachment on the accompanying CD.

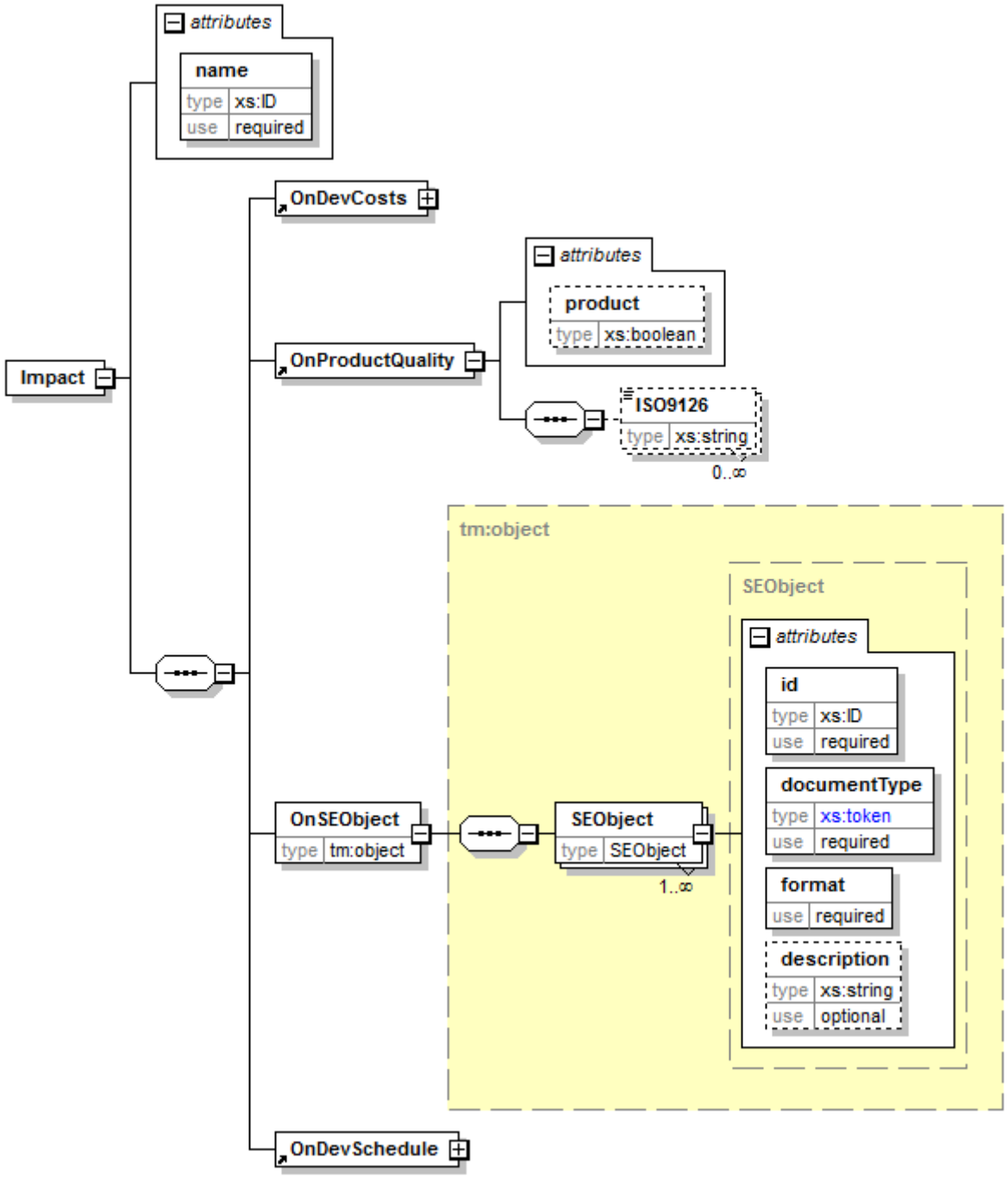

Figure 32. XSL Schema of the impact model (starting at the *Impact* attribute)

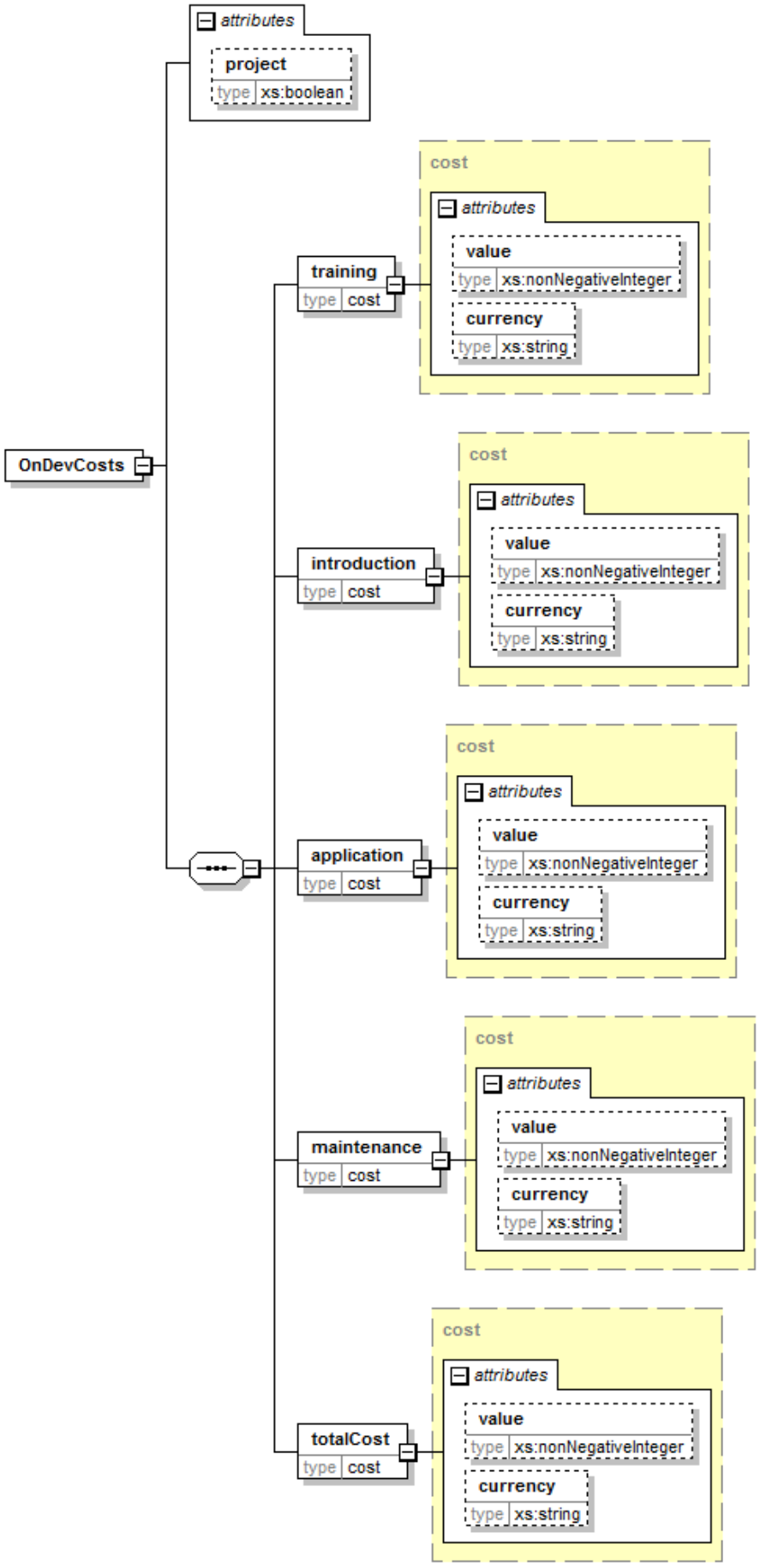

Figure 33. XSL Schema of the impact model (starting at the *OnDevelopmentCosts* attribute)

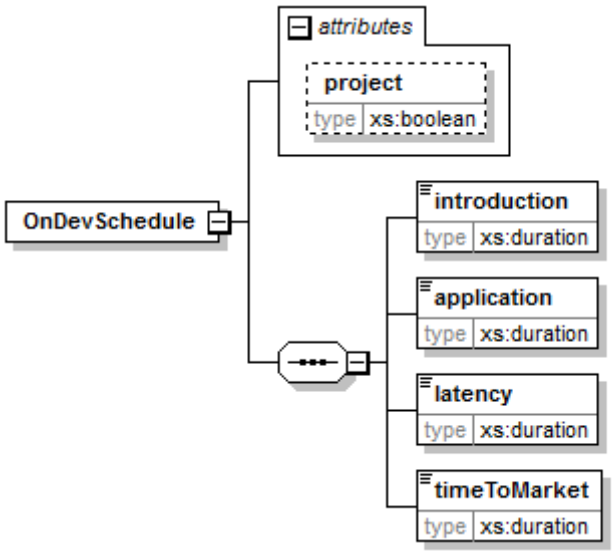

Figure 34. XSL Schema of the impact model (starting at the *OnDevelopmentSchedule* attribute)

Appendix C – Technical Realization Aspects

This appendix section contains the technical aspects of the realization of the PCF. Although the technical realization was not planned as part of this thesis, parts if it had been implemented for a precise specification of the rules used in the framework.

The architecture of it, depicted in Figure 35, contains several technological aspects needed for the realization. Those aspects range from XML database to the Drools rule engine and will be shortly explained in the next paragraphs.

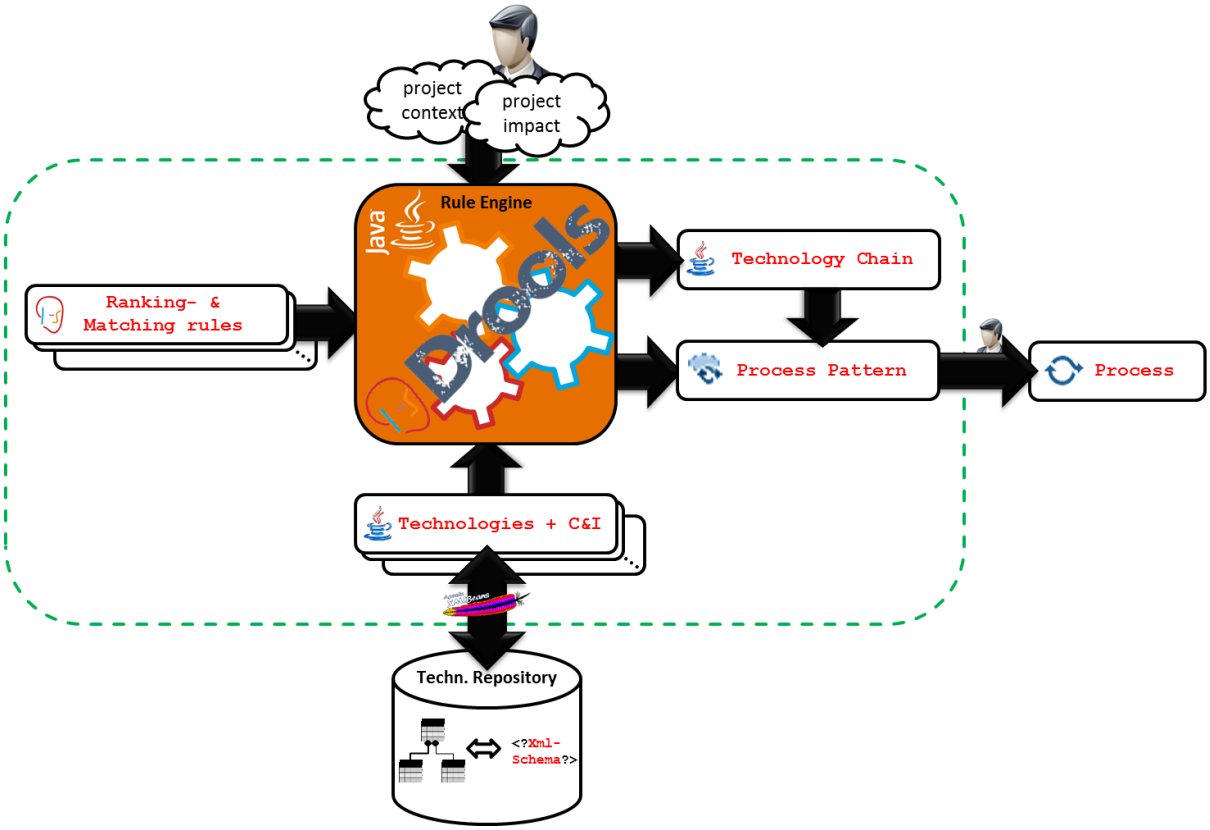

Figure 35. Process Configuration Framework Architecture

The foundation of the framework is the technology repository, already introduced in Chapter 5. This repository contains all the technologies and their respective contexts and impacts. It is an XML database and those models are stored as XML files, because we need the flexibility and the tree structure of XML for the framework. The XML files are based on the XML schemas given in Appendix B.

The usage of XML is needed, because of the later following rule engine which requires Java objects to work on. XML Beans [Apa03] seems to be the best way to realize this, because it transfers the technologies, specified in the repository, into Java objects (cf. Figure 35, "Technologies + C&I"). From this step on the framework works on the Java objects for performance reasons.

Now the rule engine Drools [JBo12] gets important, which is the main part of the technical architecture of the framework. This java rule engine takes the input of the user, which can only be a mixture of projects' context or input. Together with the input the rules are applied by the engine on the Java objects.

The output of the process configuration framework contains two parts, the Technology Chain and the corresponding process pattern ( , parts of SPEM). In the static approach the Technology Chain contains one technology for each SE phases (and alternatives of those technologies). The patterns provide the activity- / workflow diagram for those technologies. Those process patterns are created for each technology of the chain based on the SPEM model (attribute of the technology model).

The last technical aspect is the manual configuration of the process, which is not quite part of the framework. This part needs to be performed manually because SPEM is based on the RUP [EM03]. Furthermore, the different companies and users make use of different kinds of processes as well as iteration steps. The creation of a process (regardless of whether using the RUP or not) is a very easy step for the user, because of the existing process pattern for the different phases. They only need to be plugged together in a company-/user-specific way. For example this task can be performed by using the EPF composer, which can also be used for publishing processes.

In addition to this technical structure of the framework, there was the idea of using this framework as a web service. This came up during the development of the framework because the best way would be a global technology repository with worldwide access. This would hopefully increase the number of technologies in the database which would lead to a more precise output of the framework. A possible solution for this idea would be the usage of web services combined with java applets or servlets as web frontend.

Concluding this chapter about the technical structure of the process configuration framework a huge number of different existing technical approaches and techniques would be helpful to use. With those techniques the implementation the framework will not expend much effort because big parts will be provided by the techniques, e.g. the Drools rule engine.

Appendix D – Evaluation

This subpart of the appendix includes more detailed information about the evaluation. This means we are presenting parts of the information which is needed to understand the results and discussions in Chapter 10 or to repeat the case study. The additional information for the evaluation which is not possible to present in the appendix, because of complexity or size are included in the digital appendix on the CD at the back of this thesis.

D.1 Scenario

Within this section we provide a more detailed description of the scenario, which was briefly mentioned in the design section of the evaluation (cf. Section 0). There it was only explained in a textual way and the Table 22 has been referenced. This table was exactly the scenario given to the experts during the case study.

Table 22. Software Project Scenario for the Evaluation

Attribute	Value
Industrial Sector	Automotive
Qualification[19]	• ISEB Certificate (for RE) • UML 2 – OMG certificate
Training[20]	• Advances MDA
Experience[21]	Req Engineer (2y), Software Architect (1y), Software Engineer (15y)
Project Size	25 Person Months
Project Kind	Embedded System
IT environment	DOORS, Enterprise Architect, MS Office (Word, Excel, Visio, …)
Development environment	C++, MCV-architecture
Development Process	V-Model XT
Paradigm	ObjectOriented (OO)
ISO 9126	• Reliability (Maturity, Fault Tolerance) • Functionality

In addition to this table we also specified some other important information for the experts because the first participant asked us about that information. For comparison reasons we then included the information for the others so that they are all working at the same scenario.

The additional general information of this project are the importance of *maintenance* in future and that it could be seen as a *mainstream project* which means that similar projects are performed more than once in future. Another important aspect is the customer itself, because it could be important for the project whether it is an *existing* or a new customer. In addition to the previous mentioned information now the more project specific details are discussed. The overall focus of this software project is on

[19] Describes the qualifications of the employees of your company
[20] Describes the possible trainings performed by the employees of your company. (y=year)
[21] Describes the experience of the employees of your company

safety, as it could also be seen from the IASO 9126-attribute in Table 22. Another important aspect is a more detailed description of the project kind. Most of our experts during the case study wanted to get to know a detailed key functionality to develop. In our case we decided to use a *motor control unit*. At last the detailed scenario description includes a list of technologies that have already been used: In our scenario this are the *Use Case Specification* as well as the both *UML technologies* for architecture and design.

D.2 Configuration Templates

In addition to the scenario explained in the previous section, now this section provides the two templates for the configuration of the Process Configuration Framework. They are illustrated in Table 23 and Table 24 as they were provided to the participants of the evaluation. With these filled configurations (cf. Table 13) it was then possible to perform the PCF and get the automatically configures Technology Chains.

Table 23. PCF Configuration Template 1

Category		Element	
Combination Value	**___%**	Interface (I/O)	**___%**
		Complement	**___%**
		Paradigm	**___%**
		Dev. Process	**___%**
Ranking Value	**___%**		

Table 23 is used for the differentiation between the combination and ranking value. In addition to this the importance values of the different elements for the combination also need to be specified in this template.

Table 24. PCF Configuration Template 2

Category		Element		Attribute	
Context	**___%**	Application Domain	**___%**	Industrial Sector	**100%**
		Project	**___%**	Size	**___%**
				Kind	**___%**
		Environment	**___%**	IT Environment	**___%**
				Dev.Environment	**___%**
				Development Process	**___%**
				Paradigm	**___%**
		Prerequisites / Static Context	**___%**	Qualification	**___%**
				Training	**___%**
				Experience	**___%**
Impact	**___%**	Quality	**___%**	ISO 9126	**100%**
		Development Costs	**___%**	Total Cost	**100%**
		Development Schedule	**___%**	Time to Market	**100%**

In contrast to the previous table, Table 24 includes all elements and attributes for the ranking of the technologies based on the input model (cf. Section 0). With this table the experts should give their opinion how important the different attributes of the input models are.

D.3 Raw Data

This last section of the evaluation appendix includes more tables with additional data that is not presented in the results section (cf. Section 10.3), because of the size of the data set. The most important aspect of this section is the ranking of all 27 possible Technology Chains in all five used configurations of the PCF in Table 26. For this table we needed Table 25 as explanation because there the possible chains are numerated.

Table 25. List of all 27 Technology Chains

TC	Technologies		
1	**RBE**	**UML**	**UMLD**
2	RBE	UML	AD
3	RBE	UML	PPM
4	RBE	MDA	UMLD
5	RBE	MDA	AD
6	RBE	MDA	PPM
7	RBE	EA	UMLD
8	RBE	EA	AD
9	RBE	EA	PPM
10	UCS	UML	UMLD
11	UCS	UML	AD
12	*UCS*	*UML*	*PPM*
13	**UCS**	**MDA**	**UMLD**
14	UCS	MDA	AD
15	*UCS*	*MDA*	*PPM*
16	UCS	EA	UMLD
17	UCS	EA	AD
18	*UCS*	*EA*	*PPM*
19	**RA**	**UML**	**UMLD**
20	RA	UML	AD
21	*RA*	*UML*	*PPM*
22	RA	MDA	UMLD
23	RA	MDA	AD
24	*RA*	*MDA*	*PPM*
25	RA	EA	UMLD
26	RA	EA	AD
27	*RA*	*EA*	*PPM*

Table 25 is needed for the explanation of the following table. It has been introduced to reduce complexity and not to mention each chain every time, by repeating all the three technologies building one chain.

Table 26. All 27 Technology Chains ranked based on the different configurations

TC	TC Stand.	TC	TC Exp.1	TC	TC Exp.2	TC	TC Exp.3	TC	TC Exp.4
10	84,1%	**1**	**72,5%**	**1**	**77,8%**	10	87,2%	10	97,2%
1	**80,2%**	10	71,9%	10	74,1%	**1**	**79,8%**	**1**	**82,9%**
13	**73,3%**	**13**	**68,0%**	**13**	**69,7%**	**13**	**75,9%**	**19**	**72,9%**
4	70,6%	16	61,8%	*18*	*67,0%*	**19**	**73,4%**	**13**	**63,3%**
19	**67,6%**	**19**	**61,1%**	16	63,9%	16	72,9%	4	57,2%
11	64,3%	22	59,0%	2	63,8%	4	69,3%	*12*	*56,0%*
2	63,8%	4	58,5%	4	63,8%	7	67,0%	11	55,5%
12	*63,8%*	2	58,4%	**19**	**62,8%**	22	64,0%	2	51,7%
16	63,2%	11	57,9%	11	62,6%	25	62,9%	16	51,3%
14	61,0%	7	57,5%	22	61,3%	11	62,1%	7	51,0%
7	61,0%	*18*	*54,7%*	14	61,1%	*12*	*60,4%*	*18*	*51,0%*
5	60,9%	25	54,6%	*15*	*60,2%*	*18*	*59,7%*	9	50,7%
22	60,5%	14	54,3%	7	60,0%	14	56,7%	5	49,3%
15	*59,6%*	*15*	*54,1%*	*27*	*59,8%*	17	56,3%	3	47,6%
3	59,0%	*12*	*52,0%*	25	58,4%	*15*	*55,7%*	14	47,3%
18	*57,1%*	3	51,9%	5	58,1%	2	53,8%	22	47,2%
6	56,1%	5	50,3%	17	57,4%	3	52,6%	*15*	*45,5%*
25	54,3%	20	49,1%	9	56,9%	9	51,8%	6	45,2%
17	53,9%	9	48,1%	*12*	*56,4%*	20	50,5%	25	43,3%
20	53,7%	*27*	*46,8%*	3	55,9%	*27*	*49,2%*	8	43,1%
8	52,6%	23	46,8%	20	55,1%	5	49,0%	20	41,6%
9	52,3%	*24*	*46,3%*	23	54,8%	6	48,9%	17	41,1%
23	49,9%	17	45,8%	8	54,4%	8	48,7%	*27*	*38,3%*
21	*48,9%*	8	45,6%	*24*	*53,1%*	*21*	*48,3%*	*21*	*37,5%*
24	*46,3%*	6	43,8%	26	51,1%	23	46,4%	23	36,9%
27	*43,8%*	*21*	*42,7%*	6	50,2%	26	46,1%	*24*	*32,8%*
26	42,8%	26	38,3%	*21*	*47,2%*	*24*	*45,2%*	26	30,8%

Table 26 is the most important table of the evaluation results. Nonetheless because of complexity and size reasons some other tables are derived from this, e.g. Table 15. It is the most important one because it includes the ranking of all 27 Technology Chains with all the five used configurations. The percentage values are the Technology Chain values of them. The colored background of the cells shows the quality of the chain, because the higher values have a green and the lower values a red background. In addition to this the top/worst five chains are bordered with green/red. The grey and italic marked chains can be seen as excluded ones because they include interdependent technologies and would be excluded before the ranking by Technology Chain Restriction (cf. Section 7.3). However, for this case study we marked those chains. In contrast to this the bold numbers represent the three manually selected Technology Chains.

The whole calculation of the Technology Chain values shown in Table 26 was done on a huge number of intermediary values, e.g., ranking values of the different technologies or combination values. All these intermediary values from the lowest element level to the overall Technology Chain values can be seen in the Excel sheet in the digital appendix.