ulm university universität ulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme
Leiter: Prof. Dr. Manfred Reichert

Context-aware Process Management for the Software Engineering Domain

Dissertation zur Erlangung des Doktorgrades Dr. rer. nat.
der Fakultät für Ingenieurwissenschaften, Informatik und Psychologie der Universität Ulm

Vorgelegt von:
Gregor Robert Grambow
geboren in Schwäbisch Hall

2016

Amtierende Dekanin: Professor Tina Seufert
Gutachter: Professor Manfred Reichert
Professor Franz Schweiggert
Professor Roy Oberhauser
Tag der Promotion: 13.05.2016

Preface

The results presented in this thesis are the outcome of my work as a research assistant at the Institute of Databases and Information Systems of Ulm University between 2012-2015 with Prof. Dr. Manfred Reichert, as well as at the Computer Science Department of Aalen University of Applied Sciences between 2009-2012 working with Prof. Roy Oberhauser on the Q-ADVICE project (Quality ADVisory Infrastructure for Collaborative Engineering), where we created the Context-aware Software Engineering Environment Event-driven frameworK (CoSEEEK).

I am very grateful to my mentor Prof. Dr. Manfred Reichert, whom I thank for his helpful advice, continuous support, and valuable feedback during the last years. He provided all the freedom needed for my research, but at the same time was always there to support me.

In addition, I am deeply grateful to my mentor Prof. Roy Oberhauser. This thesis would not have been possible without his initiative and extensive help and support, and I thank him for being a part of it. I could go to him with any problem or worry. He was a motivation and inspiration to me: not only his passion for science, but also his passion to really improve the situation for practitioners in software engineering, as well as his creativity and innovative ideas.

Besides my mentors, I would like to thank my second proofreader Prof. Dr. Franz Schweiggert and the members of my doctoral committee Prof. Dr. Peter Dadam, Prof. Dr. Helmuth Partsch, and Prof. Dr. Uwe Schöning for providing feedback for my thesis.

My thanks extend to my colleagues and students from the Q-ADVICE project for their support and help with the implementation of our concepts: Stefan Lorenz, Andreas Kleiner, Muhammer Tüfekci, Andreas Nägeli, and Alexander Grünwald.

I would also like to thank my colleagues and friends from Ulm University. It was a great time at DBIS, sharing so many different tasks with you.

Personally, I also want to thank Susanne for her help and continuous support in every situation. Her care and support during most of my thesis time made this the best time of my life, despite my spending so much time and attention on this project. She stood by me, and I am really looking forward to our future together.

Last but not least, I would like to thank my parents Heiderose and Peter Grambow. They were always there for me during the past decades, and always supported me with good advice in every situation. They enabled my studies in Aalen, Karlsruhe, and Ulm. For all of these things, and many more, I am deeply thankful.

Abstract

Historically, software development projects are challenged with problems concerning budgets, deadlines and the quality of the produced software. Such problems have various causes like the high number of unplanned activities and the operational dynamics present in this domain. Most activities are knowledge-intensive and require collaboration of various actors. Additionally, the produced software is intangible and therefore difficult to measure. Thus, software producers are often insufficiently aware of the state of their source code, while suitable software quality measures are often applied too late in the project lifecycle, if at all.

Software development processes are used by the majority of software companies to ensure the quality and reproducibility of their development endeavors. Typically, these processes are abstractly defined utilizing process models. However, they still need to be interpreted by individuals and be manually executed, resulting in governance and compliance issues. The environment is sufficiently dynamic that unforeseen situations can occur due to various events, leading to potential aberrations and process governance issues. Furthermore, as process models are implemented manually without automation support, they impose additional work for the executing humans. Their advantages often remain hidden as aligning the planned process with reality is cumbersome.

In response to these problems, this thesis contributes the *Context-aware Process Management (CPM)* framework. The latter enables holistic and automated support for software engineering projects and their processes. In particular, it provides concepts for extending process management technology to support software engineering process models in their entirety. Furthermore, CPM contributes an approach to integrate the enactment of the process models better with the real-world process by introducing a set of contextual extensions. Various events occurring in the course of the projects can be utilized to improve process support and activities outside the realm of the process models can be covered. That way, the continuously growing divide between the plan and reality that often occurs in software engineering projects can be avoided. Finally, the CPM framework comprises facilities to better connect the software engineering process with other important aspects and areas of software engineering projects. This includes automated process-oriented support for software quality management or software engineering knowledge management. The CPM framework has been validated by a prototypical implementation, various sophisticated scenarios, and its practical application at two software companies.

Contents

Part I

Problem Statement and Requirements

1. Introduction

Software Engineering (SE) is a discipline that implies special properties for process enactment. On one hand, these are correlated with the special properties of the produced product, i.e., the software: complexity, conformity, changeability, and invisibility [Broo87]. On the other, IT support for SE processes is not mature yet, since SE implies a highly dynamic and creative process. Furthermore, the impact of process management in SE has been underestimated for a long time [Wall07]. Over decades, many SE process models as well as models for SE process improvement have been developed and been introduced to practice. In other areas, like industrial production, such processes have been automated and supported by process management technology [LeRo00]. Yet implementation and automated enactment of processes is not prevalent in SE, mostly due to the dynamic nature of these knowledge-intensive processes that contradict the rigid sequencing of process activities necessary for an automated enactment.

SE processes are essentially knowledge-intensive, i.e., they depend on knowledge workers to a large extend [KeHa02]. The highly intellectual SE process implies a high amount of communication. Compared to industrial production processes, SE processes rely much more on humans and highly collaborative team interactions. Note that each SE project constitutes a development project, producing a unique outcome. For such projects, the rigidity of prescribed processes mostly does not fit. Further, it was already stated that dynamic processes supporting collaboration as well as communication can be beneficial [Shet97]. Usually, SE processes deal with the development of a new product (i.e., the software), which is a knowledge-intensive task [RaTi99]. In this context, necessary facts, much information and comprehensive knowledge are handled manually and implicitly by the humans involved. Hence, automation is not feasible and SE processes are usually performed manually in a documentation-centric way [RBTK05]. In turn, this often implies high manual efforts for humans as they have to manage the process models. Moreover, actual process enactment largely depends on humans. Many tasks and activities are not part of the process models. Especially on the operational level, where activities like coding and testing are performed, only limited support for the software engineers is available from SE process models. Thus, there is a growing gap between the specified process and the one actually executed.

Another specialty of SE projects is the product developed. Software has special properties whose combination differentiates it from many other products: complexity, conformity, changeability, and invisibility [Broo87]. These properties make it difficult to be aware of the status of the software. In many software producing companies, human tasks, requirements, and the realization of the requirements are managed in some way. However, due to the often high number of humans working concurrently on numerous source code artifacts, the quality of the source code can deteriorate unnoticed. Thus, many projects struggle with bad source code quality [Jone10]. However, to be enforced, software quality must be defined and measured [Kan02]. In many cases, resources are wasted by neglecting software quality issues and respective software quality measures until the final stages of a project [Hami88, SHK98]. Another issue of software quality comes with the lacking ability of many companies to actually control, manage and support their knowledge-intensive and human-centric processes [BDS+99, Ambl02, Wall07, Dust04, SBBK08]. Due to these issues, projects suffer from bad software quality as well as exceeded budgets and deadlines having issues on both the process and the product side. Altogether, it is desirable to integrate software quality assurance tightly and smoothly with process management to enable the continuous monitoring of product quality.

Business Process Management and the introduction of Process-Aware Information Systems (PAIS) has been a continuous trend in various business areas. In particular, the explicit governance of activities by a PAIS enables improved repeatability of the process and can thus improve the quality of the product [ReWe12, DAH05]. Domains in which PAIS have been successfully introduced include

health care [LeRe07], automotive engineering [MHHR06, MRH08], finance [GoAk03] and transportation [Bass05]. To be able to comprehensively cover all activities executed as well as optimize the whole process executed in an organization, the business process lifecycle [GeTs98, vdAa04, WRWR09] is roughly separated into several phases (cf. Figure 1-1): First the *process* is defined, which implies a *design* process. Further, this phase might include the discovery of executed processes through process mining [vdAa11, vdWe04, vWM04]. Following the *design phase*, the process is *implemented*. In the subsequent *enactment phase*, the process is used to govern the activities it was designed for. Data from this enactment phase is then used for the *diagnosis phase*, which can be applied to optimize the process as well as to adapt it to environmental changes. With the results from the diagnosis phase the cycle can be restarted.

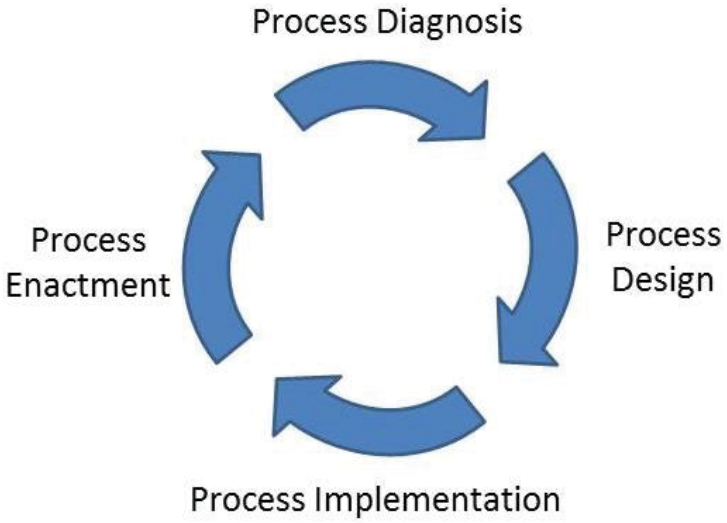

Figure 1-1: Process lifecycle (adopted from [vdAa04])

To enable continuous support and guidance for a process, automated IT support is desirable. To achieve the latter, processes can be implemented using PAIS [vdvH02]. Such systems provide support for automated process enactment, automated task distribution to humans, coordination, and monitoring of different process instances. That way, process enactment can be guided and process diagnosis be supported, since the executed activities are explicitly governed. This makes the entire process enactment more traceable and repeatable.

The described factors hamper successful process enactment in the SE domain. Many of the issues discussed, however, are related to the dynamics of SE projects [BDS+99, Ambl02]. In fact, numerous obstacles inhibit automated SE process management (SEPM) at the operational level. These include a high number of dynamically executed small tasks like bug fixing, coding, developer tests, or integration tests. Respective tasks may not even be covered on the more abstract planning levels where the entire project, its process, and different phases are managed. Such activities also imply many contextual dependencies, i.e., they rely on properties relating to the current situation, e.g., time pressure in the projects or technology used. Another factor is the great number of involved artifacts, e.g., documentation artifacts, specifications, or the source code itself. These artifacts often have many relations with each other and are frequently changed by various persons. This involves a great amount of tacit knowledge crucial to the projects that is only implicitly managed by these persons. In turn, this puts high pressure on them: Because of the high dynamicity, the concurrent enactment of multiple projects, the absence of clearly defined and stable requirements, and many other factors, much is left to them. This constitutes a great burden as well as high efforts for software engineers. Due to the lack of repeatability and guidance of these knowledge processes, it is rather likely that the knowledge worker forgets important tasks or unintentionally introduces new problems to the source code.

1.1. Problem Statement

As shown by many studies, SE projects have been suffering from problems with exceeded budgets, missed schedules, and low product quality for a long time [NaRa68, Broo87, Glas98, Kruc04, Jone10]. Many of these problems are resulting from the adolescence of SE as a discipline and special properties of this discipline having a great impact on SE projects. These properties (e.g., the intangible product or the knowledge-intensive, human-centric SE process) are exhibited, in both the created product and in the SE process. Based on this, three main topics introducing serious issues to SE can be observed: First, the knowledge-intensive process puts much pressure on the involved humans. Second, the intangible product makes it difficult to control the latter and might introduce severe quality issues. Looking at these two problematic sides of SE projects, a third topic comes into mind: there exist tools that support various SE aspects but no comprehensive and automatic process support, incorporating humans and artifacts, is prevalent.

Note that we will use the term process and workflow with different meanings. Process will be referred to as something rather abstract that is not implemented in software. Workflow will be referred to as something more concrete and operational as well as something that is implemented using a software tool and, therefore, as an automated facility to govern the flow of activities.

Manual process implementation. Process automation was mainly applied in areas in which foreknown activity sequences exist, but not in scenarios requiring the enactment of a human-centric and knowledge-intensive process [MBR15]. In SE, therefore, there exists not much experience with process automation. Process models are available containing information important for the projects [BWHW06, RiJa00, Mall09]. However, these remain rather abstract and prescriptive [BDS+99, Ambl02]. Hence, manual implementation becomes necessary. Consequently, the involved persons are responsible for enacting the SE process without automated governance or enforcement. This implies shortcomings with respect to guidance, traceability, monitoring, and diagnosis of the activities executed, as abstract process models mostly do not reach the actual executing persons [Wall07]. In particular, they tend to fail in providing operational guidance. Since the quality of the software product is depending on the quality of the SE process [Wall07], this affects product quality as well. The gap between the abstract process models and the actual executed activities also prevents comprehensive coverage of all activities in the SE process. Many activities are executed ad-hoc and cannot be traced. However, if many are activities executed outside the SE process, knowledge about actual process enactment cannot be established. In turn, this makes it difficult to enable reproducibility of processes and projects or the process improvement measures applied.

Knowledge-intensive processes. The issue with lacking process automation is epitomized by the fact that the process is both complex and knowledge-intensive. As stated, SE processes involve new product development, which is a knowledge-intensive task [RaTi99]. Even if not dealing with product development, SE processes are mostly knowledge-intensive [KeHa02]. There is a need for capturing and sharing various types of information, including domain knowledge, knowledge about technologies, or knowledge about national or local policies [LiRu02]. Supporting this with an automated tool can be beneficial [TFB00]. Often, Wikis are used for SE knowledge management since they can be easily created and information can be quickly accessed [SBBK08]. However, retrieving contextually relevant information from Wikis is a difficult task [SBBK08]. Thus, knowledge management as well as knowledge transfer is hampered. However, SE is essentially a collaborative activity [JYW07, CoCh06]. Consequently, the other side of knowledge-intensive processes concerns the collaborations of the various individuals working in these processes [MBR15, MuRe14]. The different connections between humans, teams, tools, and artifacts are of crucial importance for SE success [JYW07, SQTR07]. However, efficient communication and process-aware collaboration remain a great challenge [Dust04], and, to a large extend, team work remains unpredictable and unplannable [BSV07]. Moreover, collaborative work in SE is still not adequately supported by tooling in SE [LeBo07].

Product quality issues. The created product – the software – has specific properties making it difficult to monitor and control its status. In turn, this complicates SE projects. Software Quality Assurance (SQA) has proven to be essential for SE. In particular, it has been shown that SQA has impact on project costs [Hami88, KKKM00, HuBo06, MSG13], which makes effective and efficient SQA mandatory. Effective application of software measurement remains a big challenge for software vendors [STT06]. Furthermore, software quality measures are often applied too late in the projects, although it has been proven that their application in earlier stages could save time and money [Hami88, SHK98]. Note that the application of quality measures is also problematic, since their effectiveness as well as efficiency depend on various factors, like the applicability of the measure, the project timing, worker competency, or correct execution of the measure [Hami88].

1.2. Contribution

This work originated from the Q-ADVICE (Quality ADVisory Infrastructure for Cooperative Engineering) project, whose goal of this project was the creation of a concept as well as a prototypical framework supporting the SE process. That concept as well as the framework shall enable the automation of various supportive aspects enhancing the quality of the SE process as well as its product. In Chapter 13, various aspects regarding the technical implementation of a prototype framework are discussed. All chapters before that deal with the abstract approach that extends process management technology to enable holistic support for SE projects taking into account the different aforementioned problem areas. We call this approach CPM (Context-aware Process Management). Its core contributions are aligned to the core problems identified:

- **SE process model implementation support:** CPM supports the implementation of entire SE process models. It provides facilities to enable automated process enactment in SE projects. This includes support for all process levels ranging from abstract processes to the concretely executed workflows. Further, CPM provides facilities to integrate process enactment directly with the project environment. Thus, a connection between the abstract process models and the concrete activities on the operational level is established. Context information is automatically collected and utilized for various purposes.
- **Advanced SE process enactment:** CPM also integrates advanced process enactment features to support dynamic domains as SE. It features dynamic processes, i.e., predefined processes may be dynamically adapted to match different situations. Furthermore, CPM enables context-sensitive adaptations, i.e., automatically collected context data is utilized to adapt running processes to the needs of the current situation. Further, CPM features facilities to model and execute dynamic workflows that are usually not covered by SE process models. Thus, these workflows can be integrated with standard process enactment, and, hence, can be guided, traced, and can profit from other CPM functionalities. Finally, CPM incorporates advanced facilities for exception handling. These take various types of context knowledge into account, as, for example, the states of activities, artifacts, or the SE process. Furthermore, CPM enables flexibility and automation for handling exceptions and is capable of not only automatically determining the right exception handling, but also the right person and time point for applying the handling.
- **Integration of processes with other areas of SE projects:** CPM integrates process enactment with other areas important for SE projects. One of these is quality assurance. This includes the automatic detection of potential problems in source code artifacts as well as the management of quality goals, proactive quality measures and reactive quality measures. Furthermore, quality measures can be prioritized according to quality goals and be automatically and context-sensitively distributed to the executing persons in alignment with their standard process activities. Another important area is SE knowledge management. CPM enables automatic management of collected SE knowledge utilizing machine-readable semantics. That way, the context-sensitive selection of applicable knowledge for SE engineers becomes possible. Furthermore, that knowledge can be automatically injected into the running process to support SE engineers in various situations. Finally, CPM supports collaboration in SE projects. It features various types of meta information that allow automatically recognizing coherences between different activities and artifacts even if they are executed in different areas or departments of a project and by different

persons. With this information, different types of automated coordination become possible ranging from simple information distribution to fully automated creation and distribution of new activities.

This work provides an evaluation of the developed concepts as well. By implementing a prototypical framework, questions regarding the technical feasibility of the approach are dealt with. Furthermore, detailed studies demonstrating the applicability of the approach were conducted and the framework was applied in two practical settings.

1.3. Outline

This thesis is split into five parts:

Part I (Problem statement and requirements) provides the motivation of holistic process and project support for SE. In Chapter 2 research question and research methodology are described. Chapter 3 provides background information on the SE domain and SE processes, whereas Chapter 4 elicits basic requirements for a tool providing automated holistic support in this domain.

Part II (Solution) is devoted to the solution. It starts with Chapter 5 providing basic information needed for understanding the work. In Chapter 6, the abstract solution approach is described. Then, Chapter 7 discusses the contextual extensions to process management concepts being the basis for all other components of the solution. Chapter 8 elaborates on the approach taken for modeling and enacting dynamic workflows extrinsic to the SE process models. In turn, Chapter 9 discusses automated contextual support for SE quality management. Chapter 10 gives insights into task coordination and Chapter 11 deals with SE process exception handling. Finally, Chapter 12 describes the automated contextual integration of knowledge management into the SE process.

Part III (Evaluation) is dedicated to the evaluation. Chapter 13 gives details on the technical feasibility and the implementation of the approach. Chapter 14 shows the practical applicability of the solution to a set of concrete scenarios. Finally, a discussion of related work and threats to validity is provided in Chapter 15.

Part IV (Conclusion) concludes the thesis with a summary and an outlook.

2. Research Methodology

This chapter presents the research questions addressed by this thesis as well as the research methodology applied.

2.1. Research Questions

Chapter 1 presented a problem statement and distilled main problem areas backed up by literature references: (1) the inadequate process support and implementation in SE; (2) the inadequate support of humans and their interaction in these knowledge-intensive processes; and (3) the inadequate integration of the product and its quality management into these processes. The first item corresponds to the process itself while the other two items refer to the integration of the process and other important aspects of the SE project. Thus, this thesis deals with three main research questions, the first being the general leading theme of the thesis and the other two refining the first.

Research Question 1: Is it possible to support SE projects by not only documenting, but operationally guiding and supporting their processes?

Research Question 2: Is it possible to operationalize and guide entire SE process models with (existing) automated tools?

Research Question 3: Is it possible to connect SE process enactment comprehensively to the actual course of the projects including artifacts and humans?

To answer these research questions the course of action is to analyze SE projects in practice as well as to do a comprehensive literature study. Based on this, we will create more concrete requirements to be fulfilled to answer the research questions.

2.2. Information Systems Research

In particular, this work deals with information systems supporting humans in SE. Therefore, this work relies on a combination of two science disciplines applied for Information Systems (IS) research that was postulated in [HMPR04, HeMa03]: design science and behavioral science [MaSm95]. In the following, we will briefly explain this combination and its suitability for this work.

Information Systems research approaches following behavioral science seek to provide a better understanding of the interplay of organizations, humans, and technologies that have a huge impact on the performance of those organizations. Behavioral approaches, therefore, develop theories to explain or predict organizational phenomena concerning the management, implementation, design, and analysis of IS. Opposed to this, design science aims to create and evaluate concrete artifacts solving the problems identified. These approaches are problem- and solution-oriented having their roots in engineering and sciences of the artificial [Simo96]. As opposed to natural science, however, design science approaches do not examine natural phenomena, but rather deal with those relating to and created by humans [Simo96].

Design plays a central role in these approaches and is to be understood as the goal-driven and deliberate organization of resources for achieving these goals. The combination of these two disciplines has been chosen in this thesis due to its applicability for complex and application-centric IS problems. The latter often cannot be precisely specified (i.e., as mathematical model) and thus cannot be optimally solved by one approach. Instead, they demand for more flexible descriptions and solutions. As an example, [Simo96] presents the creation of a robust IS architecture and classifies solutions to such problems as 'satisfying'. This means that they may not be optimal, but well suited and good enough for a certain class of problems.

In this work, the framework created in [HMPR04, HeMa03] is applied as shown in Figure 2-1:

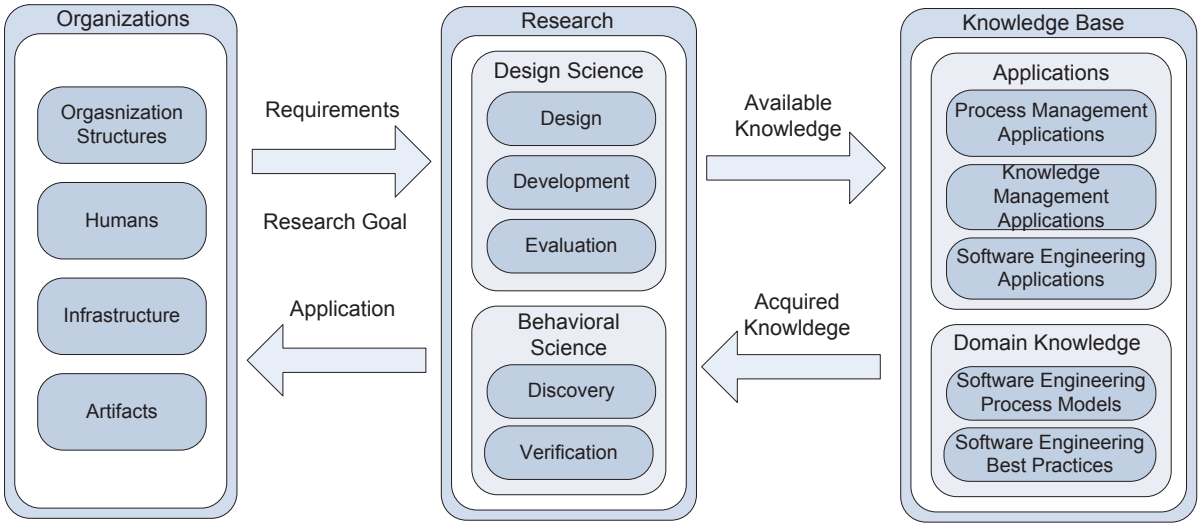

Figure 2-1: Research framework

As the first step, concrete experiences and information were gathered from two software-producing small- or medium-sized enterprises (SMEs). On one hand, this information comprises concrete requirements and the research goals. On the other, it consists of information about the concrete infrastructure of these organizations, including artifacts, humans and tools.

In the second step, a detailed literature study was conducted revealing information crucial for the SE domain. This included information from various applications for purposes like knowledge management or process management as well as SE domain knowledge (i.e., process models or best practices).

Based on the information gathered and aggregated, a framework was designed and developed. To ensure practical applicability, the evaluation not only included different case studies but also a concrete application of the developed framework in two practical settings.

For combining design science and behavioral science in IS research, [HMPR04, HeMa03] postulated seven guidelines that shall ensure the validity and effectiveness of that research (cf. Table 2-1). In the following, the application of these seven guidelines to this work is briefly discussed. Guidelines 1 and 7 are strictly followed as all efforts of this work result in concepts, algorithms and methods published in scientific papers. The relevance of the objectives was proven by sources from literature as well as the information gathered from two industrial software companies. The evaluation recommended by Guideline 3 is conducted through practical usage by two software companies. As such industrial evaluation with only three small teams is relatively fuzzy and error-prone, a set of concrete case studies has been created to evaluate the applicability of the different contributions of this thesis. Further, this work has a set of concrete contributions (cf. Guideline 4) outlined in Section 1.2. The research rigor (cf. Guideline 5) is facilitated by not only creating design artifacts, but also using these artifacts to create a concrete applicable solution (software) that can be practically applied. The search

process recommended in Guideline 6 was also followed. By the usage of concrete scenarios and a practical application, solutions that may not be optimal but yet satisfying for the problem were found.

Table 2-1: Research guidelines (adopted from [HMPR04])

Guideline	**Description**
G1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, model, method, or instantiation.
G2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
G3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
G4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, foundations, and/or methodologies.
G5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
G6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends, while satisfying laws in the problem environment.
G7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

3. Background

This chapter provides background information on the characteristics of the SE process and SE process models, respectively. Section 3.1 briefly discusses basic properties of SE process enactment. Section 3.2 then introduces prevalent SE process models as suggested in literature. We provide historical background and then present models prevalent in contemporary SE projects.

3.1. The Software Engineering Process

[BrBe11] provides the following definition of a software process: "A software process is a framework for carrying out the activities of a project in an organized and disciplined manner. It imposes structure and helps to guide the many humans and activities in a coherent manner. A software project progresses through different phases, each interrelated and bounded by time. A software process expresses the interrelationship among the phases by defining their order and frequency, as well as defining the deliverable of the project. ... Specific software processes are called software process models."

As already stated, the SE process is highly dynamic. On one hand, this results from the properties of the created product (i.e., the software). On the other, the creation of the product implies a highly intellectual, creative process that, in turn, necessitates much communication. The latter is needed across different abstraction levels (i.e., from the high level process of a project down to the operational level where concrete activities are executed) as well as different project areas (e.g., 'Quality Management' or 'Software Implementation'). This section enumerates the various groups of persons involved, tasks executed, and artifacts processed in order to explain communication channels as well as the highly dynamic properties of an SE process.

Usually, SE involves various roles [BrBe11]. First of all, there are the *software vendor* and its *customer*, who have to agree on the product to be delivered. As part of the software vendor, there exist vertically and horizontally divided areas. Vertically, there are levels such as company and business management, project management, and project staff. Horizontally, aligned from the first product idea to the final product, different teams participate: the *requirements analysts* communicate with the customer eliciting concrete requirements for the product to be developed. They represent the customer towards the developers. The *architects* are responsible for the technical foundations as well as architecture of the software and design decisions. In turn, the *developers* are in charge of the concrete realization of the requirements based on the chosen architecture. A *test team* verifies the technical functionality of the software, while the requirements analysts are in charge of the functional inspection of the software. Other responsibilities are related to *configuration management*, *problem and change management,* and *administration*. Furthermore, it is common that multiple companies collaborate to create one product or that in one company many projects are executed concurrently. Figure 3-1 shows a schematic description of a selection of different actors, artifacts, activities, and areas of an SE project together with their relations.

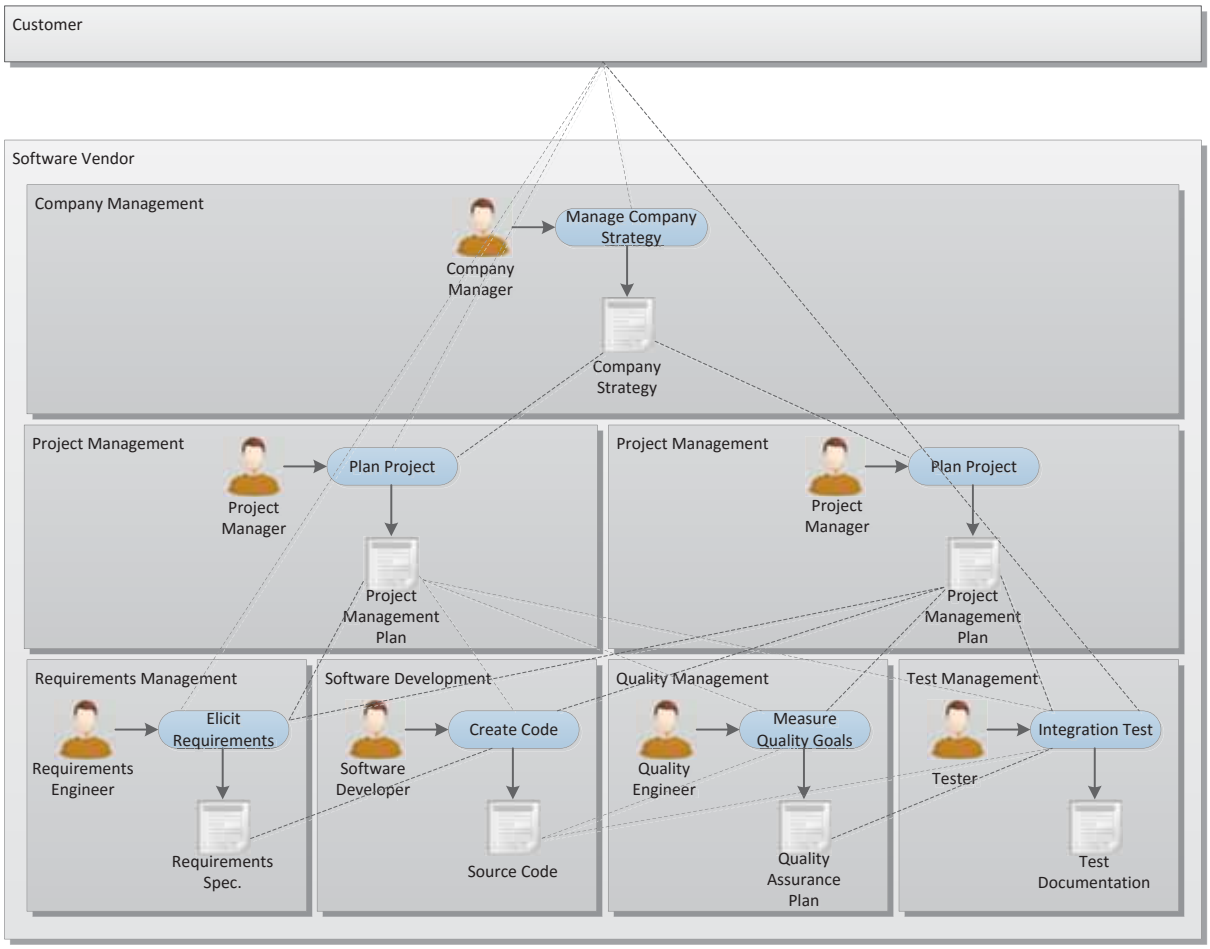

Figure 3-1: Examples of SE entities

Besides the source code, the artifacts processed in a SE project include various plans and specifications. In the following, a selection of artifacts are presented that have been standardized by the Institute of Electrical and Electronics Engineers (IEEE). The software requirements specification (SRS) [IEEE98a] covers the requirements of the software to be developed. The requirements can be spilt into two parts: customer requirements (similar to the German 'Pflichtenheft') and development requirements (similar to the German 'Lastenheft'). A software quality assurance plan (SQAP) [IEEE02] covers all development, testing and training activities in the project. The software configuration management plan (SCMP) [IEEE05], in turn, describes the necessary configuration management activities. The software test documentation (STD) [IEEE07] contains the documents needed for documenting the software tests. The software validation and verification plan (SVVP) [IEEE04] manages how the validation and verification of the software shall be documented. The design of the software is captured in the software design description (SDD) [IEEE09] and the governance of the entire project is described in a software project management plan (SPMP) [IEEE98b].

Usually, SE projects aim to create or extend software. In this context, various tasks need to be accomplished and coordinated among different groups of persons implying different artifacts. A project begins with the elicitation of its requirements. After their definition, the system architecture must be chosen and built. In parallel, the solution concept needs to be developed, which is mostly done in more than one step producing a preliminary concept first. The actual realization phase starts after having determined all parameters. To be finally deployed the solution must first be tested and, eventually, its different parts be integrated. The entire SE process is rather dynamic due to different factors: The intangibility of the created product makes it difficult to preplan it comprehensively implying a thing called ‘requirements creep’ [Jone96]. The latter describes the fact that in most SE projects requirements are evolving and cannot be concretely defined upfront. Another negative effect of the software’s properties is its aggravated measurability according to quality. To be able to improve

the latter, quality goals must be defined and measured [Kan02]. However, many companies are suffering severe problems in implementing effective measurement programs [STT06].

3.2. Software Engineering Process Models

Explicit SE process models have been developed and used for a long time in SE in order to enable governance, guidance and support for the SE process. In addition, such SE process models shall improve quality of the SE process as well as the produced product by enhancing repeatability and avoiding uncoordinated ad-hoc activities. Furthermore, process models can be the basis for process improvement since a process must be known to improve it. The following sub-sections give a brief overview about common SE process models and approaches.

3.2.1. Classical Approaches

Classical approaches in process specification have existed for many decades. Compared to the more recent agile approaches, they are based on a rather static and heavyweight process model.

Waterfall Model

The waterfall model [Royc70], which can be seen as the earliest structured system development approach, was mentioned first in 1970. It describes a sequential SE process, which originates from the manufacturing industries, and includes the phases depicted in Figure 3-2.

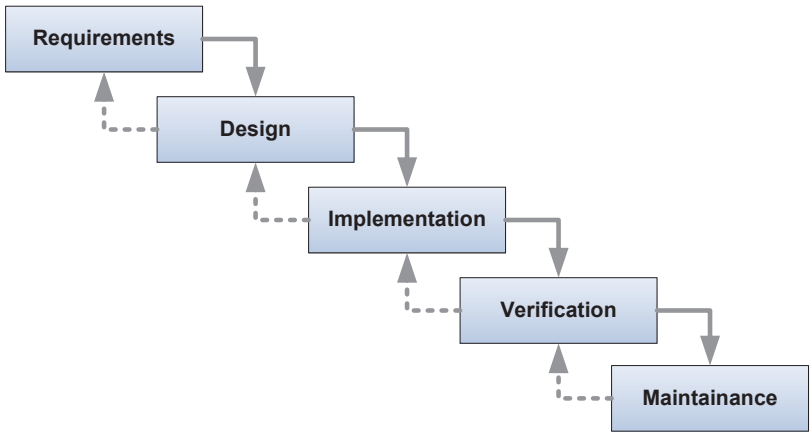

Figure 3-2: The waterfall model

These phases are processed sequentially, assuming that a phase transition is only executed if the current phase is finished. The process allows going back one step to the preceding phase, but not further. The waterfall model has turned out to only poorly capture the properties of the SE process. In particular, in SE it is usual that the requirements cannot be completely elicited before development starts. As another disadvantage in SE, designs often cannot be translated into working products in a straightforward way due to various limitations like, e.g., regarding technology.

Spiral Model

The spiral model combines elements of prototype-driven process methods with the classical SE process of the waterfall model [Boeh88]. The former takes into account that it may be difficult to know all system requirements upfront and thus proposes the development of system prototypes first. Its primary focus is to manage and reduce the risks of the overall SE process. Figure 3-3 shows the different phases of the process model.

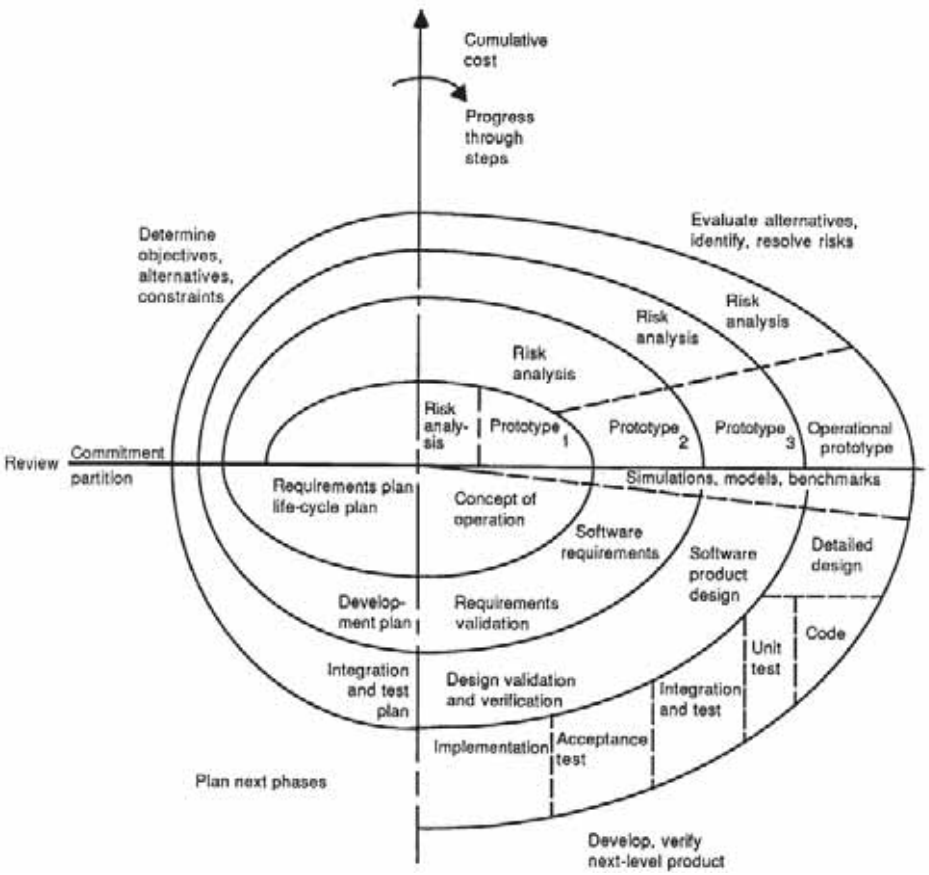

Figure 3-3: The spiral model (adopted from [Boeh88])

The process of the spiral model is represented as an expanding spiral corresponding to iterative developments. The inner cycles represent early development stages with system analysis and prototyping. In turn, the outer cycles represent the classic development cycle. Each cycle begins with the activity of risk analysis to incrementally identify critical factors in the project. The model is intended for big projects in risky areas and may imply too much management overhead for smaller projects.

V-Model

The V-Model is named after the alignment of its activities in the process model: activities are aligned like a 'V' as illustrated in Figure 3-4. The left side represents the elicitation of requirements and the creation of various specifications, whereas the right side represents the verification and integration of the developed system parts. The main objectives are the improvement of product quality and the minimization of risks as well as the facilitated communication of stakeholders and cost reduction for the whole project. The V-Model was initially developed for the German Federal Ministry of Defense in 1986. It was refined later to the V-Model 97 incorporating new approaches like object orientation. In 2005, it re-experienced a major refinement to the V-Model XT (eXtreme Tailoring) [IABG15]. The focus of the new model was to be easily tailorable to various organizations. It further considered stronger involvement of the customer, stronger modularization, and orientation towards incremental approaches. As opposed to the models described before, the V-Model XT is a rather heavyweight model, not only roughly describing different development phases, but also comprehensively covering different project roles and groups as well as their communication (i.e., describing 'Who' has to do 'What' and 'When').

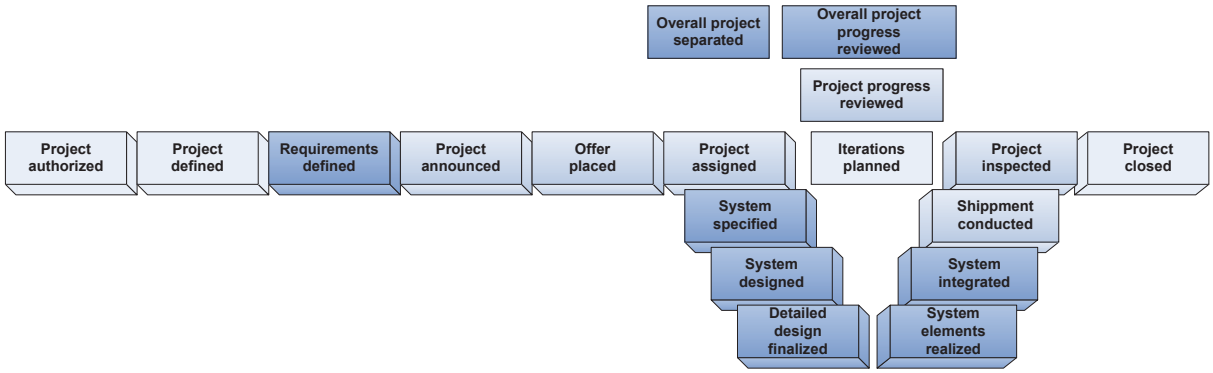

Figure 3-4: The V-Model XT (adopted from [IABG15])

As shown in Figure 3-4, the V-Model not only comprises development tasks, but project acquisition and definition tasks as well. For concrete development tasks (starting with 'System specified' until 'Shipment conducted'), multiple iterations may be applied. The activities to reach the milestones are rather abstract and comprise a number of more fine grained sub-activities. To group the latter, so called process modules are used. For example, process module 'System Development' comprises 49 activities (e.g., 'Preparing overall system specification') of which some are even specified as a workflow. Furthermore, the mentioned process module comprises 73 so-called products (i.e. artifacts) like 'In-service documentation' (includes all data needed by the customer to properly operate the system). These products have relations to the various activities as well as to roles (e.g., 'Requirements Analyst'). Furthermore, they have complex mutual relations, which include the 'Content-Related Product Dependencies' describing content-wise relations in the products, and 'Generative Product Dependencies' describing that one product is needed creating another. A key feature of the V-Model XT (eXtreme Tailoring) is its capability to tailor it to the current project by adding or omitting certain process modules even while the project is active.

3.2.2. Agile Approaches

Agile SE approaches [FoHi01] have emerged since classical approaches often fail to cover the dynamic nature of the SE process. In particular, agile approaches put more emphasis on the humans enacting the process as on the process itself. Responding to change is more favored than rigidly implementing a process model. Consequently, small, self-organizing teams are installed. Furthermore, the customer is more tightly integrated into the SE process in order to be able to quickly communicate changing requirements. Another important aspect concerns the utilization of short cycles, which should always produce a working product. Thus, the customer can already get familiar with the product and requirements changes can be communicated earlier.

Scrum

Scrum [DeSt90, TaNo86, ScBe01] is rather a framework than a full process model. Thereby, many of the decisions in the SE process are left up to the team. Scrum teams are self-organizing and cross-functional, meaning they comprise members of different groups such as developers, requirements analysts, or testers. The Scrum process defines three main roles: 'Scrum Master', 'Product Owner', and 'Scrum Team'. The 'Scrum Master' is a kind of team leader whose main responsibility is the support of the 'Scrum Team' by removing impediments that prevent the team from completing its tasks. In turn, the 'Product Owner' is something like a proxy for the customer of the project: He analyzes business needs and defines the requirements for the 'Scrum Team'. The latter is in charge of realizing the functionalities of the software to be produced. Figure 3-5 illustrates the process model.

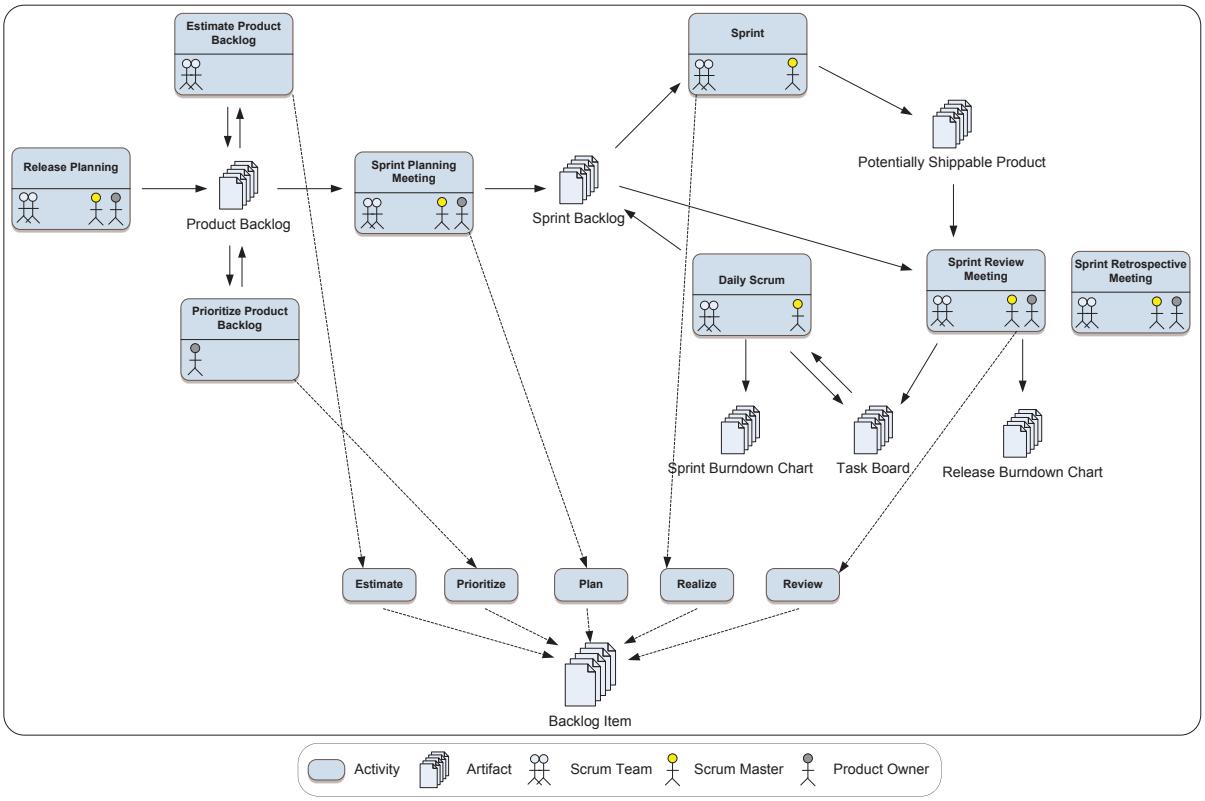

Figure 3-5: The Scrum process

As shown in Figure 3-5, the Scrum process features different artifacts called ‘Work Products’. These can be separated into two categories: ‘Task Board’ and ‘Burndown Charts’ that list different activities to be accomplished within a certain timeframe. The requirements (i.e., different functionalities of the software) are represented by the ‘Product Backlog’, ‘Sprint Backlog’ and ‘Potentially Shippable Product’. At the beginning of a project, which starts with the ‘Release Planning’ activity, the ‘Product Backlog’ (including all desired functionalities) is specified and the number and length of sprints is determined. After that, the backlog items are estimated by the team and prioritized by the ‘Product Owner’. In the ‘Sprint Planning Meeting’, it is determined which items shall be realized in the current sprint. These items are then moved to the ‘Sprint Backlog’. Within a sprint, all scheduled backlog items are realized and everyday a short ‘Daily Scrum’ meeting is conducted for coordination purposes. At the end of a sprint, the backlog items are reviewed with a presentation of the ‘Potentially Shippable Product’ in the ‘Sprint Review Meeting’. Following the latter, there is an additional ‘Sprint Retrospective Meeting’ to discuss the past sprint.

eXtreme Programming

eXtreme programming [Beck00a, Beck00b] targets at smaller teams and the programming tasks constitute the main focus. As fundamental assumption, the customer does not know all requirements prior to project start. Therefore, the entire process is organized incrementally and dynamically. Requirements are described in terms of user stories which are a lean form of use cases focusing on the user’s view of the system. Extreme programming describes an open, fluent process that relies heavily on the participation of humans.

Some key practices are mentioned in the following: Programming is mostly done as pair programming where two developers share one computer to develop the software. That way, knowledge transfer shall be furthered and the error detection rate shall become high. Tasks are not distributed to humans, but to the team, and then become dynamically distributed. Humans do not have strict responsibilities and work is always shared. The dynamic process builds on permanent testing, integration and refactoring of the code.

Criticisms of extreme programming target at the low level of governance it provides, while relying heavily on the participation of the involved humans, which presumes ideal developers and customers. The process can be seen as too dynamic because it assumes continuous change. It has been proven that changes of the requirements get more expensive in later project stages. Furthermore, in extreme programming, it can be difficult to guarantee an exact amount of functionality at an exact time point.

Unified Process

The unified process [JBR99, Scot02] is an iterative SE process framework that is very popular and has many derivates. As its two main characteristics, this process strongly focuses on the architecture of the developed software and on addressing the risks in early project stages. The unified process knows four project phases as depicted in Figure 3-6.

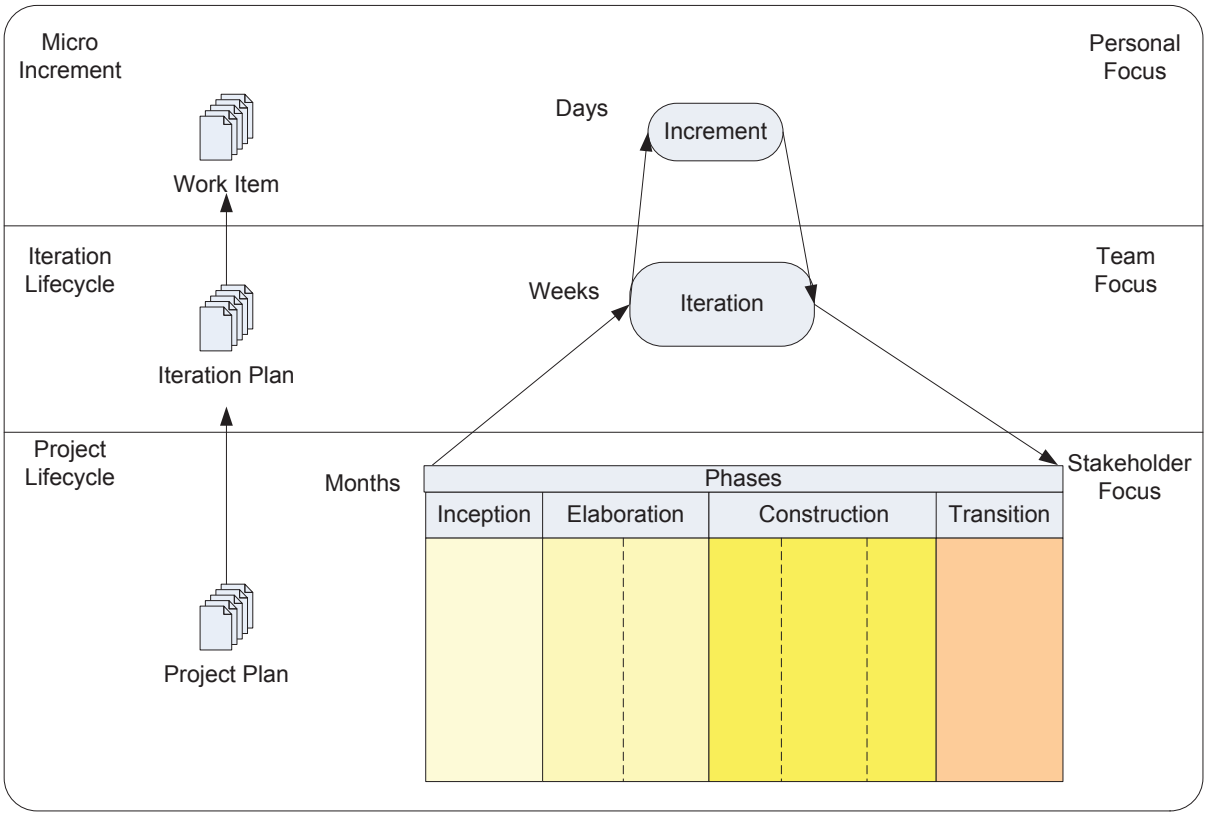

Figure 3-6: The Unified Process (OpenUP, adopted from [EcFo15])

The four phases (Inception, Elaboration, Construction and Transition) are separated into iterations. In each phase, different amounts of work in the core disciplines are accomplished. These disciplines are business modeling, requirements analysis, system analysis & design, implementation, testing, deployment, configuration & change management, project management, and environmental tasks.

The unified process comprises several refinements with different focus. Probably, the most well-known is the rational unified process (RUP) [Kruc99]. RUP is a sophisticated and heavyweight variant, which governs activities in great detail. RUP contains over 30 roles and over 130 activities. Thus, it can be used effectively only in teams of more than ten humans. Another refinement is the Open Unified Process (OpenUP) [Ecfo15], which is part of the Eclipse process framework (EPF). All these RUP variants aim to provide a simpler, open version of the process, while capturing all essential characteristics of the unified process or RUP. OpenUP features three levels of granularity: At the project level there are four phases (as defined by the Unified Process): Inception (roughly agree upon the goals of the project), Elaboration (agree on the technical approach), Construction (realize main part of the system), and Transition (make the system ready for its transition to customer). Within each of

these phases, multiple iterations may take place. The iterations, in turn, comprise several more fine-grained activities (e.g., ‘Develop Solution Increment’ for developing a new part of the software). These activities may be specified in terms of workflows containing even more fine-grained activities (like ‘Implement Solution’ or ‘Implement Developer Test’). The OpenUP process features different kinds of guidance to support the project participants. Activities on the most concrete level are supported by so called steps that roughly outline what has to be done to complete the activity. Furthermore, the model features concrete checklists to be applicable at certain points.

3.3. Summary

This section provides a summary about the SE process extracted from the properties and criticisms of the introduced process models. Due to the numerous efforts regarding explicit process models, it is evident that they are essential for SE projects enhancing repeatability, traceability and, first of all, quality of the process and thus of the product as well. Yet, it cannot be guaranteed that process models are followed since they are mostly abstract and applied manually and documentation-centric.

The waterfall model and in the particular the criticism on it show that a rigid process is not the right choice for mirroring the dynamic properties of the SE process. This results to a great extend from the fact that all requirements can be known a priori only in very rare cases. The Spiral model, in turn, tackles this issue as it provides an iterative process, which strongly targets at risk analysis and prevention. Criticism on that model include that it is too heavyweight and not applicable to all kinds of organizations. Finally, the V-Model XT incorporates far-reaching tailoring facilities to be applicable to different organizations. It also puts a strong focus on risk management and communication support. Yet it is still rather heavyweight and thus not suitable for small projects or teams.

Agile approaches were developed as answer to the heavyweight classical process models. Scrum, in turn, puts a strong focus on humans and small self-organizing teams. eXtreme Programming is even more targeted towards the individual. These approaches are lean, but criticism includes that there is not enough governance and thus unpredictable results might be produced. The Unified Process is more static focusing on the architecture. However, some refinements, including RUP, are considered too heavyweight, same as the classical approaches.

All in all, comparing the criticism of the classical and the agile approaches, one can state that it is difficult to provide appropriate process support for SE projects. On one hand, comprehensive process models are often too static and require much cumbersome additional work imposed by the process model. On the other, leaner and more dynamic process models often lack comprehensive support. In particular, they considered to be chaotic and heavily relying on humans. Another fact we discussed constitutes the diversity in SE process models. It is therefore not easy to select process support matching the current company, organization and situation. None of the mentioned process models seems to be applicable to all types of organizations. Altogether, it can be stated that striking a balance can be beneficial, i.e., to provide process guidance without implying too much distractive additional work. A tool providing automated assistance may aid in reaching that goal, taking cumbersome tasks in heavyweight process models and supporting agile teams in the background.

4. Requirement Analysis

This chapter deals with concrete problems and elicits basic requirements from the abstract problems discussed in Chapter 1.

4.1. Concrete Problems

Basically, this work aims to support humans and to address the abstract problems discussed in the preceding sections by a framework. In SE, various tools are prevalent supporting different aspects of process implementation, knowledge management, or quality management. However, many problems remain unsolved as mentioned by the various studies we reference in the relating chapters. To better understand these problems and to support requirements elicitation, we split the three abstract problems up to derive a greater set of more concrete problems that can be better connected to SE tool support. As summary of these problem statements, we extract eight concrete problems relating to SE projects and their process having a big impact on the quality of both the SE projects and the products created by them (cf. Figure 4-1). The latter is separated vertically: On the left side, process specification utilizing abstract process models is depicted. In the middle, the (automatically supported) implementation of such process models is shown for concrete projects. Finally, the left side depicts the SE process as it is really executed by humans creating and manipulating artifacts using SE tools.

We will further support these problem statements by concrete scenarios. The latter were created with information from literature and especially with information gathered from two practical settings. For confidentiality reasons the scenarios are abstracted and generalized. Further, they are centered around a fictional company called ‘The Company’. Not all of the scenarios directly correlate with an abstract problem identified in the Problem Statement. In particular, the first three problems (Automated Process Governance, Context Integration, and Process Dynamicity) are of abstract nature playing a role in most of the scenarios.

Lack of Automated Process Governance (Prob:AutoProc). One problem area concerns process tracking and guidance, referred to as automated process governance in the following. If a project is to be executed in an effective, efficient and repeatable manner, studies have shown that it should be based on a defined process [GGK06]. Furthermore, process models may contain important information about the projects [BWHW06, RiJa00, Mall09]. As discussed in Chapter 3, many SE process models have been developed including Scrum [ScBe01], the Unified Process [JBR99], or the V-Model XT [IABG15, RBTK05]. As a problem, typically, these models exist only on paper or web pages, i.e., they are only used for process specification and documentation. In many cases, the process is rigid and prescriptive, and it differs from the real dynamic work performed in a project [BDS+99, Ambl02]. Furthermore, the impact of the models on actors and concrete activities often remains low [Wall07]. Automated support for enacting such process models is desirable. There are numerous tools capable of automated workflow governance. These tools strongly focus on the control-flow perspective meaning they are capable of governing the sequencing of different activities and transferring different tasks to the humans. In addition, they often provide limited means for integrating data objects and an organizational model. However, they fail in covering the different aspects of process models like guidelines or checklists, or dynamic features like the V-Model XT’s dynamic tailoring (cf. Chapter 2). Consequently, the automatically assisted implementation of a whole process model with such tools remains a challenge.

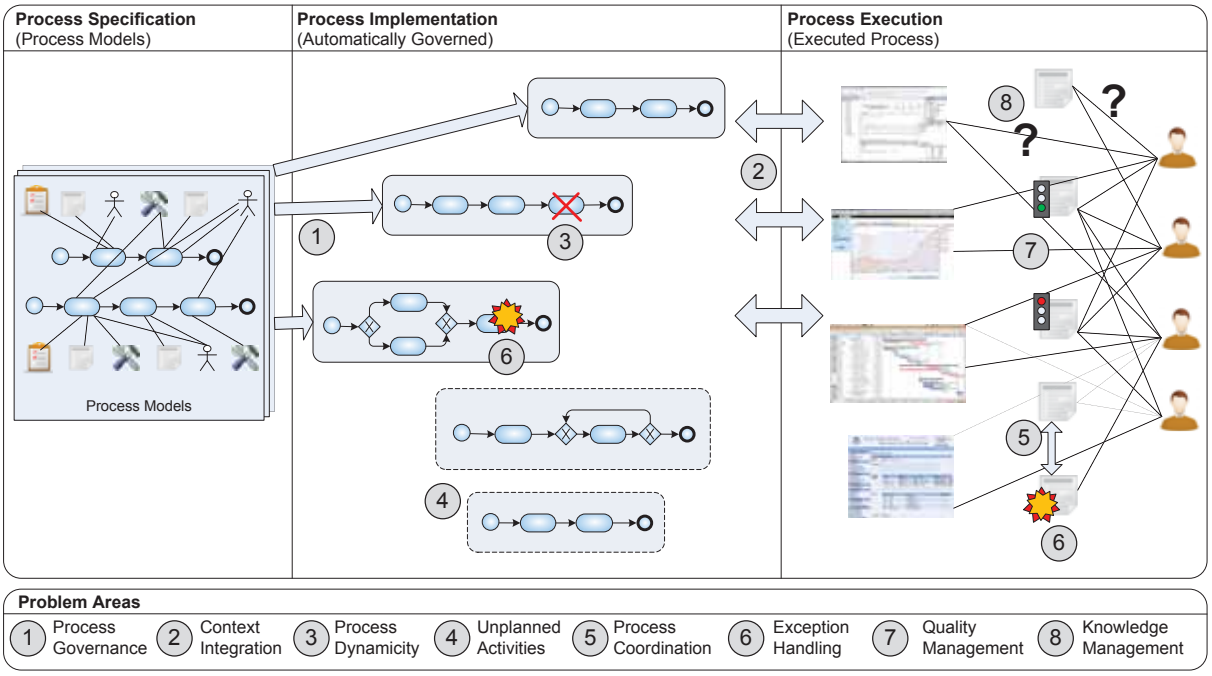

Figure 4-1: SE project problems [GOR14]

Lack of Context Integration (Prob:ContInt). A second important problem area concerns contextual integration: even if some automated process implementation and guidance is present in a project, this does not necessarily mean that the specified and actually executed processes align. In reality, a myriad of environmental variables affect process enactment [Schw97, MaVe03, BPNS07]. In SE, the latter mostly deal with various actors using different tools (e.g., requirement management tools or IDEs) to manipulate various artifacts being crucial for the process. In turn, these activities and tool interactions are not directly captured in the process models since they are too fine-grained. Thus, a dichotomy between the planned and the actually executed process may exist.

Process Dynamicity (Prob:ProcDyn). Reality has shown that project enactment not always happens exactly as planned [BDS+99, RHD98]. A planned process is a good starting point. However, if the real course of a project deviates from this plan, it will be a challenge to keep the plan in line with reality [ReDa98]. Most contemporary PAIS still rely on rigidly predefined workflows and only feature rather limited abilities to cope with such dynamic changes [Pevd06, ReWe12]. Thus, the planned and the actually executed process diverge more and more, and the former becomes irrelevant over time.

In the following, we present a scenario relating to problems with process model implementation in SE projects. The scenario does not deal with a concrete use case or situation, but rather with a specific process model and issues relating to its automatically supported implementation. For this purpose, we chose the OpenUP [EcFo15] for several reasons:

- Availability: OpenUP is a freely available derivate of the Unified Process. It requires no licensing fees or other costs.
- Understandability: OpenUP is clearly structured and the Eclipse Foundation provides comprehensive documentation free of charge.
- Comprehensiveness: OpenUP covers both abstract and operational process areas including workflows ranging from abstract phases of a project to concrete developer workflows.
- Contextual relations: OpenUP specifies a rich set of entities that relate to real entities or persons in an SE project like artifacts, tools and roles.
- Comprehensive human tasks: OpenUP features various different activities and tasks of different granularities for humans.
- Comprehensive support features: OpenUP comprises a rich set of supportive artifacts like checklists or guidelines.

OpenUP is manually implemented. In particular, the whole process definition exists as web pages [EcFo15]. When implementing OpenUP in an SE project, the involved persons must gather information manually and apply it to the project. We will use OpenUP as scenario for illustrating the following problems: the basic automated implementation of the whole model (cf. Prob:AutoProc), the establishment of connections from this implementation to the 'real world' (cf. Prob:ContInt), and the dynamic nature of SE process enactment (cf. Prob:ProcDyn). For the sake of illustration, Figure 4-2 shows five excerpts from the OpenUP website comprising a list of various activities, an operational workflow specification, relations of artifacts and roles, a checklist, and fine-grained activity steps.

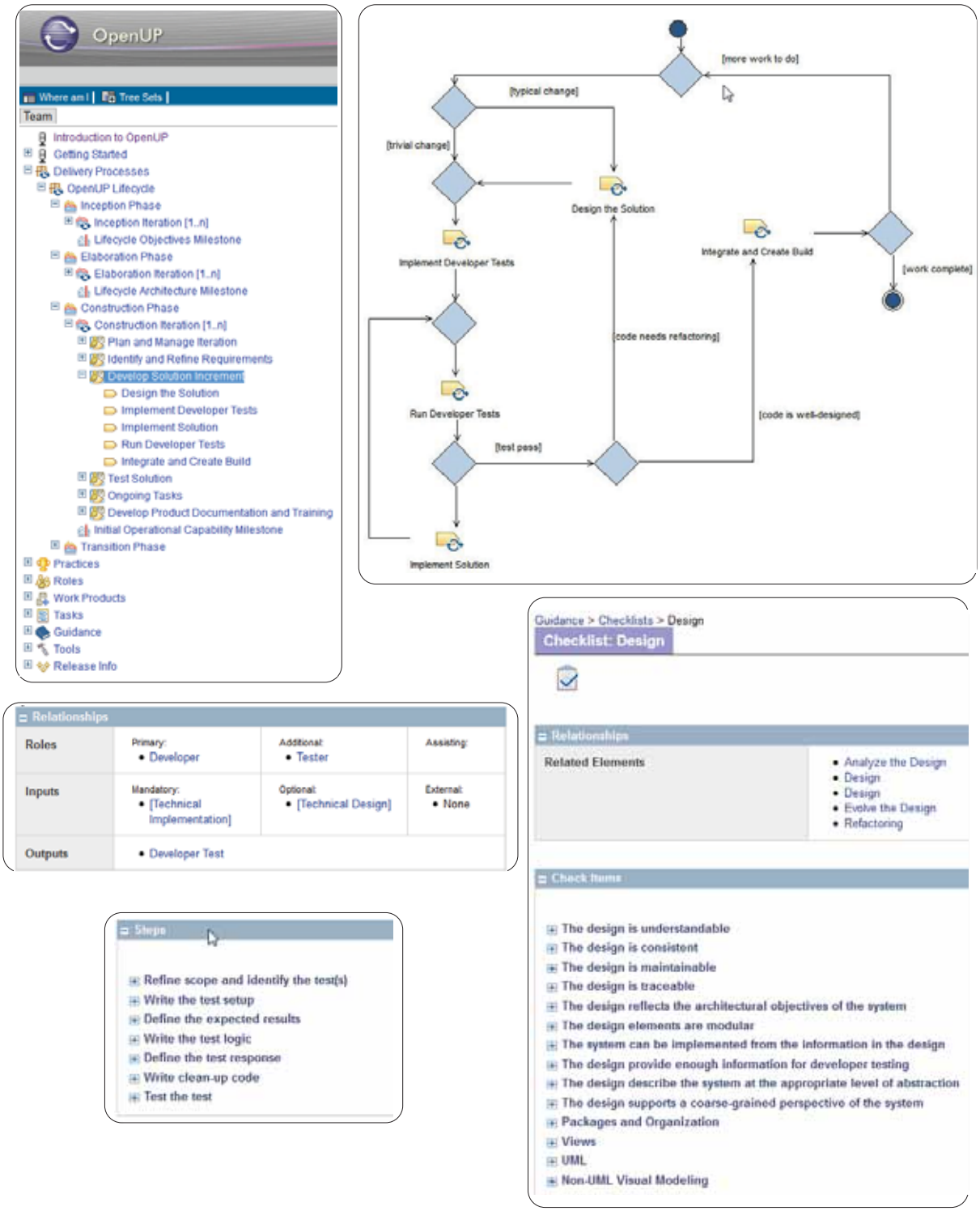

Figure 4-2: OpenUp excerpts (adopted from [EcFo15])

These excerpts show that the OpenUP indeed comprises important and useful information to aid the SE process. However, there is neither a tool implementing or supporting this process model nor a strategy on how to achieve this with any tool in place. Thus, this information remains disconnected from the real process enacted in the SE project. The information not only needs to be gathered manually by humans, it is also not tailored to the concrete project or situation. It does feature detailed information from abstract phases of a project to operation workflows. However, the latter, like the 'Develop Solution Increment' workflow, are rigidly predefined and not integrated with other tools or the humans in the project. Thus, support for handling unforeseen situations and adapting process enactment are also not in place.

Unplanned Activities (Prob:UnplanAct). The application of PAIS technology in dynamic and evolving domains such as SE is difficult [JaCo93]. Reality often diverges from rigidly pre-defined processes [McCo01, CNGM95] in that domain. In fact, process models cannot cover all workflows actually executed in an SE project. Hence, we distinguish between intrinsic workflows being part of the process and extrinsic workflows (cf. Chapter 8) being unforeseen in the latter. Such extrinsic workflows can be executed based on specific situations, but can be also recurring common tasks (e.g., bug fixing or technology evaluation). These tasks rely heavily on the current situation, remain unplanned and untraced, and may impact timely process enactment (cf. Example 4-1).

Example 4-1 (Ad-hoc activity):
Consider an ad-hoc activity as it was perceived during an interview conducted with a developer as part of an industrial case study. During the interview, a requirements analyst came in, telling the developer that he had to do a presentation for the customer soon. He had already received a current version of the software. However, shortly before the presentation he found a new bug endangering the success of the presentation. Hence, the developer quickly started to work on that issue, was able to fix it, and the requirements analyst received the fix via USB stick. According to the developer, such ad-hoc activities occur often, take up to half an hour, and remain untraced.

In the following, we will use the term *process coverage* to refer to the coverage of the actually executed processes in an SE project the used SE process model can cover. The models feature a list of standard SE processes. However, they do not cover a great number of activities executed in daily work in an SE project. Thus, these activities remain unplanned and untraced and can even influence the planned processes enactment. Due to these uncaptured activities the planned as well as the actually executed process can move increasingly apart from each other. Furthermore, the planned process can be delayed without exposing the reason for the delay. Finally, the unplanned activities are not guided, supported or governed. They are executed completely manually without any process or knowledge support. Example 4-2 deals with a concrete situation for such extrinsic workflows.

Example 4-2 (Process coverage shortcomings):
The Company uses a SE process model for standard development activities. However, there are various issues in everyday work not covered by such a model. These include activities like bug fixing, refactoring, technology swapping, or infrastructural issues. There have been efforts in The Company to model workflows for these issues in order to provide the humans with automated support and guidance. Since there are various kinds of issues with ambiguous and subjective delineation, however, it is difficult and burdensome to universally and correctly model them in advance for acceptability and practicality. Many activities may appear in multiple issues, but are not necessarily required, bloating different SE issue workflows with many conditional activities if pre-modeled. Figure 4-3 shows such a workflow for bug fixing that contains nearly 30 activities, many of them being conditionally executed for accomplishing different tasks like testing or documentation. An example is provided by static analysis activities that are eventually omitted for urgent cases. Furthermore, there are various reviewing activities, having different parameters (like effectiveness or efficiency), where the choice can be based on certain project parameters (e.g., risk or urgency). The same applies to different testing activities. Moreover, it has to be determined whether a bug fix should be merged into various other branches in the source control system.

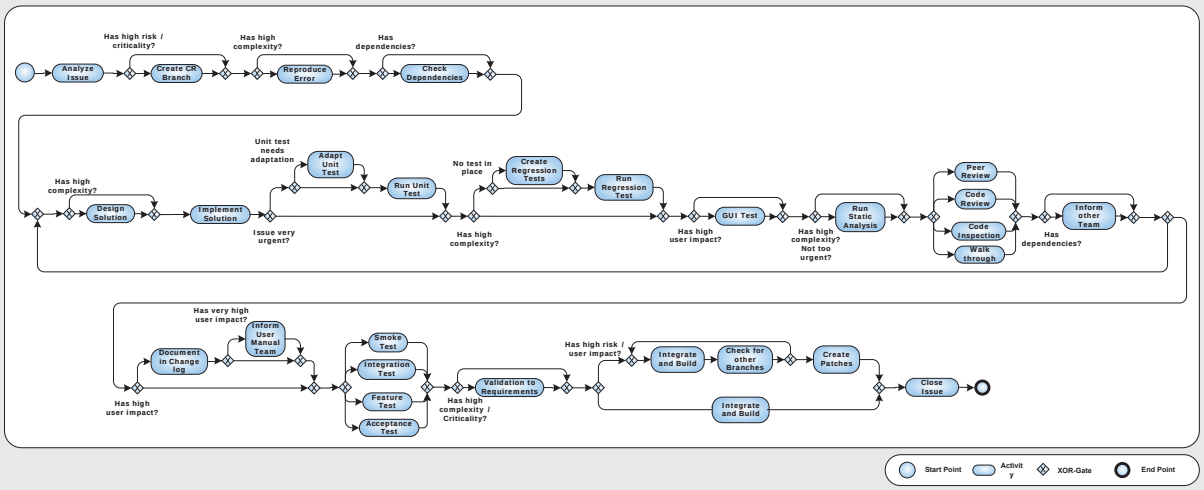

Figure 4-3: Example of pre-modeled workflow for bug fixing

As many decisions in the workflow rely on properties of the situation, many activities could be excluded prior to enactment as each situation requires another workflow that marks a subset of the workflow shown in Figure 4-3. However, the situational information for making such decisions is not always in place and gathering it would require additional efforts from humans. Another option, modeling many smaller workflows for different situations is also problematic, as the matching workflow for each situation would have to be determined manually. Additionally, that solution would result in a large number of modeled workflows making the selection of them even more inefficient. Finally, many of the activities and even whole fragments of the workflows would appear in multiple workflows resulting is redundant modeling. Usually, such redundant model fragments are difficult to maintain and might lead to diverging models over time [WRMR11].

Uncoordinated Collaboration (Prob:Collab). In a complex project, there are always persons, tools, activities, and artifacts related to each other [JYW07, SQTR07]. This fact implies that an activity a person executes to change an artifact can have an impact on other artifacts, which again has an impact on the activities of other persons. As example of a relation consider architectural specifications and relating source code artifacts. As some of these activities may be covered by the process, while others are not, this can result in problematic artifact states if many related adaptations by different humans are applied in an uncoordinated manner. As aforementioned, collaboration remains one of the biggest challenges in SE projects [Dust04] and team work is still not adequately supported [LeBo07].

As the sizes of companies, departments and projects grow, communication between collaborating humans and teams becomes increasingly challenging. Humans are often involved in multiple projects in parallel, each of them having its own artifact base the humans work on. Hence, humans are often switching between the projects and concurrently manipulate artifacts of these different projects. In turn, this can lead to a myriad of different problems relating to the artifacts or tasks conducted. Example 4-3 concretizes this.

Example 4-3 (Coordination shortcomings):
Being a growing small to medium sized enterprise (SME), The Company suffers from the inability to satisfy increased coordination needs. Team sizes are growing and various projects are executed in parallel. Humans often have to switch between different projects and within each project larger numbers of humans are working on the same artifacts. Without additional coordination effort things might be easily forgotten.

One concrete problem reported by developers is related to frequent project switches. A person doing this in such a multi-team / multi-project environment must manually gather context information after a switch in order to work effectively: Which assignment has to be processed for which project? What

are potential milestones and deadlines? What is the state of the currently processed assignment? What are upcoming activities to complete it?

Two other problems relate to cooperatively working on the same artifact base. As the first issue in this situation, activities and accompanied changes to artifacts often remain unnoticed by other humans. For example, if two teams (e.g. a development team and a test team) are working on the same source code artifacts they might want to get informed about changes of them. Such information is often transferred manually and is therefore prone to omissive errors.

The third problem directly relates to the artifacts and their relations: Artifact changes often imply certain follow-up actions that are hitherto coordinated manually. Figure 4-4 depicts a scenario detailing this: It deals with a source code artifact being part of an interface component: since the file belongs to an interface component, the applied changes might not only affect the unit tests of the file, but also other artifacts such as the architecture specification or integration tests. Usually, these additional activities are neither covered by the SE process nor governed by any workflow; manual coordination can lead to impacts being forgotten and result in inconsistencies, e.g., between the source code and the tests or specifications. The fact that these activities belong to different project areas with often also different responsible persons makes this even more difficult. Even if not forgotten, follow-up actions could benefit from automated governance and support. Furthermore, it can be difficult to determine which stakeholder should be informed about which change and when, especially considering the dynamic and diverse nature of the artifact-to-stakeholder relationship and various information needs.

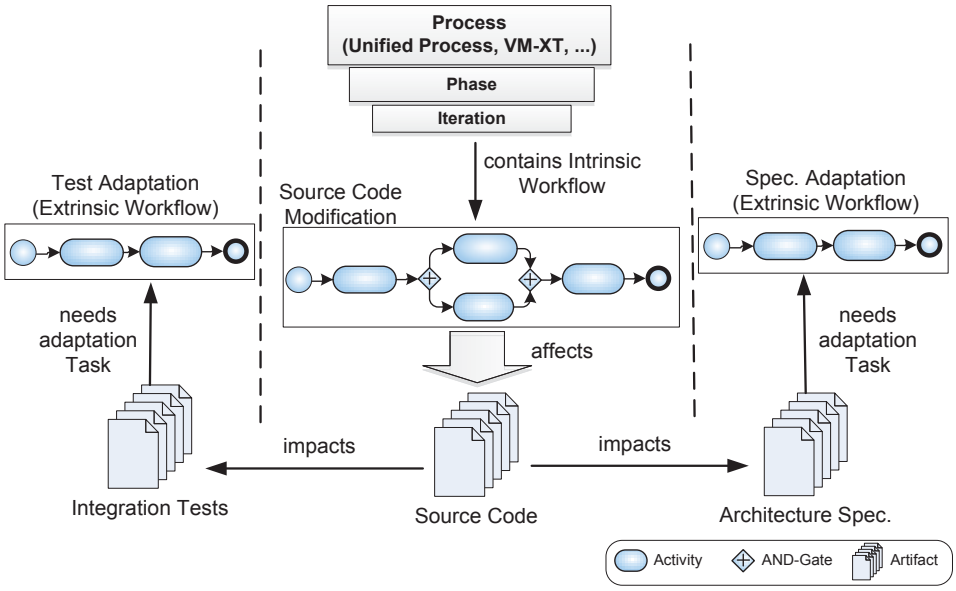

Figure 4-4: Artifact and implied activity relations

Process Exceptions (Prob:ProcExc). During project enactment, unforeseen and exceptional situations occur as the SE process is not fully predictable [Schw97, BDS+99]. In turn, this poses a big challenge to any framework seeking to provide holistic process support for such projects. Contemporary workflow management technology has limited capabilities in this area, only dealing with exceptions directly relating to activities [ReWe12, RAH06]. In practice, process exceptions are often not that simple and also not easily detectable. Further, they may relate to processed artifacts even without the person working on these artifacts noticing them. Finally, to select an exception handling suitable for both the situation and person is challenging.

Complex exceptions are not related to the malfunction of a single tool or program, but to the prescribed process or other more complex coherences in an SE project. Such exceptions can relate to activities being part of the prescribed process or to others being extrinsic to the latter. They may also relate to artifacts processed in the course of the project, even if all activities seem to be executed as intended. Such exceptions are difficult to detect and handle even if The Company uses a PAIS providing process implementation support. In the following, two concrete examples (Example 4-4 and Example 4-5) are provided to illustrate this.

Example 4-4 (Exception handling shortcomings):
The Company uses an SE process model. However, there is no tool in place to govern, support or enforce the executed process. Consider the following situation: A developer creates new code as intended as part of a project: Assume that it is prescribed by the process of that project that he shall create and execute a unit test for this code. As the process is neither enforced nor supported, however, he can intentionally or unintentionally omit these activities. If such things happen, often a growing portion of the code remains untested. This, in turn, endangers reliability of the code base.

A second scenario deals with a known bug in a source code artifact that is, for example, reported by a customer, tracked in a bug tracking software. The bug is then assigned to a developer who shall fix it. When applying a bug fix to the source code file, the removal of the defect might unintentionally introduce other problems to that file. For example, source code complexity might increase if multiple humans applied “quick and dirty” fixes. Thus, the understandability and maintainability of that file might drop dramatically and raise the probability of further defects.

Non-optimal Quality Management (Prob:QualMan). Another problem affecting many SE projects concerns the quality of the software produced [Jone10]. Hence, quality assurance is a crucial factor for any SE project. However, in many SE projects, quality assurance is understood as applying some bug fixes at the end of the project when time allows for this. Studies have shown that this is ineffective and quality measures should be applied systematically during SE project enactment [Hami88, SHK98]. In particular, this requires proactive as well as reactive quality measures. The challenge is to effectively and efficiently integrate the application of these quality measures with the SE process. Concrete issues include the following: quality management is often considered a ‘nice to have’ discipline creating no additional value. Very often it is difficult to integrate quality management activities with the course of the standard SE process. Furthermore, quality management is often only executed in a reactive fashion applying fixes for known bugs. No quality goals are defined that could be proactively supported to prevent the occurrence of bugs. Example 4-5 illustrates such a situation.

Example 4-5 (Quality management shortcomings):
The Company, being a growing SME, starts with various efforts to support reproducibility of project enactment as well as product quality with process management and quality management. As aforementioned, a process model for the SE process is used. Furthermore, as the number of bugs reported by customers shall decrease, quality management tools are applied. This includes bug trackers and static code analysis tools. However, both quality and process management are not well governed or supported. Quality goals are not defined for projects and thus, no proactive quality management can be applied. There is no real awareness of the execution of planned development activities. Thus, it is difficult to integrate quality management activities into the standard SE process. Static code analysis is only used at the end of projects and due to the time pressure often present in that situations, many detected problems still remain unsolved.

Unutilized Knowledge (Prob:Knowl). The creation and modification of software is a complex and knowledge-intensive task [RaTi99] and software is an intangible asset. It involves knowledge from different sources, all of which are crucial for the success of the task [LiRu02]. This includes information on the process, the coding style and other specifics of the company, the used framework or area (frontend or backend development), and so forth. Companies often neglect this fact and do not implement proper knowledge management. Even if some knowledge store is implemented, knowledge

retrieval and effective knowledge usage remain an issue [SBBK08]. This often leaves software engineers without all required knowledge and thus makes their work ineffective and error prone.

As mentioned in Chapter 1, wikis are often used to let project participants store specific knowledge. They make recoding of information easy, but management and retrieval of the latter often constitutes a challenge. This is aggravated by the fact that knowledge related to SE is usually context-dependent meaning that it must match the properties of the situation and the involved person. As scenario for illustrating knowledge management shortcomings a situation comprising different information needs in The Company is presented in Example 4-6.

Example 4-6 (Knowledge management shortcomings):
As a growing SME, The Company frequently hires new developers. The latter get training at the beginning to ensure that they can work effectively as early as possible. However, they still might not have a great share of the concrete information relating to projects, tools or the process. For example, this might include information about the coding style applied in The Company. Also, specific process-related information might be recorded somewhere, but the new developer might not know exactly when and where to acquire that information. Another example are technical specifics about the project he starts to work in as, for example, how source control management is applied including information about different development branches and the commit procedure. Lacking all that information, there is a high probability that the developer will cause many issues when he starts working.

4.2. Basic Requirements

This section gives an overview on the high level requirements for a tool providing automated support for the SE process. We elicit these requirements based on two foundations: First, we refer to the problems discussed in Chapter 1, including their support by literature. Second, we refer to our observations from practice as indicated by the scenarios in this section. As aforementioned, the basic requirements listed in the following will be detailed with sub-requirements in the relating chapters of this work.

- *Requirement Automated Process (R:AutoProc)*: The most basic requirement to a tool enabling holistic SE project support is to provide SE process support. Related problems have been discussed in Prob:AutoProc. This includes the automatic implementation and enactment of processes in the tool.
- *Requirement Context Integration (R:ContInt)*: As elucidated in the problem statement (cf. Prob:ContInt), there is a myriad of contextual information in the project having a significant impact on process enactment. For example, context information plays an important role for collaboration, quality management, or exception handling. Therefore, a tool aiming at holistic SE project support must have facilities to integrate process enactment with context data.
- *Requirement Dynamic Process (R:DynProc)*: SE projects are dynamic as already shown in the problem statement (cf. Prob:DynProc) and confirmed by the scenarios in this chapter (e.g. relating exception handling or quality management). Therefore, a tool supporting these projects must be capable of coping with dynamically changing situations and aligning the process with their properties.
- *Requirement Process Coverage (R:ProcCoverage)*: SE process models cover many workflows. However, as shown (cf. Prob:UnplanAct), they disregard many activities and processes executed dynamically as part of everyday work. When aiming at true holistic support for SE projects, a tool must include these workflows and activities as well.
- *Requirement Coordination (R:Coord)*: SE projects comprise numerous different areas, actors, and artifacts. Projects are executed in parallel and multiple persons work on the same artifact base. Activities, roles and artifacts have relations to each other and collaboration is not easy to maintain. This can lead to various issues as discussed (cf. Prob:Collab). A tool providing holistic SE project support must be aware of such connections and be capable of managing coordination and collaboration in such a project.

- *Requirement Exception Handling (R:Exc)*: In an SE project, many unforeseen problematic situations might occur (cf. Prob:ProcExc). Newly created problems might not directly show up and be obscured from their creator. A tool aiming at SE project support should have facilities to detect such complex exceptions and automatically assist humans in handling them.
- *Requirement Quality Management (R:Qual)*: Quality management is a crucial as well as an underestimated part of SE projects. This has been agreed upon in literature (cf. Prob:QualMan) and our practical observations confirmed this, as well. The intangibility of the produced asset (the code) makes it difficult to even be aware of its state. Furthermore, if quality problems are detected, counter measures must be executed in alignment with the process. If a tool shall provide holistic support for SE projects, it must be capable of supporting these complex tasks.
- *Requirement Knowledge Management (R:Know)*: As SE projects are knowledge-intensive undertakings with a multitude of different complex information, the latter is not easy to manage. This is confirmed by literature (cf. Prob:Knowl) and practice (cf. the scenario in this chapter). A tool aiming at holistic SE project support must enable the collection, management and dissemination of that knowledge in a process-centered and context-sensitive manner.

4.3. Requirements Verification

We have conducted a comprehensive literature study comprising many aspects of SE projects. One reason for this was to support the elicitation of requirements for a framework providing comprehensive support for SE projects. However, the study also included a myriad of tools and approaches aiming at the support of different aspects of SE projects. In this section we discuss how the results of this study can be used to verify the requirements we have elicited.

We have examined approaches of different areas that relate to the topics identified as important for this thesis. An important area are Software Engineering Environments (SEEs). These are tools aiming at comprehensibly supporting SE projects. In this area, we have examined various CASE (Computer-Aided Software Engineering) tools (e.g., [EKS93]), Process-Centered Software Engineering Environments (PCSEEs, e.g., [BFGL94, CLH95, BEM94, Barg92b]), modern SE Environments (e.g., [dZR+04, JYW07, HaLa10, WEB+09, dFOT10]), and other contemporary SE approaches (e.g., [BWHK12, PVPB12, GTS10, CAG12]). For a thorough discussion we refer to Chapter 6.

Another important area are processes and their automated enactment. In this area, we have examined WfMS/PAIS (e.g., [Cumb07, Inta15, vdtH05]), process configuration approaches (e.g., [RSS10, Gott09, HBR10, LDH09]), artifact-centric process approaches (e.g., [BHS09, KüRe11a]), process adaptation approaches (e.g., [Wesk01, SMO00, WRWR09, MTS08]), semantic process annotation approaches (e.g., [Mich15, PDB+08, AFKK07, BGM07, ABB+07]), declarative process approaches (e.g., [Pesi08]), and approaches for contextual process integration (e.g., [LSH+06, DGD07]). Besides that we also took into account context modeling approaches like [KMK+03, FaCl04, GPZ04] (see Chapters 7 and 8 for details). In the context of dynamic processes, we have also reviewed approaches for process exception handling (e.g., [MGR04]). Such approaches are discussed in more detail in Chapter 11.

We have also examined various approaches from other areas identified as important for SE projects. These are knowledge management approaches (e.g., [BjDi08, Liao03, BWT04]) and collaboration and coordination approaches (e.g., [LeBo07, BSV07, Dust04]). For a more thorough discussion we refer to Chapters 10 and 12. Furthermore, we examined different approaches for SE quality management. These included approaches for software metric application (e.g., [OfJe97, GKMK02]), software measurement tools (e.g., [ScJe06, LiZh05]), and approaches for the Goal-Question-Metric (GQM) technique (e.g., [FaWu09, STS05, HuFa05]). For more information on these see Chapter 9.

Relevance and Completeness

The various approaches examined show the relevance of the elicited requirements for SE projects relating to various application cases. The need for tool support for SE processes (R:AutoProc) is confirmed by the various SEE approaches. Automated process support in general is the target of countless PAIS and WfMS approaches. The importance for contextual integration (R:ContInt) is discussed by various SEEs as well. According to them, such information comprises artifacts, various types of knowledge, or persons and their interaction.

As many approaches confirm, processes also need to be handled dynamically (R:DynProc). On one hand, various SEEs cover this topic and provide capabilities to change running processes. On the other, there exist a myriad of approaches for configuring or changing processes. Such approaches even offer the capability to automatically change running processes. This is often used for handling exceptions occurring during process enactment. This also confirms that process exception handling (R:Exc) is a relevant topic for a tool that automates processes. In addition to that, much attention has been paid to unstructured processes that are not pre-planned as part of a process model. Constraint-based and declarative process approaches deal primarily with such processes. This confirms the importance of capabilities of a tool to also cover such processes (R:ProcCoverage).

A crucial factor for any SE projects is quality management (R:Qual). This is confirmed not only by approaches explicitly dealing with this topic, but also by many SEE approaches that take into account source code artifacts and aim at supporting and improving their management. The same applies for collaboration and coordination support (R:Coord). Many specific approaches stress the importance of this topic for SE. In addition to this, various SEEs also integrate facilities to support this. Another important area for SE projects is knowledge management (R:Know). This is confirmed both by various dedicated approaches as well as the integration of knowledge management capabilities in many SEEs.

The goal of our approach cannot be to solve each and every problem in SE. Therefore, the requirements also cannot be considered as complete for SE. However, we can show that the selected requirement areas cover important aspects also mentioned in a myriad of other approaches and that those approaches do not discuss or cover important areas that we have omitted. SEEs have existed for multiple decades now and each of them covers different areas and capabilities. However, topics that repeatedly occur are the following: processes, with a strong focus on dynamicity as well as people and collaboration aspects. Furthermore, they deal with various entities that can be considered as context to the tools and processes, as, e.g., artifacts and people. Furthermore, they deal with different kinds of knowledge that is crucial to SE projects. To the best of our knowledge, these SEEs do not cover other core aspects that we have omitted in our discussion. Contemporary SE approaches, however, show two trends gaining momentum: cloud-based SE and global SE, which both correspond to each other. We have decided to put this not to focus in this work as it can be considered primarily as a technical aspect. Our requirement areas are more focused on content-related issues of SE projects.

Relatedness and Generalization

The various approaches we have reviewed not only show that our requirements are relevant for SE projects, they can also serve as indicator that they are related to each other and that their combination is essential for successful SE projects. Again, SEEs serve best as comparative approaches as they share the same goal as our approach. These approaches often have a strong focus on the SE processes and they connect it with various other areas. None of them combines all of them but multiple approaches respectively combine it with contextual data, collaboration support, knowledge management, and quality management relating the SE artifacts. For a fine grained discussion of the different features of different approaches see Chapter 15.

In the first place, our approach is targeted at SE projects. This means neither the approach nor its requirements can be automatically seen as generally applicable for all domains. However, SE is a vast field and not necessarily a distinct domain. Software is developed in various domains like the

automotive or the healthcare sectors. Furthermore, in SE slightly different approaches to project and process management are utilized. Some projects apply huge and heavyweight process models with hundreds of controlled artifacts. Others apply lightweight and agile approaches that prescribe hardly anything and mostly rely on people. The various approaches and tools we have reviewed also show this diversity. They are applied in various domains like the automotive or the healthcare sectors or are applied in projects regulated by state authorities. Furthermore, they involve all the different approaches to project and process management that are prevalent in SE (e.g., Scrum or V-Model XT). Therefore, we assume that the requirements we elicited are applicable for the vast majority of SE projects regardless of their domain or process approach. Furthermore, in the evaluation of this thesis, we will show the application of our approach to slightly different process approaches for SE and also to a process of the software modernization domain.

4.4. Summary

This chapter elicited eight basic requirements for a tool that aims to provide holistic project and process support for SE projects (cf. Table 4-1). These requirements are aligned with the abstract problems discussed in Section 1.1. To further illustrate the requirements and demonstrate their practical relevance, a set of concrete scenarios was presented, which will be taken into account for validating the developed approach (cf. Chapter 13).

Table 4-1: High level requirements

Requirement Area	Requirement ID	Description	Detailing Chapter
Basic functionality	Requirement R:AutoProc	Automated process enactment / implementation	6
Basic functionality	Requirement R:ContInt	Contextual integration of process enactment	7
Basic functionality	Requirement R:DynProc	Dynamic process enactment	7
Extended functionality	Requirement R:ProcCoverage	Extended process coverage	8
Extended functionality	Requirement R:Coord	Task coordination	10
Extended functionality	Requirement R:Exc	Process exception handling	11
Specific functionality	Requirement R:Qual	Quality management integration	9
Specific functionality	Requirement R:Know	Knowledge management integration	12

The requirements are categorized as follows: Basic functionality requirements cover the basic facilities a tool must provide to holistically support the SE process. They do not refer to functionalities providing additional value to humans. However, they are crucial and constitute the basis for the other functionalities. Extended functionality requirements cover functionalities that enable general automatic and contextual support for humans in various areas of a project. The third area refers to specific functionality requirements and is targeted to specific areas of SE projects. Note that not all possible areas of an SE project are covered in this work as this would go beyond the scope of this thesis. Rather, the focus is on two areas of knowledge and quality management as these have been proven important parts of each SE project.

Part IV

Conclusion

16.Summary and Outlook

SE projects are complex, long-running, and knowledge-intensive, depending on a myriad of different factors that are not easily controllable. Furthermore, the developed product, i.e., the software, constitutes an intangible asset whose quality state cannot be measured easily. This places pressure on the knowledge workers in SE projects. Many aspects of the projects, their process, and the produced product are implicitly managed and prone to forgetfulness or other errors.

Due to these various issues, SE projects have always been problematic. From the beginning of SE until today many projects have exceeded their budgets and schedules, delivered low-quality erroneous software, or even failed completely. To make projects more repeatable as well as to support their execution, SE process models have been developed. This started in the 1970s with classical models such as the Waterfall Model [Royc70] or the Spiral Model [Boeh88]. However, these SE models often were too rigid and could not mirror the dynamic SE project execution in reality. More recently, the agile trend took account of this property and agile processes like Scrum [ScBe01] have been popularized. These developments have improved the situation, but still many projects struggle with time, resources and software quality.

A remaining problem concerns the operational support for the projects, their processes, and, first and foremost, the involved software engineers. Projects and their processes are often planned up-front and their execution does not match this plan, resulting in an ever-growing gap between plan and reality. Software engineers utilize a large set of SE tools supporting different tasks like IDEs, source control management systems, or bug trackers. However, the complexity of SE projects keeps growing; e.g., the sizes of projects keep growing, the different tools are often rather complicated, and their number grows as well. Moreover, holistic SE support is missing. Tools may comprehensively support a specific task, but are not connected well to the other tools. Much is still left to the software engineers without providing any guidance to them. Although their collaboration is crucial, it has not been properly supported or governed yet. Crucial project knowledge remains only in the heads of the humans, and is not properly stored, managed and disseminated among the project participants.

This work presents a holistic approach to support both SE projects and, especially, SE processes. In the following we will briefly summarize the core contributions of this thesis:

Automated process support: The CPM framework provides an infrastructure for comprehensive SE project and SE process support. It unites different state-of-the-art technologies encapsulated in loosely-coupled components. The set of components comprises, among others, dedicated components for process enactment, context integration, or knowledge management. Thus, it not only enables the modeling and enactment of workflows, but also the extension of the workflows with a myriad of additional data sets that support the implementation of entire SE process models. As SE process enactment is known to be complex and dynamic, the CPM framework comprises additional basic components, enabling the simple definition and execution of configurable automatisms to support humans in recurring standard situations. It further enables CPM to cope with various dynamic situations whose exact configuration and course might not be estimated a priori.

Context integration: The execution of an SE project and SE process depend on various contextual factors, like the properties of the executing humans or the states of involved artifacts. The CPM framework integrates different facilities to deal with such information. First, it features a set of sensors that can be integrated into various SE tools to automatically gather information. Second it enables the automatic processing of such information to derive meaningful information from the numerous events happening in an SE project. Third, by an extended process specification, it enables the direct and tight integration of process enactment with the context of the project.

Process dynamicity: SE project execution is rather dynamic and mostly differs to what was planned. Therefore, the CPM framework incorporates dynamic processes. Thus, the different workflows executed in the context of an SE process can be dynamically changed during run-time to adhere to changing situations. However, as the projects comprise many different areas and the set of influential context factors is high, it can be challenging for a human to apply a process adaptation on account of this data. To support this, the CPM framework not only enables manual process adaptations, but also automated and context-aware ones.

Extrinsic process coverage: Process models cover a substantial portion of the work done in an SE project. However, many workflows cannot be covered by them for various reasons. Such workflows are characterized by three main properties. First, they cannot be completely foreseen. Second, they are rather dynamic. Third, they depend on their context even more than the ones belonging to the SE process models. Therefore, the CPM framework incorporates a declarative and dynamic way of modeling such workflows that allows directly integrating contextual influences. Furthermore, it enables a uniform way of enacting them similarly to imperative workflows.

Quality assurance integration: Quality assurance is a crucial part of any SE project. However, many projects struggle with bad source code quality. Therefore, the CPM framework integrates facilities to automatically measure the source code quality and to distribute software quality measures to the software engineers in case of quality problems. This comprises a monitoring of the human activities to be able to find the right point in the process for inserting a software quality measure as well as a dynamic tailoring of the latter to select the right measure for the right human and situation. Finally, the CPM framework automatically assesses the applied measures to optimize the measure distribution over time.

Collaboration and coordination: The collaboration of the involved knowledge workers constitutes a crucial part of any SE project. This collaboration might get complicated and error-prone in large projects. Therefore, the CPM framework integrates facilities to support such collaboration in two ways. First, it fosters automated information distribution, informing one human about important changes to their environment as, for example, the status of the activities of their colleagues. Second, it is capable of automatically initiating follow-up activities for certain changes in a project impacting other humans.

Process exception handling: In an SE project many things do not work exactly as planned. Many exceptions might occur relating to the process, its activities, the involved humans, or the processed artifacts. The CPM framework uses its contextual infrastructure to detect as many of these complex exceptions as possible. Furthermore, it is capable of automatically determining an exception handling procedure and distributing it to the appropriate human to apply it.

Knowledge provisioning: SE projects largely depend on the knowledge of the humans involved possess. However, the management and distribution of such knowledge remains a challenge. The CPM framework fosters gathering, storing, and managing of such knowledge. Furthermore, due to its process- and context-related capabilities, it is capable of automatically distributing knowledge to project participants that matches their current situation and problems.

The CPM framework delivers a set of functions we believe to be unique. It unites various areas like dynamic process management, human assistance, and quality management. In these areas, however, there exist specific approaches as well. For example, [HMMR14] focuses on supporting the human by providing process visualizations and additional information. Others support enactment of parts of processes for specific humans based on process views [KoRe13]. The knowledge worker is supported by various approaches as well. Examples include flexible checklist support [MuRe14] or mobility support of knowledge workers by approaches like [PMR14]. Another area is quality management for processes with approaches like [LoRe15]. All of these approaches have their strengths and go beyond the capabilities of the CPM framework in a specific area. The main strength of the latter is, however,

is the comprehensive, applicable and usable integration of a large set of different areas to better support humans in SE projects.

The CPM framework presented in this work solves many problems of SE projects as it provides holistic SE project and SE process support. However, there exist various options for further improving and extending the approach. In the following we will highlight some of the most important ways, the CPM framework might be extended.

We have already discussed and created a set of extensions and additions to the CPM framework not directly being part of this work. One of these extensions is related to the modeling of the contextually extended processes. In the CPM framework, humans model the workflows directly in the WfMS and the different extensions in a web GUI. Such modeling would be simplified if humans had modeled the complete process in one tool and notation. To enable this, we have already created a preliminary approach for a SE workflow language comprising all necessary properties and can then be automatically transformed into the workflows of a WfMS and the additional contextual extensions applied in the CPM framework. For further reading on this topic, we refer to [GOR11a].

In SE, the assessment of processes and their improvement is a crucial topic as well. Process assessment and improvement approaches like ISO 15504 (SPICE) and CMMI (for more information regarding these two, see [Wall07]) have therefore received much attention. Thus, an integration of such approaches into an approach for SE project and SE process support is desirable. We have already created such an extension of the CPM framework enabling the semi-automatic assessment of an executed SE process with models like CMMI or SPICE (see [GOR12e, GOR13]).

Another interesting option will be the application of the CPM framework in other domains as standard SE projects. In the future, with a set of specific other sensors, an application in other knowledge-intensive domains becomes possible. We have already started to investigate such options. As we did not have the resources to develop a completely new set of sensors, we investigated a type of project that can rely on the sensors we already have. In [GOR14] we have discussed the application of the CPM framework in the context of specific software modernization projects.

The preliminary evaluation showed that our concepts have great potential for really supporting SE projects. Regarding the industrial application, however, a larger scale industrial evaluation remains as a future task. To enable the latter, we will need to add various features. For being usable in large scale productive projects, a consistent privacy approach will be necessary. Furthermore, some of the applied technologies will have to be adapted to ensure performance and scalability in larger industrial environments.

Bibliography

[ABB+07] Alazeib, A., Balogh, A., Bauer, M., Bouras, A., Friesen, A., Gouvas, P., Mentzas, G., Pace, A.: Towards semantically-assisted design of collaborative business processes in EAI scenarios. Proc 5th IEEE Int'l Conf on Industrial Informatics, pp. 779-784, 2007.

[Abde88] Abdel-Hamid, T.K.: The economics of software quality assurance: A simulation-based case study. MIS Quarterly, 12(3): 395-411, 1988.

[ACF97] Ambriola, V., Conradi, R., Fuggetta, A.: Assessing process-centered software engineering environments. ACM Transactions on Software Engineering and Methodology, 6(3): 283-328, 1997.

[ADOV02] Arbaoui, S., Derniame, J.-C., Oquendo, F., Verjus, H.: A comparative review of process-centered software engineering environments. Annals of Software Engineering, 14(1-4): 311-340, 2002.

[AFdK11] Aleixo, F.A., Freire, M.A., dos Santos, W.C., Kulesza, U.: Automating the variability management, customization and deployment of software processes: A model-driven approach. Enterprise Information Systems. Springer, pp. 372-387, 2011.

[AFKK07] Abramowicz, W., Filipowska, A., Kaczmarek, M., Kaczmarek, T.: Semantically enhanced business process modelling notation. Proc Workshop on Semantic Business Process and Product Lifecycle Management, pp. 88-91, 2007.

[AHM+08] Ayewah, N., Hovemeyer, D., Morgenthaler, J.D., Penix, J., Pugh, W.: Using static analysis to find bugs. IEEE Software, 25(5): 22-29, 2008.

[Ambl02] Ambler, S.: Agile modeling: effective practices for extreme programming and the unified process. John Wiley & Sons, 2002.

[Aon97] AONIX: Software through Pictures/Object Modeling Technique: Creating OMT Models. Aonix, Inc., 1997.

[ATW+14] Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems. Information and Software Technology, 57, pp. 248-276, 2014.

[BaDa02] Bansiya, J., Davis, C.G.: A hierarchical model for object-oriented design quality assessment. IEEE Transactions on Software Engineering 28(1): 4-17, 2002.

[Barg92a] Barghouti, N.S.: Concurrency control in rule-based software development environments. PhD Thesis, Department of Computer Science, Columbia University, 1992.

[Barg92b] Barghouti, N.S.: Supporting cooperation in the Marvel process-centered SDE. Proc 5th ACM SIGSOFT Symposium on Software Development Environments, pp. 21-31, 1992.

[Bass05] Bassil, S.: Workflow technology for complex socio-technical systems. PhD Thesis, University of Montreal, 2005.

[BCCG07] Bendraou, R., Combemale, B., Crégut, X., Gervais, M.P.: Definition of an executable SPEM 2.0. Proc 14th Asia-Pacific Software Engineering Conf, pp. 390-397, 2007.

[BCJ10] Bruneliere, H., Cabot, J., Jouault, F.: Combining model-driven engineering and cloud computing. Proc 4th Workshop on Modeling, Design, and Analysis for the Service, 2010.

[BCL+01] Basili, V., Costa, P., Lindvall, M., Mendonca, M., Seaman, C., Tesoriero, R., Zelkowitz, M.: An experience management system for a software engineering research organization. Proc 26th Annual NASA Software Engineering Workshop, pp. 29-35, 2001.

[BCM+07] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F.: The description logic handbook: theory, implementation, and applications. Cambridge Univ Pr, 2007.

[BDS+99] Beedle, M., Devos, M., Sharon, Y., Schwaber, K., Sutherland, J.: SCRUM: An extension pattern language for hyperproductive software development. Pattern Languages of Program Design 4: 637-651, 1999.

[BDZ10] Begel, A., DeLine, R., Zimmermann, T.: Social media for software engineering. Proc FSE/SDP Workshop on Future of Software Engineering Research, pp. 33-38, 2010.

[Beck00a] Beck, K.: Extreme Programming: Die revolutionäre Methode für Softwareentwicklung in kleinen Teams. Addison-Wesley, 2000.

[Beck00b] Beck, K.: Extreme programming explained: embrace change. Addison-Wesley Professional, 2000.

[Beck03] Beck, K.: Test-driven development: by example. Addison-Wesley Professional, 2003.

[BEK+06] Brockmans, S., Ehrig, M., Koschmider, K., Oberweis, A., Studer, R.: Semantic alignment of business processes. Proc 18th Int'l Conf on Enterprise Information Systems, pp. 191-196, 2006.

[BeKl99] Beckstein, C., Klausner, L.: A planning framework for workflow management. Proc Workshop on Intelligent Workflow and Process Management, 1999.

[BEM94] Belkhatir, N., Estublier, J., Melo, W.: The ADELE-TEMPO experience: an environment to support process modeling and enaction. Software Process Modelling and Technology. Research Studies Press Ltd., pp. 187-222, 1994.

[BFGL94] Bandinelli, S., Fuggetta, A., Ghezzi, C., Lavazza, L.: SPADE: an environment for software process analysis, design, and enactment. Software Process Modelling and Technology. Research Studies Press Ltd., pp. 223-247, 1994.

[BGM07] Bouras, A., Gouvas, P., Mentzas, G.: Enio: An enterprise application integration ontology. 18th Int'l Conf on Database and Expert Systems Applications, pp. 419-423, 2007.

[BGMT89] Boudier, G., Gallo, F., Minot, R., Thomas, I.: An overview of PCTE and PCTE+. 3rd ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, pp. 248-257, 1989.

[BHS09] Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. Handbook of Research on Business Process Management. IGI Global, pp. 503–531, 2009.

[BiCa04] Biegel, G., Cahill, V.: A framework for developing mobile, context-aware applications. Proc 2nd IEEE Conf on Pervasive Computing and Communication, pp. 361-365, 2004.

[BiKa08] Bibeault, B., Kats, Y.: jQuery in Action. Dreamtech Press, 2008.

[BjDi08] Bjornson, F.O., Dingsoyr, T.: Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. Information and Software Technology, 50(11): 1055-1068, 2008.

[BLWR12] Barba, I., Lanz, A., Weber, B., Reichert, M., del Valle, C.: Optimized time management for declarative workflows. Proc Business Process Modeling, Development, and Support Working Conf, pp. 195-210, 2012.

[Boeh88] Boehm, B.W.: A spiral model of software development and enhancement. Computer, 21(5): 61-72, 1988.

[BPEL07] BPEL. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf, 2007. Visited: December 2015.

[BPNS07] Bern, A., Pasi, A., Nikula, U., Smolander, K.: Contextual factors affecting the software development process–an initial view. 2nd AIS SIGSAND European Symposium on Systems Analysis and Design, pp. 1-8, 2007.

[BPR99] Bellifemine, F., Poggi, A., Rimassa, G.: JADE–A FIPA-compliant agent framework. Proc 4th Intl. Conf and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents, pp. 97-108, 1999.

[BrBe11] Braude, E.J., Bernstein, M.E.: Software engineering: modern approaches. J. Wiley & Sons, 2010.

[BRK+06] Bassil, S., Rinderle, S., Keller, R., Kropf, P., Reichert, M.: Preserving the context of interrupted business process activities. Enterprise Information Systems VII. Springer, pp. 149-156, 2006.

[BrNi98] O'Brien, P.D., Nicol, R.C.: FIPA—towards a standard for software agents. BT Technology J, 16(3): 51-59, 1998.

[Broe96] Brockers, A., Differding, C., Threin, G.: The role of software process modeling in planning industrial measurement programs. Proc Int'l Metrics Symposium, pp. 31-40, 1996.

[Broo87] Brooks, F.P., Jr.: No silver bullet: essence and accidents of software engineering. Computer, 20(4): 10-19, 1987.

[Brow09] Browne, P.: JBoss Drools Business Rules. Packt Publishing, 2009.

[BSV07] Blackburn, T., Swatman, P., Vernik, R.: Cognitive dust: A framework that builds from CSCW concepts to provide situated support for small group work. Computer Supported Cooperative Work in Design III. Springer, pp. 1-12, 2007.

[BWHK12] Blankenship, E., Woodward, M., Holliday, G., Keller, B.: Professional team foundation server 2012. John Wiley & Sons, 2012.

[BWHW06] Biffl, S., Winkler, D., Höhn, R., Wetzel, H.: Software process improvement in europe: potential of the new V-modell XT and research issues. Software Process: Improvement and Practice, 11(3): 229-238, 2006.

[BWT04] Barros, M.O., Werner, C.M.L., Travassos, G.H.: Supporting risks in software project management. J Systems and Software, 70(1-2): 21-35, 2004.

[CAG12] Cordeiro, J., Antunes, B., Gomes, P.: Context-based recommendation to support problem solving in software development. Proc 3rd Int'l Workshop on Recommendation Systems for Software Engineering, pp. 85-89, 2012.

[CALO94] Coleman, D., Ash, D., Lowther, B., Oman, P.: Using metrics to evaluate software system maintainability. IEEE Computer, 27(8): 44-49, 1994.

[Case85] Case, A.F.: Computer-aided software engineering (CASE): technology for improving software development productivity. ACM SIGMIS Database, 17(1): 35-43, 1985.

[CBD+94] Canals, G., Boudjlida, N., Derniame, J.C., Godart, C., Lonchamp, J.: ALF: a framework for building process-centred software engineering environments. Software Process Modelling and Technology. Research Studies Press Ltd., Taunton, UK, pp. 153–185, 1994.

[CCI04] Cook, C., Churcher, N., Irwin, W.: Towards synchronous collaborative software engineering. Proc 11th Asia-Pacific Software Engineering Conf, pp. 230-239, 2004.

[CCPP99] Casati, F., Ceri, S., Paraboschi, S., Pozzi, G.: Specification and implementation of exceptions in workflow management systems. ACM Transactions on Database Systems, 24(3):405-451, 1999.

[CDL+04] Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R., Vetere, G.: DL-Lite: practical reasoning for rich DLs. Proc Intl Workshop on Description Logics, pp. 92, 2004.

[CED10] Carmel, E., Espinosa, J.A., Dubinsky, Y.: "Follow the sun" workflow in global software development. J of Management Information Systems, 27(1):17-38, 2010.

[ChCh13] Chawla, M.K., Chhabra, I.: Capturing OO Software metrics to attain quality attributes–a case study. Int J of Scientific & Engineering Research, 4(6): 359-363, 2013.

[ChKe94] Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on Software Engineering, 20(6): 476-493, 1994.

[CHL+94] Conradi, R., Hagaseth, M., Larsen, J.O., Nguyen, M., Munch, B., Westby, P., Zhu, W., Jaccheri, M., Liu, C.: Object-oriented and cooperative process modelling in EPOS. In: Software Process Modelling and Technology. Research Studies Press Ltd., Taunton, UK, pp. 9–32, 1994.

[CHW03] Chen, T., Far, B.H., Wang, Y.: Development of an intelligent agent-based GQM software measurement system. Proc 12th Asian Test Symposium, pp. 188-197, 2003.

[CKd07] Cirilo, E., Kulesza, U., de Lucena, C.J.P.: GenArch-a model-based product derivation tool. Proc Brazilian Symposium on Software Components, Architectures and Reuse, pp. 31-46, 2007.

[CLH95] Conradi, R., Liu, C., Hagaseth, M.: Planning support for cooperating transactions in EPOS. Information Systems, 20(4): 317-336, 1995.

[CMMI10] CMMI Product Team: CMMI for Development, Version 1.3. Technical Report, Carnegie Mellon Software Engineering Institute Pittsburgh, 2010.

[CNGM95] Cugola, G., Di Nitto, E., Ghezzi, C., Mantione, M.: How to deal with deviations during process model enactment. Proc 17th Int'l Conf on Software Engineering, pp. 265-273, 1995.

[CNW89] Chen, M., Nunamaker Jr, J.F., Weber, E.S.: Computer-aided software engineering: present status and future directions. ACM SIGMIS Database, 20(1): 7-13, 1989.

[Cobe15] Cobertura. Website: http://cobertura.github.io/cobertura/. Visited: December 2015

[CoCh06] Cook, C., Churcher, N.: Constructing real-time collaborative software engineering tools using CAISE, an architecture for supporting tool development. Proc 29th Australasian Computer Science Conf, pp. 267-276, 2006.

[CoGa09] Combi, C., Gambini, M.: Flaws in the flow: The weakness of unstructured business process modeling languages dealing with data. Proc On the Move to Meaningful Internet Systems: OTM 2009, pp. 42-59, 2009.

[Cole92] Coleman, D.: Assessing maintainability. Proc Software Engineering Productivity Conf, pp. 525-532, 1992.

[Cope05] Copeland, T.: PMD applied. Centennial Books, 2005.

[CPPM12] Christidis, K., Paraskevopoulos, F., Panagiotou, D., Mentzas, G.: Combining activity metrics and contribution topics for software recommendations. Proc 3rd Int'l Workshop on Recommendation Systems for Software Engineering, pp. 43-46, 2012.

[CrMu03] Crubézy, M., Musen, M.A.: Ontologies in support of problem solving. Handbook on ontologies, pp. 321–341, 2003.

[CSS09] Coman, I.D., Sillitti, A., Succi, G.: A case-study on using an automated in-process software engineering measurement and analysis system in an industrial environment. Proc Int'l Conf on Software Engineering, pp. 89-99, 2009.

[CuGh1998] Cugola, G., Ghezzi, C.: Software processes: a retrospective and a path to the future. Software Process: Improvement and Practice, 4(3): 101-123, 1998.

[Cumb07] Cumberlidge, M.: Business process management with JBoss JBPM: a practical guide for business analysts; Develop Business Process Models for Implementation in a Business Process Management Sytem, Packt Publishing, 2007.

[DAH05] Dumas, M., van der Aalst, W.M., ter Hofstede, A.H.: Process-aware information systems: bridging people and software through process technology. John Wiley & Sons, 2005.

[DaRe09] Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Computer Science - Research & Development, 23(2): 81-97, Springer 2009.

[Dask92] Daskalantonakis, M.K.: A practical view of software measurement and implementation experiences within Motorola. Proc IEEE Transactions on Software Engineering, 18(11): 998-1010, 1992.

[dBBG10] da Silva, M.A.A., Bendraou, R., Blanc, X., Gervais, M.-P.: Early deviation detection in modeling activities of mde processes. In: Model Driven Engineering Languages and Systems. Springer, pp. 303-317, 2010.

[deLe09] de Leoni, M.: Adaptive process management in highly dynamic and pervasive scenarios. Proc 4th European Young Researchers Workshop on Service Oriented Computing, pp. 83–97, 2009.

[Denn12] Denninger, O.: Recommending relevant code artifacts for change requests using multiple predictors. Proc 3rd Int'l Workshop on Recommendation Systems for Software Engineering, pp. 78-79, 2012.

[DeSt90] DeGrace, P., Stahl, L.H.: Wicked problems, righteous solutions: a catalogue of modern software engineering paradigms. Yourdon Press, 1990.

[dFOT10] de Lucia, A., Fasano, F., Oliveto, R., Tortora, G.: Fine-grained management of software artefacts: the ADAMS system. Software: Practice and Experience, 40(11): 1007-1034, 2010.

[DGD07] Dietze, S., Gugliotta, A., Domingue, J.: A semantic web services-based infrastructure for context-adaptive process support. Proc IEEE Int'l Conf on Web Services, pp. 537-543, 2007.

[DHM+96] Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., Zbyslaw, A.: Freeflow: mediating between representation and action in workflow systems. Proc ACM Conf on Computer Supported Cooperative Work, pp. 190-198, 1996.

[Dijk72] Dijkstra, E.W.: Notes on structured programming. Structured programming. Academic Press Ltd., pp. 1-82, 1972.

[dKM97] de Panfilis, S., Kitchenham, B., Morfuni, N.: Experiences introducing a measurement program. Information and Software Technology, 39(11): 745-754, 1997.

[dLPF06] de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The web service modeling language wsml: an overview. The Semantic Web: Research and Applications, pp. 590-604, 2006.

[DoDu07] Dorn, C., Dustdar, S.: Sharing hierarchical context for mobile web services. Distributed and Parallel Databases, 21(1): 85-111, 2007.

[DRGC06] Debnath, N., Riesco, D., Cota, M.P., Perez-Schofield, J.B.G., Uva, D.: Supporting the SPEM with a UML extended workflow metamodel. Proc IEEE Conf on Computer Systems and Applications, pp. 1151-1154, 2006.

[DRK00] Dadam, P., Reichert, M., Kuhn, K.: Clinical workflows—the killer application for process-oriented information systems? Proc Int'l Conf on Business Information Systems, pp. 36–59, Springer 2000.

[DRR+10] Dadam, P., Reichert, M., Rinderle-Ma, S., Lanz, A., Pryss, R., Predeschly, M., Kolb, J., Ly, L.T., Jurisch, M., Kreher, U.: From ADEPT to AristaFlow BPM suite: a research vision has become reality. Proc Business Process Management Workshops, pp. 529-531, 2010.

[DSW06] Davies, J., Studer, R., Warren, P.: Semantic web technologies: trends and research in ontology-based systems, Vol. 3. John Wiley & Sons, 2006.

[Dust04] Dustdar, S.: Caramba—a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. Distributed and Parallel Databases, 15(1): 45-66, 2004.

[DZG10] Döhring, M., Zimmermann, B., Godehardt, E.: Extended workflow flexibility using rule-based adaptation patterns with eventing semantics. LNI P-175, pp. 216–226, 2010.

[dZR+04] de Oliveira, K.M., Zlot, F., Rocha, A.R., Travassos, G.H., Galotta, C., de Menezes, C.S.: Domain-oriented software development environment. J of Systems and Software, 72(2): 145-161, 2004.

[EcFo15] Eclipse Foundation: OpenUP. http://epf.eclipse.org/wikis/openup/. Visited: December 2015

[EHK84] Estublier, J., Ghoul, S., Krakowiak, S.: Preliminary experience with a configuration control system for modular programs. ACM SIGPLAN Notices, 19(5): 149-156, 1984.

[EKR95] Ellis, C., Keddara, K., Rozenberg, G.: Dynamic change within workflow systems. Proc ACM Conf on Organizational Computing Systems, pp. 10-21, 1995.

[EKS93] Emmerich, W., Kroha, P., Schäfer, W.: Object-oriented database management systems for construction of CASE environments. Database and Expert System Applications, pp. 631-642, 1993.

[ELU10] Eberle, H., Leymann, F., Unger, T.: Transactional process fragments-recovery strategies for flexible workflows with process fragments. Proc IEEE Asia-Pacific Services Computing Conf, pp. 250-257, 2010.

[EMMA15] EMMA. Website: http://www.emma.sourceforge.net. Visited: December 2015

[Espe15] Esper. Website: http://www.espertech.com/products/esper.php. Visited: December 2015

[EvVe09] Eveleens, J., Verhoef, C.: Quantifying IT forecast quality. Science of Computer Programming, 74(11-12): 934-988, 2009.

[FaCl04] Fahy, P., Clarke, S.: CASS–a middleware for mobile context-aware applications. Proc Workshop on Context-awareness, 2004.

[FaWu09] FanJiang, Y.Y., Wu, C.H.: Towards a multi-agents architecture for GQM measurement system. Proc 9th Int'l Conf on Hybrid Intelligent Systems, pp. 277-280, 2009.

[Fell13] Fellmann, M.: Semantic process engineering Konzeption und Realisierung eines Werkzeugs zur Semantischen Prozessmodellierung. PhD Thesis, University of Osnabrück, 2013.

[FFM+10] Friedrich, G., Fugini, M., Mussi, E., Pernici, B., Tagni, G.: Exception handling for repair in service-based processes. IEEE Transactions on Software Engineering, 36(2): 198-215, 2010.

[FIRR15] Fdhila, W., Indiono, C., Rinderle-Ma, S., Reichert, M.: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Systems, 49: 1-24, 2015.

[FMZ06] Feng, Y., Mingshu, L., Zhigang, W.: Spem2xpdl: Towards SPEM model enactment. Proc Int'l Conf on Software Engineering Research and Practice, 2006.

[FoHi01] Fowler, M., Highsmith, J.: The agile manifesto. Software Development, 9(8): 28-35, 2001.

[Fox92] Fox, M.: The tove project towards a common-sense model of the enterprise. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, pp. 25-34, 1992.

[FrGr97] Franklin, S., Graesser, A.: Is it an agent, or just a program?: a taxonomy for autonomous agents. Intelligent Agents III Agent Theories, Architectures, and Languages, pp. 21-35, 1997.

[Frie03] Friedman-Hill, E.: JESS in Action. Manning Greenwich, CT, 2003.

[Fros07] Frost, R.: Jazz and the eclipse way of collaboration. Software, IEEE, 24(6): 114-117, 2007.

[Fugg93] Fuggetta, A.: A classification of CASE technology. Computer, 26(12): 25-38, 1993.

[FUSI15] European Project FUSION. http://www.fusionweb.org/FUSION/home.asp. Visited: December 2015

[GAF00] Grüninger, M., Atefi, K., Fox, M.S.: Ontologies to support process integration in enterprise engineering. Computational & Mathematical Organization Theory, 6(4): 381-394, 2000.

[GAJL08] Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable workflow models. Int'l J Cooperative Information Systems, 17(2): 177–221, 2008.

[Gali04] Galin, D.: Software quality assurance: from theory to implementation. Addison-Wesley, 2004.

[GDD06] Gasevic, D., Djuric, D., Devedzic, V.: Model driven architecture and ontology development. Springer-Verlag, 2006.

[GeFi92] Genesereth, M.R., Fikes, R.E., Computer Science Department, S.U.: Knowledge interchange format-version 3.0: reference manual. Technical Report, Department of Computer Science, Stanford University, 1992.

[GeLa81] Genrich, H.J., Lautenbach, K.: System modelling with high-level Petri nets. Theoretical Computer Science, 13(1): 109-135, 1981.

[Gele85] Gelernter, D.: Generative communication in Linda. ACM Transactions on Programming Languages and Systems, 7(1): 80-112, 1985.

[GeTs98] Georgakopoulos, D., Tsalgatidou, A.: Technology and tools for comprehensive business process lifecycle management. Workflow Management Systems and Interoperability, pp. 356-395, 1998.

[GGK06] Gibson, D.L., Goldenson, D.R., Kost, K.: Performance results of CMMI-based process improvement. Technical Report, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, 2006.

[GHM98] Grundy, J.C., Hosking, J.G., Mugridge, W.B.: Coordinating distributed software development projects with integrated process modelling and enactment environments. Proc IEEE 21st Int'l Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 39-44, 1998.

[GHS95] Georgakopoulos, D., Hornick, M., Sheth, A.: An overview of workflow management: From process modeling to workflow automation infrastructure. Distributed and parallel databases, 3(2): 119-153, 1995.

[GKKL12] Grundy, J., Kaefer, G., Keong, J., Liu, A.: Guest editors' introduction: software engineering for the cloud. Software, IEEE, 29(2): 26-29, 2012.

[GKMK02] Gopal, A., Krishnan, M.S., Mukhopadhyay, T., Goldenson, D.R.: Measurement programs in software development: Determinants of success. IEEE Transactions on Software Engineering, 28(9): 863-875, 2002.

[Glas98] Glass, R.L.: Software runaways: monumental software disasters. Prentice Hall, 1997.

[GoAk03] Gordijn, J., Akkermans, J.: Value-based requirements engineering: Exploring innovative e-commerce ideas. Requirements engineering, 8(2): 114-134, 2003.

[GOR10a] Grambow, G., Oberhauser, R., Reichert, M.: Semantic workflow adaption in support of workflow diversity. Proc 4th Int'l Conf on Advances in Semantic Processing, pp. 158-165, 2010.

[GOR10b] Grambow, G., Oberhauser, R., Reichert, M.: Employing semantically driven adaptation for amalgamating software quality assurance with process management. Proc 2nd Int'l Conf on Adaptive and Self-adaptive Systems and Applications, pp. 58-67, 2010.

[GOR11a] Grambow, G., Oberhauser, R., Reichert, M.: Towards a workflow language for software engineering. Proc 10th IASTED Conf on Software Engineering, pp. 130-137, 2011.

[GOR11b] Grambow, G., Oberhauser, R., Reichert, M.: Semantically-driven workflow generation using declarative modeling for processes in software engineering. Proc 4th Int'l Workshop on Evolutionary Business Processes, pp. 164-173, 2011.

[GOR11c] Grambow, G., Oberhauser, R., Reichert, M.: Towards automatic process-aware coordination in collaborative software engineering. Proc 6th Int'l Conf on Software and Data Technologies, pp. 5-14, 2011.

[GOR11d] Grambow, G., Oberhauser, R., Reichert, M.: Contextual injection of quality measures into software engineering processes. Int'l J on Advances in Software, 4(1 & 2): 76-99, 2011.

[GOR11e] Grambow, G., Oberhauser, R., Reichert, M.: Event-driven exception handling for software engineering processes. Proc 5th Int'l Workshop on event-driven Business Process Management, LNBIP 99, pp. 414-426, 2011.

[GOR11f] Grambow, G., Oberhauser, R., Reichert, M.: Towards dynamic knowledge support in software engineering processes Proc 6th Int'l Workshop on Applications of Semantic Technologies, LNI 192, pp. 149, 2011.

[GOR12a] Grambow, G., Oberhauser, R., Reichert, M.: Enabling automatic process-aware collaboration support in software engineering projects. Selected Papers of the ICSOFT'11 Conf, CCIS 303, pp. 73-89, 2012.

[GOR12b] Grambow, G., Oberhauser, R., Reichert, M.: Contextual generation of declarative workflows and their application to software engineering processes. Int'l J on Advances in Intelligent Systems, 4(3 & 4):158-179, 2012.

[GOR12c] Grambow, G., Oberhauser, R., Reichert, M.: User-centric abstraction of workflow logic applied to software engineering processes. Proc 1st Workshop on Human-Centric Process-Aware Information Systems, LNBIP 112, pp. 307-321, 2012.

[GOR12d] Grambow, G., Oberhauser, R., Reichert, M.: Knowledge provisioning: a context-sensitive process-oriented approach applied to software engineering environments. Proc 7th Int'l Conf on Software and Data Technologies, pp. 506-515, 2012.

[GOR12e] Grambow, G., Oberhauser, R., Reichert, M.: Towards automated process assessment in software engineering. Proc 7th Int'l Conf on Software Engineering Advances, pp. 289-295, 2012.

[GOR13] Grambow, G., Oberhauser, R., Reichert, M.: Automated software engineering process assessment: supporting diverse models using an ontology. Int'l J on Advances in Software, 6(1 & 2): 213 - 224, 2013.

[GOR14] Grambow, G., Oberhauser, R., Reichert, M.: Providing automated holistic process and knowledge assistance during software modernization. Uncovering Essential Software Artifacts through Business Process Archaeology. IGI Global, pp. 20-63, 2014.

[GOR16] Grambow, G., Oberhauser, R., Reichert, M.: Context-aware and process-centric knowledge provisioning - an example from the software development domain. Innovations in Knowledge Management. Springer, pp. 179-209, 2016.

[Gott09] Gottschalk, F.: Configurable process models. Ph.D. Thesis. Eindhoven University of Technology, 2009.

[GPZ04] Gu, T., Pung, H.K.: A middleware for building context-aware mobile services. Proc IEEE Vehicular Technology Conf, pp. 2656-2660, 2004.

[GrOb10] Grambow, G., Oberhauser, R.: Towards automated context-aware selection of software quality measures. Proc 5th Intl. Conf on Software Engineering Advances, pp. 347-352, 2010.

[Gruh02] Gruhn, V.: Process-centered software engineering environments, a brief history and future challenges. Annals of Software Engineering, 14(1): 363-382, 2002.

[GTS10] Grammel, L., Treude, C., Storey, M.-A.: Mashup environments in software engineering. Proc 1st Workshop on Web 2.0 for Software Engineering, pp. 24-25, 2010.

[GuAl10] Guha, R., Al-Dabass, D.: Impact of web 2.0 and cloud computing platform on software engineering. Proc Int'l Symposium on Electronic System Design, pp. 213-218, 2010.

[HaLa09] Hattori, L., Lanza, M.: An environment for synchronous software development. ICSE-Companion 2009, pp. 223-226, 2009.

[HaLa10] Hattori, L., Lanza, M.: Syde: A tool for collaborative software development. Proc 32nd Int'l Conf on Software Engineering, pp. 235-238, 2010.

[Hall10] Hallerbach, A.: Management von Prozessvarianten. PhD Thesis. University of Ulm, 2010.

[Hami88] Abdel-Hamid, T.K.: The economics of software quality assurance: A simulation-based case study. MIS Quarterly, 12(3): 395-411, 1988.

[HBR08a] Hallerbach, A., Bauer, T., Reichert, M.: Context-based configuration of process variants. Proc 3rd Int'l Workshop on Technologies for Context-Aware Business Process Management, pp. 31-40, 2008.

[HBR08b] Hallerbach, A., Bauer, T., Reichert, M.: Managing Process Variants in the Process Lifecycle. Proc. 10th Int'l Conf on Enterprise Information Systems, pp. 154-161, 2008.

[HBR10] Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. J of Software Maintenance and Evolution: Research and Practice, 22(6 & 7): 519-546, 2010.

[HBZ+14] Haisjackl, C., Barba, I., Zugal, S., Soffer, P., Hadar, I., Reichert, M., Pinggera, J., Weber, B.: Understanding declare models: strategies, pitfalls, empirical results. Software & Systems Modeling, pp. 1-28, 2014.

[HeMa03] Hevner, A.R., March, S.T.: The information systems research cycle. Computer, 36(11): 111-113, 2003.

[HKR11] Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of semantic web technologies. CRC Press, 2011.

[HLD+05] Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic business process management: A vision towards using semantic web services for business process management. Proc IEEE Int'l Conf on e-Business Engineering, pp. 535-540, 2005.

[HMMR14] Hipp, M., Mutschler, B., Michelberger, B., Reichert, M.: Navigating in process model repositories and enterprise process information. Proc IEEE Eighth Int'l Conf on Research Challenges in Information Science, pp. 1-12, 2014.

[HMPR04] Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly, 28(1): 75-105, 2004.

[HMR12] Hipp, M., Mutschler, B., Reichert, M.: Navigating in complex business processes. Proc Database and Expert Systems Applications, pp. 466-480, 2012.

[Holl93] Hollingsworth, D.: Workflow management coalition: The workflow reference model, 1993.

[HPN08] Hill, J., Pezzini, M., Natis, Y.: Findings: confusion remains regarding BPM terminologies. Gartner Research, 501(G00155817), 2008.

[HuBo06] Huang, L.G., Boehm, B.: How much software quality investment is enough: A value-based approach. IEEE Software, 23(5): 88-95, 2006.

[HuFa05] Huang, J., Far, B.H.: Intelligent software measurement system (ISMS). Proc Canadian Conf on Electrical and Computer Engineering, pp. 1033-1036, 2005.

[IABG15] IABG: Das V-Modell. Release 2.0 http://www.v-modell.iabg.de/, 2015. Visited: December 2015

[IBM08] IBM Jazz, 2008. Website: http://www.jazz.net. Visited: December 2015

[IEEE02] IEEE standard for software quality assurance plans, IEEE Std 730-2002, 2002.

[IEEE04] IEEE Standard for software verification and validation, IEEE Std 1012-2004, 2004.

[IEEE05] IEEE standard for software configuration management plans, IEEE Std 828-2005 2005.

[IEEE07] IEEE standard for software and system test documentation, IEEE Std P829-2007, 2007.

[IEEE09] IEEE standard for information technology - systems design - software design descriptions, IEEE Std 1016-2009, 2009.

[IEEE98a] IEEE recommended practice for software requirements specifications, IEEE Std 830-1998, 1998.

[IEEE98b] IEEE standard for software project management plans, IEEE Std 1058-1998, 1998.

[Inta15] Intalio BPMS, www.intalio.com. Visited: December 2015

[JaCo93] Jaccheri, M.L., Conradi, R.: Techniques for process model evolution in EPOS. IEEE Transactions on Software Engineering, 19(12): 1145-1156, 1993.

[JaSo04] Jang, M., Sohn, J.-C.: Bossam: An extended rule engine for OWL inferencing. Rules and Rule Markup Languages for the Semantic Web. Springer, pp. 128-138, 2004.

[JBR99] Jacobson, I., Booch, G., Rumbaugh, J.: The unified software development process. Addison-Wesley, 1999.

[Jhaw15] Jhawk. Website: http://www.virtualmachinery.com/jhawkprod.htm. Visisted: April 2015

[John07] Johnson, P.M.: Requirement and design trade-offs in Hackystat: An in-process software engineering measurement and analysis system. Proc 1st Int'l Symposium on Empirical Software Engineering and Measurement, pp. 81-90, 2007

[Jone10] Jones, C.: Get software quality right. Dr Dobb's Journal, 2010.

[Jone96] Jones, C.: Strategies for managing requirements creep. Computer, 29(6): 92-94, 1996.

[JPSW94] Junkermann, G., Peuschel, B., Schäfer, W., Wolf, S.: MERLIN: Supporting cooperation in software development through a knowledge-based environment. Software Process Modeling and Technology. Research Studies Press, pp. 103–129, 1994.

[JYW07] Jiang, T., Ying, J., Wu, M.: CASDE: An environment for collaborative software development. In: Computer Supported Cooperative Work in Design III, pp. 367-376, 2007.

[JYWF06] Jiang, T., Ying, J., Wu, M., Fang, M.: An architecture of process-centered context-aware software development environment. Proc 10th Int'l Conf on Computer Supported Cooperative Work in Design, pp. 1-5, 2006.

[Kan02] Kan, S.H.: Metrics and models in software quality engineering. Pearson Education India, 2003.

[KeHa02] Kess, P., Haapasalo, H.: Knowledge creation through a project review process in software production. Int'l J of Production Economics, 80(1): 49-55, 2002.

[KeMu06] Kersten, M., Murphy, G.C.: Using task context to improve programmer productivity. Proc 14th ACM SIGSOFT Int'l Symposium on Foundations of Software Engineering, pp. 1-11, 2006.

[KHB00] Kiepuszewski, B., ter Hofstede, A., Bussler, C.: On structured workflow modelling. Proc 12th Conf on Advanced Information Systems Engineering, pp. 431-445, 2000.

[KHKRS08] Hitzler, P., Krötzsch, M., Rudolph, S.: York Sure: Semantic Web - Grundlagen. Springer, 2008.

[Kind06] Kindler, E.: On the semantics of EPCs: Resolving the vicious circle. Data & Knowledge Engineering, 56(1): 23-40, 2006.

[KKKM00] Krishnan, M.S., Kriebel, C.H., Kekre, S., Mukhopadhyay, T.: An empirical analysis of productivity and quality in software products. Management Science, pp. 745-759, 2000.

[KKL+04] Kloppmann, M., Konig, D., Leymann, F., Pfau, G., Roller, D.: Business process choreography in WebSphere: Combining the power of BPEL and J2EE. IBM Systems J, 43(2): 270-296, 2004.

[KMK+03] Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., Malm, E.J.: Managing context information in mobile devices. IEEE Pervasive Computing, 2(3): 42-51, 2003.

[KNS92] Keller, G., Nüttgens, M., Scheer, A.W.: Semantische Prozessmodellierung auf der Grundlage “Ereignisgesteuerter Prozessketten (EPK)”. Veröffentlichungen des Instituts für Wirtschaftsinformatik, 89, pp. 1992.

[Kolo92] Kolodner, J.L.: An introduction to case-based reasoning. Artificial Intelligence Review, 6(1): 3-34, 1992.

[KoOb05] Koschmider, A., Oberweis, A.: Ontology based business process description. Proc CAiSE´05 Workshops, pp. 321-333, 2005.

[KoRe13] Kolb, J., Reichert, M.: A flexible approach for abstracting and personalizing large business process models. ACM SIGAPP Applied Computing Review, 13(1): 6-18, 2013.

[Kreh14] Kreher, U.: Konzepte, Architektur und Implementierung adaptiver Prozessmanagementsysteme. PhD thesis. University of Ulm, 2014.

[KRFR13] Knuplesch, D., Reichert, M., Fdhila, W., Rinderle-Ma, S.: On enabling compliance of cross-organizational business processes. Proc Business Process Management 2013, pp. 146-154, 2013.

[KRL+13] Knuplesch, D., Reichert, M., Ly, L.T., Kumar, A., Rinderle-Ma, S.: Visual modeling of business process compliance rules with the support of multiple perspectives. Proc Int'l Conf Conceptual Modeling, pp. 106-120, 2013.

[KRM+12] Knuplesch, D., Reichert, M., Mangler, J., Rinderle-Ma, S., Fdhila, W.: Towards compliance of cross-organizational processes and their changes. Proc Business Process Management Workshops, pp. 649-661, 2013.

[Kruc04] Kruchten, P.: The rational unified process: an introduction. Addison-Wesley Professional, 2004.

[Kruc99] Kruchten, P.: Der Rational Unified Process. Eine Einführung. Addison-Wesley, 1999.

[KuJe04] Kurniawati, F., Jeffery, R.: The long-term effects of an EPG/ER in a small software organisation. Proc Australian Software Engineering Conf, pp. 128-136, 2004.

[Künz13] Künzle, V.: Object-aware process management. PhD Thesis. University of Ulm, 2013.

[KüRe11a] Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object aware process management. J of Software Maintenance and Evolution: Research and Practice, 23(4): 205-244, 2011.

[KüRe11b] Künzle, V., Reichert, M.: Striving for Object-aware Process Support: How existing approaches fit together. Proc 1st Int'l Symposium on Data-driven Process Discovery and Analysis, pp. 169-188, 2011.

[KüRe11c] Künzle, V., Reichert, M.: A Modeling paradigm for integrating processes and data at the micro level. Proc 12th Int'l Working Conf on Business Process Modeling, Development and Support, LNBIP 81, pp. 201-215, 2011.

[KVL+08] Karastoyanova, D., Van Lessen, T., Leymann, F., Ma, Z., Nitzsche, J., Wetzstein, B., Bhiri, S., Hauswirth, M., Zaremba, M.: A reference architecture for semantic business process management systems. Proc Multikonferenz Wirtschaftsinformatik, pp. 1727-1738, 2008.

[KVV06] Krötzsch, M., Vrandecic, D., Völkel, M.: Semantic mediawiki. Proc Int'l Semantic Web Conf, pp. 935-942, 2006.

[KWR11] Künzle, V., Weber, B., Reichert, M.: Object-aware business processes: fundamental requirements and their support in existing approaches. Int'l J of Information System Modeling and Design, 2(2):19-46, 2011.

[Lava00] Lavazza, L.: Providing automated support for the GQM measurement process. IEEE Software, 17(3): 56-62, 2000.

[LavBa05] Lavazza, L., Barresi, G.: Automated support for process-aware definition and execution of measurement plans. Proc 27th Int'l Conf on Software Engineering, pp. 234-243, 2005.

[LBW07] Liu, R., Bhattacharya, K., Wu, F.Y.: Modeling business contexture and behavior using business artifacts. Adv. Inform. Syst. Eng. 4495, pp. 324–339, 2007.

[LDH09] La Rosa, M., L.R., Dumas, M., ter Hofstede, A.H.M.: Modelling business process variability for design-time configuration. Handbook of Research on Business Process Modeling. Idea Group Inc, pp. 204-228, 2009.

[LDTM11] La Rosa, M., Dumas, M., Ter Hofstede, A.H.M., Mendling, J.: Configurable multi-perspective business process models. Information Systems, 36(2): 313-340, 2011.

[LeBo07] Lewandowski, A., Bourguin, G.: Enhancing support for collaboration in software development environments. Computer Supported Cooperative Work in Design III, pp. 160-169, 2007.

[LeFr07] Lemcke, J., Friesen, A.: Composing web-service-like abstract state machines (ASMs). Proc IEEE Congress on Services, 2007.

[LEPV10] Lanubile, F., Ebert, C., Prikladnicki, R., Vizcaíno, A.: Collaboration tools for global software engineering. Software, IEEE, 27(2): 52-55, 2010.

[LeRe07] Lenz, R., Reichert, M.: IT support for healthcare processes-premises, challenges, perspectives. Data & Knowledge Engineering, 61(1): 39-58, 2007.

[LeRo00] Leymann, F., Roller, D.: Production workflow: concepts and techniques. Prentice Hall, 2000.

[Liao03] Liao, S.: Knowledge management technologies and applications--literature review from 1995 to 2002. Expert systems with applications, 25(2): 155-164, 2003.

[LiCo93] Liu, C., Conradi, R.: Automatic replanning of task networks for process model evolution in EPOS. Proc European Software Engineering Conf, pp. 434-450, 1993.

[LiRu02] Lindvall, M., Rus, I.: Knowledge management in software engineering. IEEE Software, 19(3): 26-38, 2002.

[LiSt05] Lin, Y., Strasunskas, D.: Ontology-based semantic annotation of process templates for reuse. Proc 10th Int'l Workshop on Exploring Modeling Methods for Systems Analysis and Design, pp. 593-604, 2005.

[LiZh05] Li, Z., Zhou, Y.: PR-Miner: automatically extracting implicit programming rules and detecting violations in large software code. ACM SIGSOFT Software Engineering Notes 30: 306-315, 2005.

[LKRD10] Lanz, A., Kreher, U., Reichert, M., Dadam, P.: Enabling process support for advanced applications with the aristaFlow BPM suite. Proc. Int'l Conf. on Business Process Management Demonstration Track, 2010.

[Lofi05] Lofi, C.: cGQM - Ein zielorientierter Ansatz für kontinuierliche, automatisierte Messzyklen. Proc 4th National Conf on Software Measurement and Metrics, 2005.

[LoRe13] Lohrmann, M., Reichert, M.: Understanding business process quality. In: Business Process Management - Theory and Applications. Studies in Computational Intelligence (444), pp. 41-73, Springer, 2013.

[LoRe15] Lohrmann, M., Reichert, M.: Effective application of process improvement patterns to business processes. Software & Systems Modeling, 2015.

[LoRo93] Lott, C.M., Rombach, H.D.: Measurement-based guidance of software projects using explicit project plans. Information and Software Technology, 35(6-7): 407-419, 1993.

[LPCR13] Lanz, A., Posenato, R., Combi, C., Reichert, M.: Controllability of time-aware processes at run time. Proc. Int'l Conference on Cooperative Information Systems, pp. 39-56, 2013.

[LPCR15] Lanz, A., Posenato, R., Combi, C., Reichert, M.: Simple temporal networks with partially shrinkable uncertainty. Proc. Int'l Conf on Agents and Artificial Intelligence, pp. 10 - 12, 2015.

[LPR12] Lenz, R., Peleg, M., Reichert, M.: Healthcare process support: achievements, challenges, current research. Int'l J of Knowledge-Based Organizations, 2(4), 2012.

[LRD10a] Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. Proc Int'l Conf on Advanced Information Systems Engineering, pp. 9-23, 2010.

[LRD10b] Lanz, A., Reichert, M., Dadam, P.: Making business process implementations flexible and robust: error handling in the AristaFlow BPM suite. Proc CAiSE'10 Forum, LNBIP 72, pp. 174-189, 2010.

[LRKD11] Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. Proc Int'l Conf on Cooperative Information Systems, pp. 82-99, 2011.

[LRW08] Li, C., Reichert, M., Wombacher, A.: On measuring process model similarity based on high-level change operations. Proc Int'l Conf Conceptual Modeling, pp. 248-264, 2008.

[LRW09] Li, C., Reichert, M., Wombacher, A.: Discovering reference models by mining process variants using a heuristic approach. Proc Business Process Management, pp. 344-362, 2009.

[LRW10] Li, C., Reichert, M., Wombacher, A.: The MinAdept clustering approach for discovering reference process models out of process variants. Int'l J of Cooperative Information Systems, 19(3 & 4): 159-203, 2010.

[LRW11] Li, C., Reichert, M., Wombacher, A.: Mining business process variants: challenges, scenarios, algorithms. Data & Knowledge Engineering, 70(5): 409-434, 2011.

[LSH+06] Lin, Y., Strasunskas, D., Hakkarainen, S., Krogstie, J., Solvberg, A.: Semantic annotation framework to manage semantic heterogeneity of process models. Advanced Information Systems Engineering, pp. 433-446, 2006.

[LSKM00] Luo, Z., Sheth, A., Kochut, K., Miller, J.: Exception handling in workflow systems. Applied Intelligence, 13(2): 125-147, 2000.

[LTXF04] Liang, L., Tang, Y., Xiao, W., Feng, Z.: The literature review of cooperative software engineering. Proc 8th Int'l Conf on Computer Supported Cooperative Work in Design, 2004., 1, pp. 648-652, 2004.

[Luck01] Luckham, D.C.: The power of events: an introduction to complex event processing in distributed enterprise systems. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2001.

[LWR14] Lanz, A., Weber, B., Reichert, M.: Time patterns for process-aware information systems. Requirements engineering, 19(2): 113-141, 2014.

[Ly13] Ly, L.T.: SeaFlows–a compliance checking framework for supporting the process lifecycle. PhD Thesis, University of Ulm, 2013.

[Mall09] Mall, R.: Fundamentals of software engineering. PHI Learning Pvt. Ltd., 2009.

[MaRa12] Matinnejad, R., Ramsin, R.: An analytical review of process-centered software engineering environments. Proc IEEE 19th Int'l Conf and Workshops on Engineering of Computer Based Systems, pp. 64-73, 2012.

[MaSa13] Mahmood, Z., Saeed, S.: Software engineering frameworks for the cloud computing paradigm. Springer, 2013.

[MaSm95] March, S.T., Smith, G.F.: Design and natural science research on information technology. Decision Support Systems, 15(4): 251-266, 1995.

[MaVe03] MacCormack, A., Verganti, R.: Managing the sources of uncertainty: matching process and context in software development. J of Product Innovation Management, 20(3): 217-232, 2003.

[MBG+02] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A., Oltramari, R., Schneider, L., Horrocks, I.: Wonderweb deliverable d17. The Wonderweb Library of Foundational Ontologies and the Dolce Ontology. 2002.

[MBR15] Mundbrod, N., Beuter, F., Reichert, M.: Supporting knowledge-intensive processes through integrated task lifecycle support. Proc. 19th IEEE Int'l Enterprise Distributed Object Computing Conf, pp. 19-28, 2015.

[McBr02] McBride, B.: Jena: A semantic web toolkit. IEEE Internet Computing, 6(6): 55-59, 2002.

[McCa76] McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering, pp. 308-320, 1976.

[McCo01] McConnell, S.: The nine deadly sins of project planning. IEEE Software, 18(5): 5-7, 2001.

[McVa04] McGuinness, D.L., van Harmelen, F.: OWL web ontology language overview. W3C recommendation, 2004.

[MDA08] Mendling, J., van Dongen, B.F., van der Aalst, W.M.P.: Getting rid of OR-joins and multiple start events in business process models. Enterprise Information Systems, 2(4): 403-419, 2008.

[MdMR09] Maciel, R.S.P., da Silva, B.C., Magalhães, P.F., Rosa, N.S.: An integrated approach for model driven process modeling and enactment. Proc 13th Brazilian Symposium on Software Engineering, pp. 104-114, 2009.

[Meie09] Meier, W.: eXist: an open source native XML database. Web, Web-Services, and Database Systems, pp. 169-183, 2009.

[Mend08] Mendling, J.: Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness. Springer-Verlag New York Inc, 2008.

[MGDM10] Moha, N., Guéhéneuc, Y.G., Duchien, L., Le Meur, A.F.: DECOR: A method for the specification and detection of code and design smells. IEEE Transactions on Software Engineering, pp. 20-36, 2009.

[MGR04] Müller, R., Greiner, U., Rahm, E.: Agentwork: a workflow system supporting rule-based workflow adaptation. Data Knowlage Engineering, 51(2): 223-256, 2004.

[MHHR06] Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. Proc 4th Int'l Conf on Business Process Management, pp. 368-377, 2006.

[Mich15] Michelberger, B.: Process-oriented information logistics: Aligning process information with business processes. PhD Thesis. Ulm University, 2015.

[Micr15] MicroTOOL in-Step. Website: http://www.microtool.de/instep/en/index.asp. Visisted: April 2015

[MMS14] Marrella, A., Mecella, M., Sardina, S.: SmartPM: an adaptive process management system through situation calculus, IndiGolog, and classical planning. Proc. Int'l Conf on Principles of Knowledge Representation and Reasoning, pp. 1-10, 2014.

[MNA10] Mendling, J., Neumann, G., van der Aalst, W.M.P.: Understanding the occurrence of errors in process models based on metrics. Proc On the Move to Meaningful Internet Systems, pp. 113-130, 2010.

[MoAm94] Montangero, C., Ambriola, V.: Oikos: constructing process-centred SDEs. Software Process Modelling and Technology. Research Studies Press Ltd., pp. 131-151, 1994.

[MOMO08] MOMOCS Project, Deliverable D3.1b "Methodology Specification", 2008.

[Mont10] Montali, M.: Specification and verification of declarative open interaction models. Springer, 2010.

[Morg02] Morgan, T.: Business rules and information systems: aligning IT with business goals. Addison-Wesley Professional, 2002.

[Mori99] Morisio, M.: Measurement processes are software too. J of Systems and Software, 49(1): 17-31, 1999.

[MPA+10] Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative specification and verification of service choreographiess. ACM Transactions on the Web, 4(1):1-62, 2010.

[MRH08] Müller, D., Reichert, M., Herbst, J.: A new paradigm for the enactment and dynamic adaptation of data-driven process structures. Proc Int'l Conf Advanced Information Systems Engineering, pp. 48-63, 2008.

[MRv10] Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). Information and Software Technology, 52(2): 127-136, 2010.

[MSG13] Manchanda, D., Singh, A., Garg, N.: An insight upon the effect of quality assurance on the cost of software development. Int'l J of Computer Applications, 80, pp. 4-10, 2013.

[MTB07] Minor, M., Tartakovski, A., Bergmann, R.: Representation and structure-based similarity assessment for agile workflows. Case-Based Reasoning Research and Development, pp. 224-238, 2007.

[MTM+07] Meier, J., Taylor, J., Mackman, A., Bansode, P., Jones, K.: Team development with Microsoft Visual Studio team foundation server: patterns & practices. Microsoft Press, 2007.

[MTS08] Minor, M., Tartakovski, A., Schmalen, D.: Agile workflow technology and case-based change reuse for long-term processes. Int'l J of Intelligent Information Technologies, 4(1): 80-98, 2008.

[Müll02] Müller, R.: Event-oriented dynamic adaptation of workflows: model, architecture and implementation. PhD Thesis. University of Leipzig, 2002.

[MuRe14] Mundbrod, N., Reichert, M.: Process-aware task management support for knowledge-intensive business processes: findings, challenges, requirements. Proc IEEE 18th Int'l Enterprise Distributed Object Computing Conf Workshops and Demonstrations, pp. 116-125, 2014.

[MWA+07] Ma, Z., Wetzstein, B., Anicic, D., Heymans, S., Leymann, F.: Semantic business process repository. Proc Workshop on Semantic Business Process and Product Lifecycle Management, pp. 92–100, 2007.

[NaRa68] Naur, P., Randell, B.: Software Engineering: Report on a conference sponsored by the NATO SCIENCE COMMITTEE. Scientific Affairs Division, NATO, 1968.

[NaSt07] Namiri, K., Stojanovic, N.: A model-driven approach for internal controls compliance in business processes. Proc Workshop on Semantic Business Process and Product Lifecycle Management, pp. 40, 2007.

[NCF+03] Noy, N.F., Crubézy, M., Fergerson, R.W., Knublauch, H., Tu, S.W., Vendetti, J., Musen, M.A.: Protege-2000: an open-source ontology-development and knowledge-acquisition environment. Proc AMIA Annual Symposium, 953, pp. 953, 2003.

[Nejm98] Nejmeh, B.A.: NPATH: a measure of execution path complexity and its applications. Communications of the ACM, 31(2): 188-200, 1988.

[NSIH02] Nemati, H.R., Steiger, D.M., Iyer, L.S., Herschel, R.T.: Knowledge warehouse: an architectural integration of knowledge management, decision support, artificial intelligence and data warehousing. Decision Support Systems, 33(2): 143-161, 2002.

[NUS05] Nystrom, N.A., Urbanic, J., Savinell, C.: Understanding productivity through non-intrusive instrumentation and statistical learning. Proc 2nd Workshop on Productivity and Performance in High-End Computing, 2005.

[NWV07] Nitzsche, J., Wutke, D., Van Lessen, T.: An ontology for executable business processes. Proc Workshop on Semantic Business Process and Product Lifecycle Management, 2007.

[OASI07] Web Services Business Process Execution Language (WSBPEL) TC. Web services business process execution language version 2.0 committee specification. http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpelv2.0-CS01.pdf, January 2007

[Ober10] Oberhauser, R.: Leveraging semantic web computing for context-aware software engineering environments. Semantic Web. In-Tech, Vienna, Austria, pp. 157-179, 2010.

[ObSc07] Oberhauser, R., Schmidt, R.: Towards a holistic integration of software lifecycle processes using the semantic web. Proc 2nd Int'l Conf on Software and Data Technologies, pp. 137-144, 2007.

[OEGQ07] Olague, H.M., Etzkorn, L.H., Gholston, S., Quattlebaum, S.: Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Transactions on Software Engineering, pp. 402-419, 2007.

[OfJe97] Offen, R.J., Jeffery, R.: Establishing software measurement programs. IEEE Software, 14(2): 45-53, 1997.

[OHA91] Oman, P., Hagemeister, J., Ash, D.: A definition and taxonomy for software maintainability. Technical Report, Software Engineering Test Laboratory, University of Idaho, 1991.

[OMG08] Object Management Group: Software & systems process engineering meta-model specification 2.0, 2008.

[OMG11a] OMG: Business Process Modeling Notation (BPMN) Version 2. 2011.

[OMG11b] OMG: Unified modeling language (UML) Version 2.4. 2011.

[OSGI15] OSGi-Alliance: OSGi-The dynamic module system for java. http://www.osgi.org. Visited: December 2015.

[OYS04] Ohira, M., Yokomori, R., Sakai, M., Matsumoto, K., Inoue, K., Torii, K.: Empirical project monitor: a tool for mining multiple project data. Proc Int'l Workshop on Mining Software Repositories, 2004.

[PaLa06] Paradauskas, B., Laurikaitis, A.: Business knowledge extraction from legacy information systems. Information Technology and Control, 35(3): 214-221, 2006.

[Part10] Partsch, H.A.: Requirements Engineering systematisch. Springer, 2010.

[Pato99] Paton, N.W.: Active rules in database systems. Springer Verlag, 1999.

[PDA08] Pedrinaci, C., Domingue, J., Alves de Medeiros, A.: A core ontology for business process analysis. In: The Semantic Web: Research and Applications, pp. 49-64, 2008.

[PDB+08] Pedrinaci, C., Domingue, J., Brelage, C., Van Lessen, T., Karastoyanova, D., Leymann, F.: Semantic business process management: Scaling up the management of business processes. Proc IEEE Int'l Conf on Semantic Computing, pp. 546-553, 2008.

[Pesi08] Pesic, M.: Constraint-based workflow management systems: shifting control to users. Technische Universiteit Eindhoven, 2008.

[Peter81] Peterson, J.L.: Petri net theory and the modeling of systems. Prentice Hall PTR, 1981.

[Pevd06] Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. Proc Business Process Management Workshops, pp. 169-180, 2006.

[PMR14] Pryss, R., Musiol, S., Reichert, M.: Integrating mobile tasks with business processes: a self-healing approach. In: Handbook of Research on Architectural Trends in Service-Driven Computing, pp. 103–135, 2014.

[PNL02] Pease, A., Niles, I., Li, J.: The suggested upper merged ontology: a large ontology for the semantic web and its applications. Proc AAAI Workshop on Ontologies and the Semantic Web, 28, 2002.

[PrSe06] Prud, E., Seaborne, A.: SPARQL query language for RDF. W3C WD 4, 2006.

[PSA07] Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: Declare: Full support for loosely-structured processes. Proc 11th IEEE Int'l Enterprise Distributed Object Computing Conf, pp. 287-298, 2007.

[PSSA07] Pesic, M., Schonenberg, M., Sidorova, N., van der Aalst, W.M.P.: Constraint-based workflow models: change made easy. Proc 15th Int'l Conf on Cooperative Information Systems, pp. 77-94, 2007.

[PTR+10] Pietschmann, S., Tietz, V., Reimann, J., Liebing, C., Pohle, M., Meißner, K.: A metamodel for context-aware component-based mashup applications. Proc 12th Int'l Conf on Information Integration and Web-based Applications & Services, pp. 413-420, 2010.

[PVPB12] Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., Beecham, S.: Tools used in global software engineering: a systematic mapping review. Information and Software Technology, 54(7): 663-685, 2012.

[PWZ+11] Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A.: Imperative versus declarative process modeling languages: an empirical investigation. Proc 2nd Int'l Workshop on Empirical Research in Business Process Management, pp. 383-394, 2011.

[RAH06] Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Exception handling patterns in process-aware information systems. Proc 23rd Int'l Conf on Advanced Information Systems Engineering, pp. 288-302, 2006.

[RaTi99] Ramesh, B., Tiwana, A.: Supporting collaborative process knowledge management in new product development teams. Decision Support Systems, 27(1-2): 213-235, 1999.

[RBH07] Ralyté, J., Brinkkemper, S., Henderson-Sellers, B.: Situational method engineering: fundamentals and experiences. Springer, 2007.

[RBTK05] Rausch, A., Bartelt, C., Ternité, T., Kuhrmann, M.: The V-Modell XT applied–model-driven and document-centric development. Proc 3rd World Congress for Software Quality, pp. 131-138, 2005.

[RCBM10] Richardson, I., Casey, V., Burton, J., McCaffery, F.: Global software engineering: a software process approach. Proc Collaborative Software Engineering, pp. 35-56, 2010.

[RDB03] Reichert, M., Dadam, P., Bauer, T.: Dealing with forward and backward jumps in workflow management systems. Software and Systems Modeling, 2(1): 37-58, 2003.

[RDR+09] Reichert, M., Dadam, P., Rinderle-Ma, S., Jurisch, M., Kreher, U., Göser, K.: Architecural principles and components of adaptive process management technology. LNI P-151. pp. 81-97, 2009.

[RDRL09] Reichert, M., Dadam, P., Rinderle-Ma, S., Lanz, A., Pryss, R., Predeschly, M., Kolb, J., Ly, L.T., Jurisch, M., Kreher, U.: Enabling poka-yoke workflows with the aristaflow BPM suite. Proc Int'l Conf. on Business Process Management Demonstration Track, 2009.

[ReDa09] Reichert, M., Dadam, P.: Enabling adaptive process-aware information systems with ADEPT2. Handbook of Research on Business Process Modeling, pp. 173-203, 2009.

[ReDa98] Reichert, M., Dadam, P.: ADEPTflex—supporting dynamic changes of workflows without losing control. J of Intelligent Information Systems, 10(2): 93-129, 1998.

[Reic00] Reichert, M.: Dynamische Ablaufänderungen in Workflow-Management-Systemen. PhD Thesis, University of Ulm, 2000.

[ReMe08] Reijers, H., Mendling, J.: Modularity in process models: review and effects. Proc Int'l Conf on Business Process Management, pp. 20-35, 2008.

[ReRi06] Reichert, M., Rinderle, S.: On design principles for realizing adaptive service flows with BPEL. Proc Workshop Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen, pp. 133–146, 2006.

[ReRo98] Reisig, W., Rozenberg, G.: Lectures on Petri nets I: Basic models. Springer Berlin Heidelberg, 1998.

[ReWe12] Reichert, M., Weber, B.: Enabling flexibility in process-aware information systems – challenges, methods, technologies. Springer, 2012.

[ReWe13] Reichert, M., Weber, B.: Process change patterns: recent research, use cases, research directions. Seminal Contributions to Information Systems Engineering. Springer, pp. 397-404, 2013.

[RHB15] Reichert, M., Hallerbach, A., Bauer, T.: Lifecycle management of business process variants. Handbook on Business Process Management 1. Springer, pp. 251-278, 2015.

[RHD98] Reichert, M., Hensinger, C., Dadam, P.: Supporting adaptive workflows in advanced application environments. Proc EDBT Workshop on Workflow Management Systems, pp. 100-109, 1998.

[RHEA04a] Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow resource patterns. Technical Report, Eindhoven Univ. of Technology, 2004.

[RHEA04b] Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow data patterns. Proc 24th Int'l Conf on Conceptual Modeling, pp. 353–368, 2004.

[RHM06] Russell, N., ter Hofstede, A.H.M., Mulyar, N.: Workflow controlflow patterns: a revised view. Technical Report, BPM Center, 2006.

[RiJa00] Rising, L., Janoff, N.S.: The Scrum software development process for small teams. IEEE Software, 17(4): 26-32, 2000.

[RKL+05] Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web service modeling ontology. Applied Ontology, 1(1): 77-106, 2005.

[RLS07] La Rosa, M., Lux, J., Seidel, S., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-driven configuration of reference process models. Proc 19th Int'l Conf on Advanced Information Systems Engineering, pp. 424-438, 2007.

[RoAa05] Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Information Systems, 32(1): 1-23, 2005.

[Royc70] Royce, W.W.: Managing the development of large software systems. Proc IEEE WESCON, pp. 1-9, 1970.

[RRD03] Reichert, M., Rinderle, S., Dadam, P.: Adept workflow management system. Proc Business Process Management. Springer, pp. 370-379, 2003.

[RRD09] Reichert, M., Rinderle-Ma, S., Dadam, P.: Flexibility in process-aware information systems. Transactions on Petri Nets and Other Models of Concurrency II, pp. 115-135, 2009.

[RRKD05] Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive process management with ADEPT2. Proc 21st Int'l Conf on Data Engineering, pp. 1113-1114, 2005.

[RSS09] Reinhartz-Berger, I., Soffer, P., Sturm, A.: Organisational reference models: supporting an adequate design of local business processes. Int. J. Business Process Integration and Management, 4(2): 134–149, 2009.

[RSS10] Reinhartz-Berger, I., Soffer, P., Sturm, A.: Extending the adaptability of reference models. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 40(5): 1045–1056, 2010.

[Rudo08] Rudolph, G.: Some guidelines for deciding whether to use a rules engine. http://herzberg.ca.sandia.gov/guidelines.shtml, 2008. Visited: December 2015

[RWZ10] Robillard, M., Walker, R., Zimmermann, T.: Recommendation systems for software engineering. IEEE Software, 27(4): 80-86, 2010.

[SBBK08] Schaffert, S., Bry, F., Baumeister, J., Kiesel, M.: Semantic wikis. IEEE Software, 25(4): 8-11, 2008.

[ScBe01] Schwaber, K., Beedle, M.: Agile software development with Scrum, Vol. 18. Prentice Hall, 2001.

[Sche01] Scheer, A.-W.: ARIS-Modellierungsmethoden, Metamodelle, Anwendungen. Springer, 2001.

[Schw97] Schwaber, K.: Scrum development process. In: Business Object Design and Implementation. Springer, pp. 117-134, 1997.

[ScJe06] Schlesinger, F., Jekutsch, S.: ElectroCodeoGram: An environment for studying programming. TeamEthno-online, 2, pp. 30-31, 2006.

[Scot02] Scott, K.: The unified process explained. Addison-Wesley Professional, 2002.

[SGWB12] Schneider, K., Gartner, S., Wehrmaker, T., Brugge, B.: Recommendations as learning: from discrepancies to software improvement. Proc 3rd Int'l Workshop on Recommendation Systems for Software Engineering, pp. 31-32, 2012.

[Shet97] Sheth, A.: From contemporary workflow process automation to adaptive and dynamic work activity coordination and collaboration. Siggroup Bulletin, 18(3): 24-27, 1997.

[SHK98] Slaughter, S.A., Harter, D.E., Krishnan, M.S.: Evaluating the cost of software quality. Communications of the ACM, 41(8): 67-73, 1998.

[SHT06] Schobbens, P.Y., Heymans, P., Trigaux, J.C.: Feature diagrams: a survey and a formal semantics. Requirements Engineering, pp. 136–145, 2006.

[SiLu12] Silva, E.A.N.d., Lucrédio, D.: Software engineering for the cloud: a research roadmap. Proc 26th Brazilian Symposium on Software Engineering, pp. 71-80, 2012.

[Simo96] Simon, H.A.: The sciences of the artificial. MIT press, 1996.

[SMH08] Shearer, R., Motik, B., Horrocks, I.: Hermit: A highly-efficient owl reasoner. Proc 5th Intl Workshop on OWL: Experiences and Directions, pp. 26-27, 2008.

[SMO00] Sadiq, S., Marjanovic, O., Orlowska, M.: Managing change and time in dynamic workflow processes. Int. J Coop Inf Systems 9(1&2): 93–116, 2000.

[Sodh91] Sodhi, J.: Software engineering: methods, management, and CASE tools. McGraw-Hill, 1991.

[SPG+07] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics: Science, Services and Agents on the World Wide Web, 5(2): 51-53, 2007.

[Spin05] Spinellis, D.: Tool writing: a forgotten art? (software tools). IEEE Software, 22(4): 9-11, 2005.

[SQTR07] de Souza, C.R., Quirk, S., Trainer, E., Redmiles, D.F.: Supporting collaborative software development through the visualization of socio-technical dependencies. Proc 2007 Int'l ACM Conf on Supporting Group Work, pp. 147-156, 2007.

[SSO01] Sadiq, S., Sadiq, W., Orlowska, M.: Pockets of flexibility in workflow specification. Proc Conceptual Modeling, pp. 513-526, 2001.

[SST05] Seyyedi, M., Shams, F., Teshnehlab, M.: A new method for measuring software processes within software capability maturity model based on the fuzzy multi-agent measurements. Proc World Academy Of Science, Engineering and Technology Vol. 4, pp. 257-262, 2005.

[Stan11] Stanierowski, M.: Evaluierung des kollaborativen Lifecycle-Managements mit der IBM Jazz-Plattform, Vol. 1. epubli, 2011.

[STDC10] Storey, M.-A., Treude, C., van Deursen, A., Cheng, L.-T.: The impact of social media on software engineering practices and tools. Proc FSE/SDP Workshop on Future of Software Eengineering Research, pp. 359-364, 2010.

[STS05] Seyyedi, M.A., Teshnehlab, M., Shams, F.: Measuring software processes performance based on the fuzzy multi agent measurements. Proc Intl Conf on Information Technology: Coding and Computing - Volume II, pp. 410-415, 2005.

[STT06] Soini, J., Tenhunen, V., Tukiainen, M.: Current practices of measuring quality in Finnish software engineering industry. Software Process Improvement, pp. 100-110, 2006.

[SUPE09] European Project SUPER - Semantics utilised for process management within and between enterprises. http://cordis.europa.eu/ist/kct/super_synopsis.htm /, 2009. Visited: December 2015

[SWGF10] Šmite, D., Wohlin, C., Gorschek, T., Feldt, R.: Empirical evidence in global software engineering: a systematic review. Empirical Software Engineering, 15(1): 91-118, 2010.

[TaNo86] Takeuchi, H., Nonaka, I.: The new new product development game. Harvard Business Review, 64(1): 137-146, 1986.

[TFB00] Teigland, R., Fey, C.F., Birkinshaw, J.: Knowledge dissemination in global R&D operations: an empirical study of multinationals in the high technology electronics industry. Management International Review, pp. 49-77, 2000.

[ThFe06a] Thomas, O., Fellmann, M.: Semantic event-driven process chains. Proc Workshop on Semantics for Business Process Management, 2006.

[ThFe06b] Thomas, O., Fellmann, M.: Semantische Ereignisgesteuerte Prozessketten. Integration, Informationslogistik und Architektur: DW2006, pp. 205-224, 2006.

[TsHo06] Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. Automated Reasoning, LNCS 4130, pp. 292-297, 2006.

[UKMZ98] Uschold, M., King, M., Moralee, S., Zorgios, Y.: The enterprise ontology. The Knowledge Engineering Review, 13(1): 31-89, 1998.

[UsGr96] Uschold, M., Gruninger, M.: Ontologies: principles, methods and applications. Knowledge engineering review, 11(2), pp. 93-136, 1996.

[vBCR02] van Solingen, R., Basili, V., Caldiera, G., Rombach, H.D.: Goal question metric (gqm) approach. Encyclopedia of Software Engineering, pp. 578-583, 2002.

[VBG12] Valetto, G., Blincoe, K., Goggins, S.P.: Actionable identification of emergent teams in software development virtual organizations. Proc 3rd Int'l Workshop on Recommendation Systems for Software Engineering, pp. 11-15, 2012.

[vdAa04] van der Aalst, W.M.P.: Business process management: a personal view. Business Process Management J, 10(2): 248-253, 2004.

[vdAa11] van der Aalst, W.M.P.: Process mining: discovery, conformance and enhancement of business processes. Springer, 2011.

[vdAa98] van der Aalst, W.M.P.: The application of petri nets to workflow management. J of Circuits, Systems, and Computers, 8(1): 21-66, 1998.

[vdBa02] van der Aalst, W.M.P., Basten, T.: Inheritance of workflows: an approach to tackling problems related to change. Theoretical Computer Science, 270(1-2): 125-203, 2002.

[vdHe09] van den Heuvel, W.-J.: Aligning modern business processes and legacy systems: a component-based perspective. The MIT Press, 2009.

[VdS03] Villela, K., de Oliveira, K.M., Santos, G., Rocha, A.R., Travassos, G.H.: Cordis-FBC: an enterprise-oriented software development environment. Proc Wissensmanagement, pp. 91-96, 2003.

[vdtH05] van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: yet another workflow language. Information Systems, 30(4): 245-275, 2005.

[vdvH02] van der Aalst, W.M.P., van Hee, K.M.: Workflow management: models, methods, and systems. The MIT press, 2004.

[vdWe04] van der Aalst, W.M.P., Weijters, A.J.M.M.: Process mining: a research agenda, Computers in Industry, 53(3): 231-244, 2004.

[Visa94] Visaggio, G.: Process improvement through data reuse. IEEE Software, 11(4): 76-85, 1994.

[vtKB03] van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases, 14(1): 5-51, 2003.

[vtW03] van der Aalst, W., ter Hofstede, A., Weske, M.: Business process management: a survey. Business Process Management, LNCS 2678, pp. 1019-1031, 2003.

[VVK08] Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Proc 6th Int'l Conf on Business Process Management, pp. 100-115, 2008.

[VVK09] Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data & Knowledge Engineering, 68(9): 793-818, 2009.

[vWM04] van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering, 16(9): 1128-1142, 2004.

[Wall07] Wallmüller, E.: SPI-Software Process Improvement mit CMMI und ISO 15504. Hanser Verlag, 2007.

[Wass90] Wasserman, A.: Tool integration in software engineering environments. Software Engineering Environments, pp. 137-149, 1990.

[WBB04] Wainer, J., Bezerra, F., Barthelmess, P.: Tucupi: A flexible workflow system based on overridable constraints. Proc 2004 ACM Symposium on Applied Computing, pp. 498-502, 2004.

[WEB+09] Weber, S., Emrich, A., Broschart, J., Ras, E., Ünalan, Ö.: Supporting software development teams with a semantic process-and artifactoriented collaboration environment. Proc Software Engineering (Workshops), pp. 243-254, 2009.

[Wesk00] Weske, M.: Workflow management systems: Formal foundation, conceptual design, implementation aspects. Habil Thesis, University of Munster, 2000.

[Wesk01] Weske, M.: Formal foundation and conceptual design of dynamic adaptations in a workflow management system. Proc Hawaii Int'l Conf on System Sciences, 2001.

[WMF+07] Wetzstein, B., Ma, Z., Filipowska, A., Kaczmarek, M., Bhiri, S., Losada, S., Lopez-Cobo, J.M., Cicurel, L.: Semantic business process management: A lifecycle based requirements analysis. Proc Workshop on Semantic Business Process and Product Lifecycle Management, pp. 1613-1673, 2007.

[WPZW10] Weber, B., Pinggera, J., Zugal, S., Wild, W.: Alaska simulator toolset for conducting controlled experiments on process flexibility. Proc CAiSE'10 Forum, LNBIP 72, pp. 205-221, 2011.

[WRMR11] Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. Computers in Industry, 62(5): 467-486, 2011.

[WRR08] Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features-enhancing flexibility in process-aware information systems. Data & Knowledge Engineering, 66(3): 438-466, 2008.

[WRW05] Weber, B., Rinderle, S., Wild, W., Reichert, M.: CCBR–driven business process evolution. Proc Int'l Conf on Cased based Reasoning, pp. 610-624, 2005.

[WRWR09] Weber, B., Reichert, M., Wild, W., Rinderle-Ma, S.: Providing integrated life cycle support in process-aware information systems. Int'l J of Cooperative Information Systems, 18(1): 115-165, 2009.

[WSR09] Weber, B., Sadiq, S., Reichert, M.: Beyond rigidity–dynamic process lifecycle support. Computer Science-Research and Development, 23(2): 47-65, 2009.

[WTA+08] Weber, S., Thomas, L., Armbrust, O., Ras, E., Rech, J., Uenalan, Ö., Wessner, M., Linnenfelser, M., Decker, B.: A software organization platform (SOP). Proc 10th Workshop on Learning Software Organizations, Rome, Italy, 2008.

[WWB04] Weber, B., Wild, W., Breu, R.: CBRFlow: Enabling adaptive workflow management through conversational case-based reasoning. Proc European Conf on Case-Based Reasoning, pp. 89-101, 2004.

[WWW04a] World Wide Web Consortium, OWL web ontology language semantics and abstract syntax, 2004.

[WWW04b] World Wide Web Consortium, Resource description framework (RDF) Concepts and Abstract Syntax, 2004

[WWW04c] World Wide Web Consortium, 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission

[WZP+14] Weber, B., Zeitelhofer, S., Pinggera, J., Torres, V., Reichert, M.: How advanced change patterns impact the process of process modeling. Proc BPMDS 14, pp. 17-32, 2014.

[Yau11] Yau, S.S., An, H.G.: Software engineering meets services and cloud computing. IEEE Computer, 44(10): 47-53, 2011.

[YaWe06] Yang, Q., Li, J.J., Weiss, D.: A survey of coverage based testing tools. Proc Intl. Workshop on Automation of Software Testing, pp. 99-103, 2006.

[Your03] Yourdon, E.: Death march. Prentice Hall PTR, 2004.

[ZaLe03] Zamli, K.Z., Lee, P.A.: Modeling and enacting software processes using VRPML. Proc 10th Asia-Pacific Software Engineering Conf, pp. 243-252, 2003.

[ZCL05] Zhao, X., Chan, K., Li, M.: Applying agent technology to software process modeling and process-centered software engineering environment. Proc ACM Symposium on Applied Computing, pp. 1529-1533, 2005.

[ZdR02] Zlot, F., de Oliveira, K.M., Rocha, A.R.: Modeling task knowledge to support software development. Proc 14th Int'l Conf on Software Engineering and Knowledge Engineering, pp. 35-42, 2002.

[ZIK05] Zamli, K.Z., Isa, N.A.M., Khamis, N.: The design and implementation of the VRPML support environments. Malaysian J of Computer Science, 18(1): 57-69, 2005

[ZPW11a] Zugal, S., Pinggera, J., Weber, B.: Creating declarative process models using test driven modeling suite. Proc CAiSE Forum, pp. 1-8, 2011.

[ZPW11b] Zugal, S., Pinggera, J., Weber, B.: The impact of testcases on the maintainability of declarative process models. Proc Int'l Working Conf on Enterprise, Business-Process and Information Systems Modeling, LNBIP 81, pp. 163-177, 2011.

[ZPW12] Zugal, S., Pinggera, J., Weber, B.: Toward enhanced life-cycle support for declarative processes. J of Software: Evolution and Process, 24(3): 285-302, 2012.

[ZSH+15] Zugal, S., Soffer, P., Haisjackl, C., Pinggera, J., Reichert, M., Weber, B.: Investigating expressiveness and understandability of hierarchy in declarative business process models. Software & Systems Modeling, 14(3): 1081-1103, 2015.

[zuRe08] zur Muehlen, M., Recker, J.: How much language is enough? Theoretical and practical use of the business process modeling notation. Proc Int'l Conf on Advanced Information Systems Engineering, pp. 465-479, 2008.

Acronyms

ALM	Application Lifecycle Management
API	Application Programming Interface
BPMN	Business Process Modeling Notation
CASE	Computer-Aided Software Engineering
CEP	Complex Event Processing
CPM	Context-aware Process Management
ECA	Event Condition Action
GI	Guidance Item
GKPI	Goal Key Performance Indicator
GQM	Goal Question Metric
GUI	Graphical User Interface
IDE	Integrated Development Environment
IS	Information System
KPI	Key Performance Indicator
MDA	Model-Driven Architecture
MDE	Modell-Driven Engineering
MVC	Model-View-Controller
OWL	Web Ontology Language
PAIS	Process-Aware Information System
PCSEE	Process-Centered Software Engineering Environment
SEE	Software Engineering Environment
QKPI	Question Key Performance Indicator
SME	Small and Medium-sized Enterprise
SPEM	Software & Systems Process Engineering Metamodel specification
SQA	Software Quality Assurance
WfMS	Workflow Management System

Part V

Appendices

A. Ontology

The ontology has been modeled and accessed only using the Protégé ontology editor. In the following, we will be present a small set of exemplary concepts of the ontology as modeled in OWL XML. Some of the properties have been renamed due to the fact that names in an OWL ontology are unique and one such property as e.g. ‘workUnitContTempl’ should not be used to connect a work unit container template to a project template and to a work unit template but rather two distinct properties are required here, in this case, ‘workUnitContTempl’, and ‘containingWUCT’.

A.1. Imperative Process Concepts

This section presents the discussed concepts for the imperative processes.

A.1.1. Template Concepts

This section presents the template concepts. First, Figure A-1 gives an overview of them (barely readable and just as visual overview) directly from the Protégé ontology editor. After that, for an exemplary concept, the XML definition from the ontology is shown (cf. Listing A-1). Keep in mind that the properties of the concepts are not directly part of the concepts. However, the restrictions on a selection of properties can be seen.

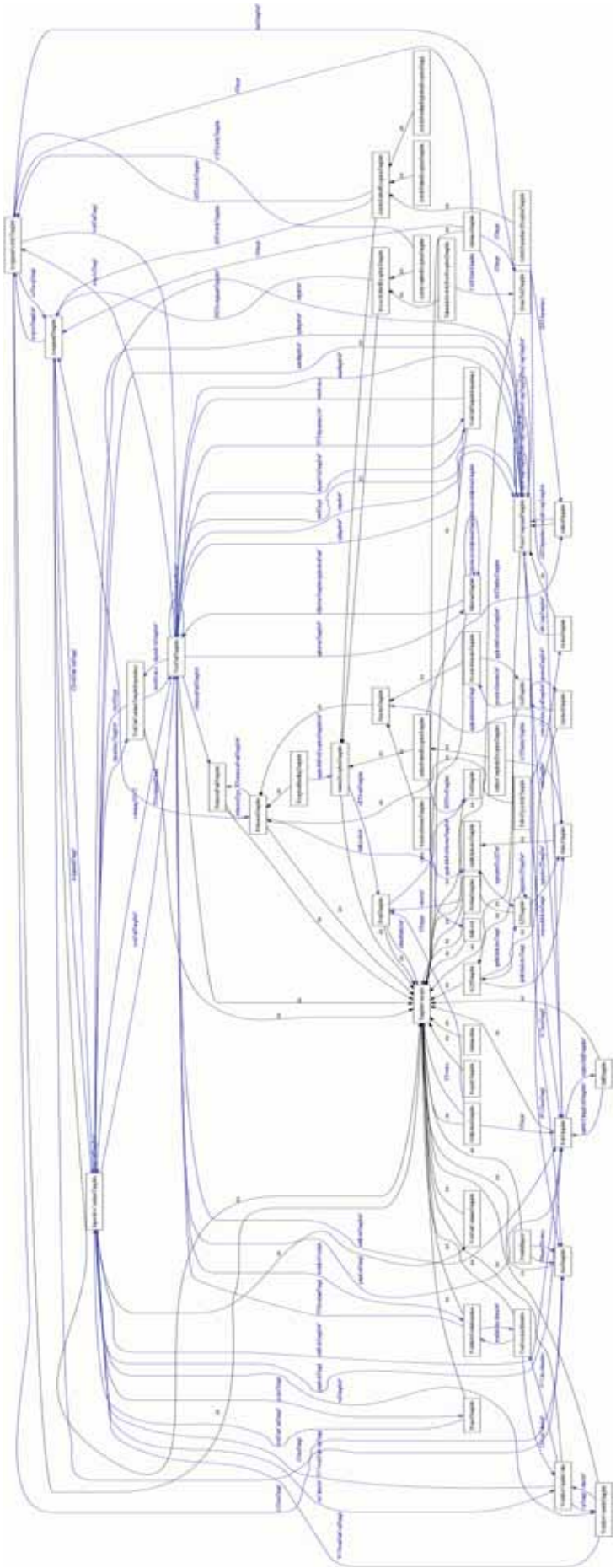

Figure A-1: Imperative template concepts in ontology

Listing A-1 (Work unit template)

```
<owl:Class rdf:ID="WorkUnitTemplate">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="workflowUserInfo"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="assignActTempl"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="repeatable"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="omittable"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="TemplateConcepts"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
      >1</owl:maxCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="extensionPointTemplSet"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="actTempl"/>
      </owl:onProperty>
      <owl:cardinality
         rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="containingWUCT"/>
```

```
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="primRoleTempl"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

A.1.2. Individual Concepts

This section presents an excerpt of the individual concepts discussed (cf. Listing A-2), preceded by a (barely readable) visual overview of the concepts and their connections in Figure A-2.

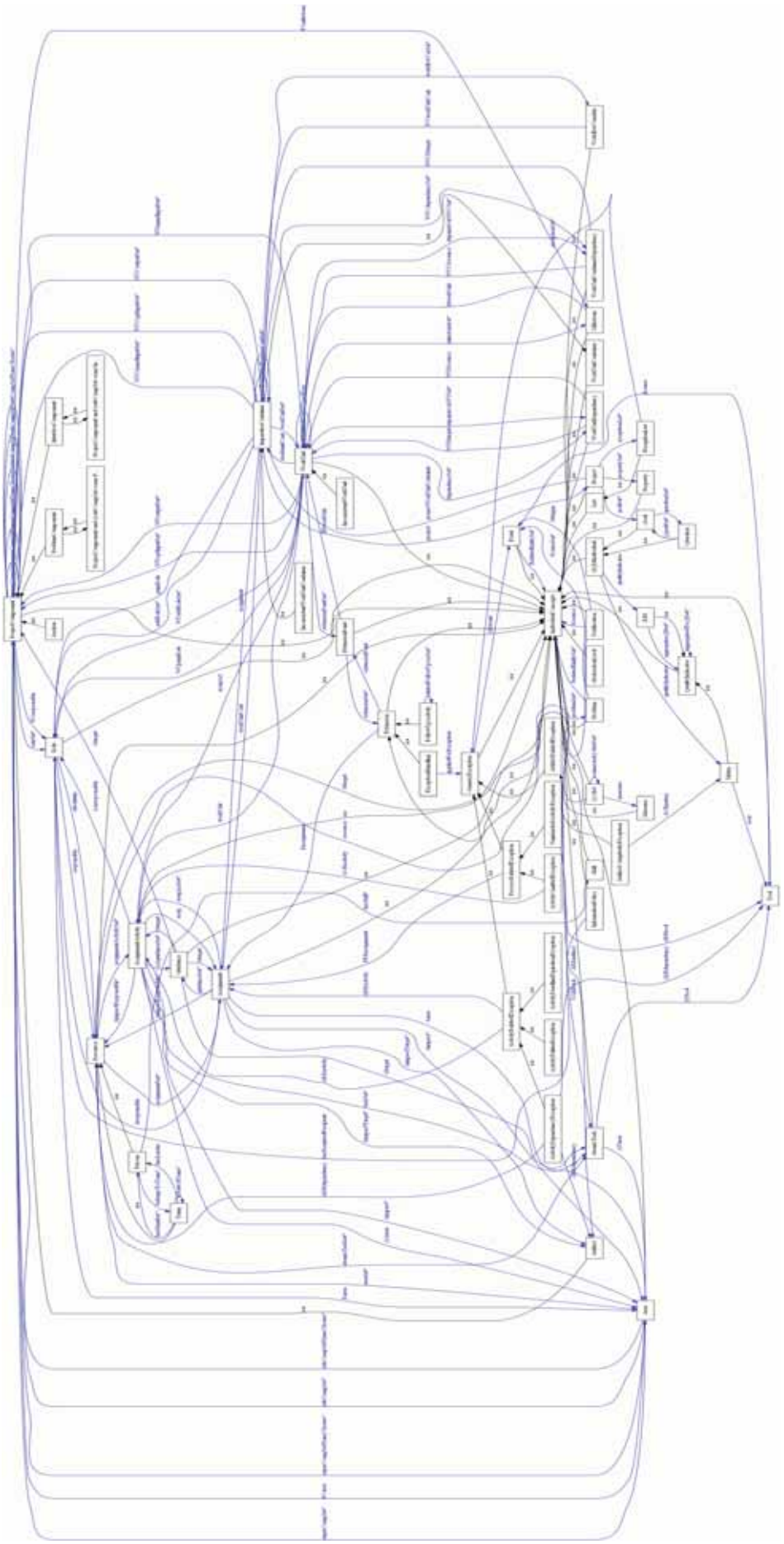

Figure A-2: Imperative individual concepts in ontology

Listing A-2 (Work unit)

```
<owl:Class rdf:about="WorkUnit">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="assignAct"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
         rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="basisWUT"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="WUfutureExec"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="singleExec"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
          rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="actInst"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="finalized"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="WUprimRole"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
```

```
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="WUpastExec"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:ID="workUnitCont"/>
        </owl:onProperty>
        <owl:cardinality
             rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:DatatypeProperty rdf:ID="WUstate"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="IndividualConcepts"/>
    </rdfs:subClassOf>
  </owl:Class>
```

A.2. Declarative Process Concepts

This section presents an excerpt of the concepts related to the declarative modeling approach from Chapter 8 (cf. Listing A-3 and Listing A-4) preceded again by a visual overview in Figure A-3. The building block template shows only one restriction because the cardinality restrictions on properties ‘info’ and ‘problem’ are realized in its super concept, the declarative modeling element. The declarative container template, in turn, is a sub concept of the declarative modeling element and the work unit container template.

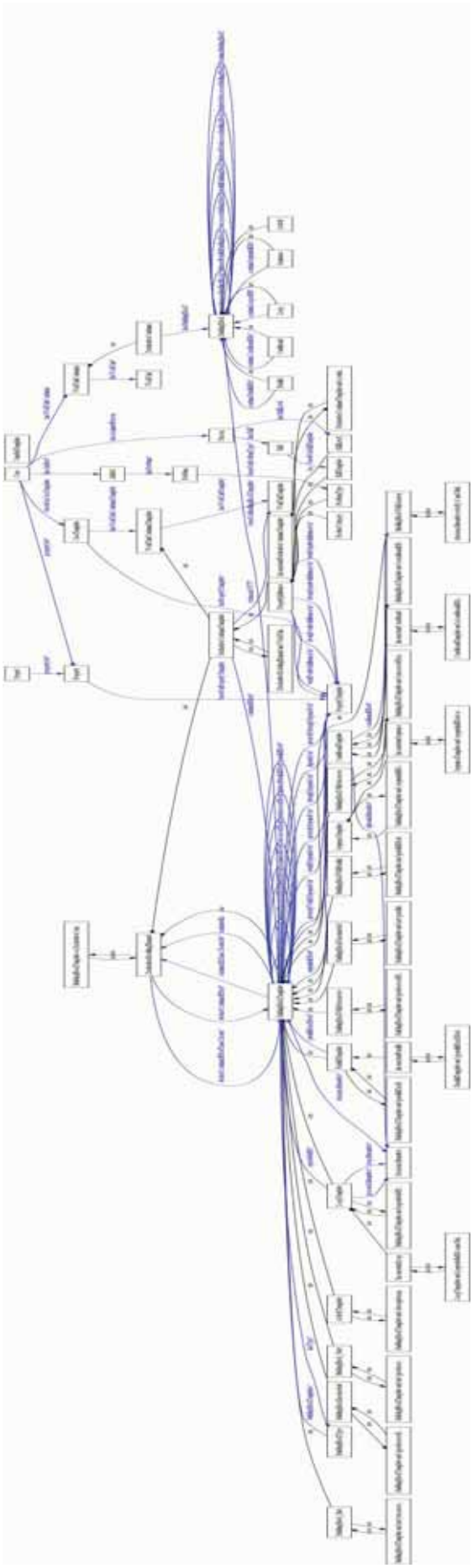

Figure A-3: Declarative concepts in ontology

Listing A-3 (Building block template)

```
<owl:Class rdf:about="BuildingBlockTemplate">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="containedIn"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="DeclarativeModelingElement"/>
</owl:Class>
```

Listing A-4 (Declarative container template)

```
<owl:Class rdf:about="#DeclarativeContainerTemplate">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Class rdf:about="#DeclarativeModelingElement"/>
            <owl:Class rdf:about="#WorkUnitContainerTemplate"/>
          </owl:intersectionOf>
        </owl:Class>
        <owl:Restriction>
          <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >1</owl:minCardinality>
          <owl:onProperty>
            <owl:ObjectProperty rdf:about="#containedBBset"/>
          </owl:onProperty>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="WorkUnitContainerTemplate"/>
  <rdfs:subClassOf rdf:resource="DeclarativeModelingElement"/>
</owl:Class>
```

B. Conceptual Framework

In this appendix, we discuss the concepts of the CPM framework and provide exemplary formal definitions. The appendix is separated into different sections to improve readability. First, different concepts for entities are shown followed by concepts for consistency checks and algorithms.

B.1. Entity Concepts

In this section, concepts for various CPM entities are discussed. For the sake of brevity we only show a selection of interesting concepts.

B.1.1. Basic Concepts

This section deals with the basic concepts of the CPM framework. We only show a selection of the definitions of these concepts. However, for completeness, Table B-1 gives an overview about all of these basic concepts. Extensions for topics like quality management are not included here.

Table B-1: Basic concepts overview

Concept	Description	Concept	Description
Identifiers	All valid identifiers over a given alphabet. All concepts have a name ε Identifiers	*Types*	All definable object types. All concepts have a distinct type ε Types
TemplateConcepts	All template concepts in the framework used for defining workflow structures.	*IndividualConcepts*	All individual concepts in the framework used for individual enactments of processes defined by the template concepts.
WFTemplates	All workflow templates within a WfMS.	*WFInstances*	All workflow instances within a WfMS.
ActivityTemplates	All activities within workflow templates in a WfMS.	*ActivityInstances*	All activities within workflow instances in a WfMS.
AreaTempls	All area templates. A set of area templates can be used to define abstract categories (or disciplines) for projects like, e.g., 'Implementation' or 'Testing'.	*Areas*	All definable areas used to categorize activities and artifacts in concrete projects as applied in many processes (e.g., the disciplines of the OpenUP process).
ProjectTempls	All project definitions within the framework, which have (among other properties) a defined type and a defined process (that is defined by work unit container template). The process depends on the type of project. Project templates also have defined area templates.	*Projects*	All concrete projects in the framework.
WorkUnitContTempls	All definable work unit container templates.	*WorkUnitConts*	All definable work unit containers.
WorkUnitTempls	All definable work unit templates.	*WorkUnits*	All definable work units.
WorkUnitContTemplDeps	All definable work unit container template dependencies.	*WorkUnitContDeps*	All definable work unit container dependencies.
WorkUnitTemplDeps	All definable work unit template dependencies.	*WorkUnitDeps*	All definable work unit dependencies.

MilestoneTempls	All definable Milestone Templates, which can be used to define abstract milestones of a process and are attached to a certain work unit template.	*Milestones*	All definable work milestones used to model the milestones of a concrete project and store information about their achievement.
AssignTempls	All definable assignment templates.	*Assigns*	All definable assignments.
AssignActTempls	All definable assignment activity templates.	*AssignActs*	All definable assignment activities.
AtomicTaskTempls	All definable atomic task templates.	*AtomicTasks*	All definable atomic tasks.
ToolTempls	All definable tool templates, which can be used to define tools types as e.g., IDE within the framework.	*Tools*	All definable tools used to capture concrete tools used in concrete projects.
ProjCompTempls	All definable project component templates, which are used to model a hierarchy of artifacts within the framework.	*ProjComps*	All definable project components capturing the structure of concrete artifact instances used in projects.
ArtifactTempls	All definable artifact templates, which are sub-concepts to the project component templates for defining the artifacts within the hierarchy.	*Artifacts*	All definable artifacts, which are sub-concepts to the project components for capturing the artifacts within the hierarchy.
SectionTempls	All definable section templates, which are sub-concepts to the project component templates for defining the structure of the hierarchy.	*Sections*	All definable sections, which are sub-concepts to the project components for structuring the hierarchy.
RoleTempls	All definable role templates, which can be used to define roles as e.g., 'Quality Manager' within the framework.	*Roles*	All definable roles used to concretely connect humans with their tasks, responsibilities, or artifacts.
EventTempls	All definable event templates, which can be used to pre-define certain events within the framework, including a relation to the tool that triggered them, as e.g., the check-in of a certain source code artifact with a source control framework.	*Events*	All definable events used to capture concrete events and their data occurring in projects.
ProblemTemplates	All definable problem templates that can be used to pre-define certain problems that might occur relating to certain events, e.g., the fact that the complexity of a source code artifact becomes too high due to code changes by different humans.	*Problems*	All definable problems capturing concrete problems and their data occurring in projects.
ExtensionPointTempls	All definable extension point templates.	*ExtensionPoints*	All definable extension points.
ExtensionTempls	All definable extension templates.	*Extensions*	All definable extensions.
WorkUserInfos	All definable workflow user information.	*WorkflowVars*	All definable workflow variables.
DecAlternatives	All definable decision alternatives.	*Resources*	All human resources within the framework comprising humans and teams of humans. Teams consist of one or more humans and have a leader which is also a human.
VarValues	All definable workflow variable values.	Persons	All humans within the framework.
VarTempls	All definable workflow variable templates.	*Teams*	All human teams within the framework.
		SkillLevels	All definable skill levels humans can possess.

Two properties shared by all concepts are type and name. The former denotes the type of concept, like work unit container, whereas the latter is a unique identifier for each concept. As both are common for all concepts, they are not further mentioned in the definitions.

Work Unit Dependency

As discussed in Chapter 7, we have added a new dependency between different workflows to the one already existent in WfMS. The template concept for this new dependency is defined in Definition B.1:

Definition B.1 (Work Unit Template Dependency)
*A **work unit template dependency** is a tuple workUnitTemplDep = (type, name, source, target, async, behavior) where*

- *source ∈ WorkUnitTempls is the source depending on target.*
- *target ∈ WorkUnitTempls is the target the source depends on*
- *async ∈ BOOLEAN indicates whether the dependency is asynchronous or synchronous.*
- *behavior ∈ {firstShot, lastShot} indicates when the dependency is satisfied.*

WorkUnitTemplDeps describes the set of all definable work unit template dependencies.

The work unit template dependency connects a work unit template (*source*) with a work unit template (*target*). When a work unit, which is based on the *source* template, comes to enactment, the work unit container containing the *source* work unit will be created (if it is not already in place). If the *async* property is set to FALSE, the termination of the *source* will depend on the termination of the *target*. As the target is a work unit in that case, it might be executed more than once in a LOOP. Therefore, the *behavior* property governs whether the source terminates with the first or *last* termination of the *target*.

Both dependencies have a related stateful individual concept capturing one individual enactment of the workflows defined by the templates. The work unit dependency (cf. Definition B.2) allows for storing the information whether its target has been executed using the property *finalized*. As the target is a work unit it might have multiple iterations. Therefore, this dependency has two properties, one indicating that the target has been executed (*executed*), the other indicating that the final execution of the target has happened (*finalized*).

Definition B.2 (Work Unit Dependency)
*A **work unit dependency** is a tuple workUnitDep = (type, name, source, target, async, behavior, executed, finalized, basis) where*

- *source ∈ WorkUnits is the source depending on target.*
- *target ∈ WorkUnits is the target of the dependency the source depends on*
- *async ∈ BOOLEAN indicates whether the dependency is asynchronous or synchronous.*
- *behavior ∈ {firstShot, lastShot} indicates when the dependency is satisfied*
- *executed ∈ BOOLEAN indicates if the target has been executed at least once*
- *finalized ∈ BOOLEAN indicates if the target has been executed for the last time*
- *basis ∈ WorkUnitTemplDeps is the template that workUnitDep is based on*

WorkUnitDeps describes the set of all definable work unit dependencies.

Human Activity Management

The human activity concepts (assignment, assignment activity, atomic task) require a particular set of runtime properties; therefore, we have added individual concepts for them. In the following we show Definition B.3 for the assignment.

Definition B.3 (Assignment)
*An **assignment** is a tuple assign = (type, name, responsible, assignActSet, workUnitCont, basis, state, guidanceSet, plannedStart, plannedEnd, actualStart, actualEnd, area, contentInfo) where*

- *responsible ∈ Resources is the resource that is responsible for assign.*
- *assignActSet is a finite set of human activities with assignAct ∈ AssignActs that are crucial to complete the assignment.*
- *workUnitCont ∈ WorkUnitConts is the work unit container assign is attributed to.*
- *basis ∈ AssignTempls is the assignment template assign is based on.*

- *state ∈ {Inactive, Active, Finished} is the state of assign.*
- *guidanceSet is a finite set of guidances(cf. Chapter 12) used to support assig.*
- *plannedStart ∈ DATETIME is the planned start time for assign.*
- *plannedEnd ∈ DATETIME is the planned end time for assign.*
- *actualStart ∈ DATETIME ∪ NULL is the actual start time for assign or undefined.*
- *actualEnd ∈ DATETIME ∪ NULL is the actual end time for assign or undefined.*
- *area ∈ Areas defines the concrete area assign is attributed to.*
- *contentInfo ∈ STRING contains information for the human on assign.*

Assigns describes the set of all definable assignments.

In order to keep track of its planned and actual execution, the assignment has four properties storing its planned and actual start and end times. In addition, it contains information on the assignment useful for the human processing it (*contentInfo*) and relations to guidances (cf. Chapter 12) for further support. Finally, it has a finite set of states whose transitions are depicted in Figure B-1.

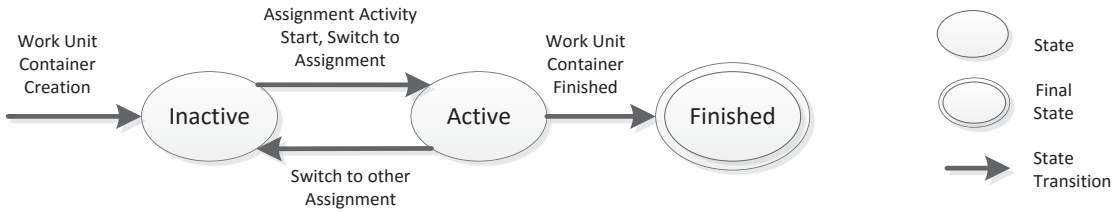

Figure B-1: Assignment states

When a work unit container is created, its related assignment is created with state ‘Inactive’ as well. It then enters state ‘Active’ when one of its assignment activities is started by the human. If he switches to another assignment, it becomes inactive again until he switches back to it. The assignment enters its final state ‘Finished’ when its work unit container is finished. The assignment activities that are part of the assignment have similar properties as well as a finite set of states. The transitions between the states are depicted in Figure B-2.

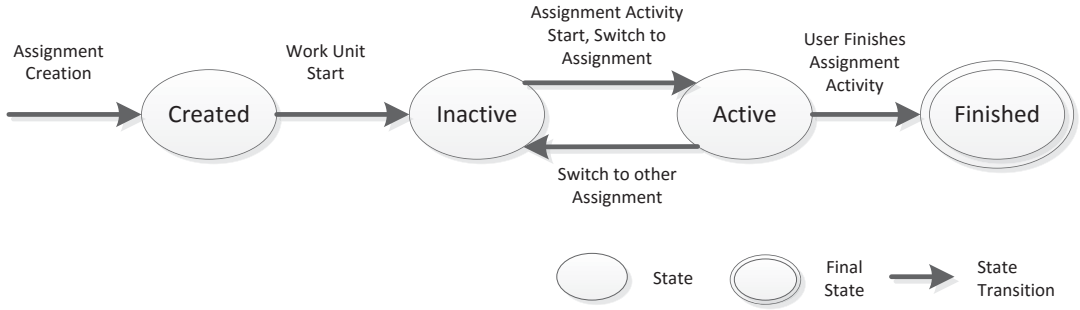

Figure B-2: Assignment activity states

When an assignment is created, the related assignment activities get created, having state ‘Created’. When the work unit related to the assignment activity starts, the activity is available for the human, therefore it enters state ‘Inactive’. When the human starts processing it, it becomes ‘Active’. If he switches to another assignment it becomes ‘Inactive’ again until he switches back. When he finally completes it, it enters final state ‘Finished’.

The assignment activity is the planned human activity with the finest granularity in the CPM framework. However, it has connections to the more fine-grained activities, the atomic tasks. As opposed to the other activity concepts, the atomic task has no properties for planned times, but an actual start and end. However, atomic tasks are fine-grained and our experiences in real projects have shown that while processing an activity, a human frequently switches between the different tasks. Therefore, not only the absolute start and end times are recorded, but the overall duration

(*taskDuration*) as well. The atomic task also has a set of states whose transitions are depicted in Figure B-3.

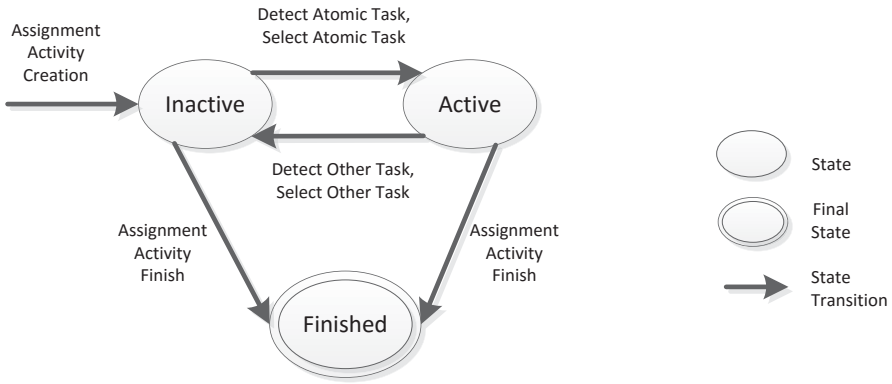

Figure B-3: Atomic task states

When an assignment activity is created, the corresponding atomic tasks (as defined by the template concepts) are also created and enter state 'Inactive'. A task enters state 'Active' when the CPM framework detects its enactment or the human explicitly selects it. It becomes inactive again when another task is selected in the same way. From both states, there is a transition to final state 'Finished' in case the human finishes the corresponding assignment activity.

Artifact Management

Artifacts being part of the SE process as well as their mutual connections cannot be modeled properly within a WfMS using the data elements being part of the workflows. To add facilities to model artifact structures, we consider the concept of the project component template (cf. Definition B.4).

Definition B.4 (Project Component Template)
*A **project component template** is a tuple projCompTempl = (type, name, reference, subCompSet, superCompSet, roleTemplSet, responsibleRoleTempl, reqCompTemplSet, stateSet, relatedCompTemplSet, areaTempl, compTemplType) where*

- *reference $\in$ STRING $\cup$ NULL is a reference to the template of a real artifact (e.g., Specification) or undefined.*
- *subCompTemplSet is a finite set of project component templates that are subordinate to projCompTmpl with projCompTempl $\in$ ProjCompTempls.*
- *superCompTemplSet is a finite set of project component templates with projCompTempl $\in$ ProjCompTempls that the projCompTempl is subordinate to.*
- *roleTemplSet is a finite set of role templates with roleTempl $\in$ RoleTempls used to define one or multiple human roles according to projCompTempl.*
- *responsibleRoleTempl $\in$ RoleTempls defines the main role template according to projCompTempl.*
- *reqCompTemplSet is a finite set of project component templates that projCompTempl requires with reqCompTempl $\in$ ProjCompTempls.*
- *stateSet is a finite set of STRINGS used to define the possible states for projCompTempl.*
- *relatedCompTemplSet is a finite set of project component templates that has a content-related relation to projCompTempl with projCompTempl $\in$ ProjCompTempls.*
- *areaTempl $\in$ AreaTempls is the area template projCompTempl is associated to.*
- *compTemplType $\in$ STRING is concretization of the type of projCompTempl.*

The project component template is an abstract concept that generalizes more concrete sub concepts. Therefore, ProjCompTempls is defined as follows:

ProjCompTempls := ArtifactTempls $\cap$ SectionTempls. ArtifactTempls and SectionTempls are disjoint subsets of ProjCompTempls that are defined by:

- *artifactTempl ∈ ArtifactTempls is a project component template for which the following applies:*
  artifactTempl.reference ≠ NULL ∧ artifactTempl.subCompTemplSet = Ø.
- *sectionTempl ∈ SectionTempls is a project component template for which the following applies:*
  sectionTempl.subCompTemplSet ≠ Ø ∧ sectionTempl.reference = NULL.

The project component template has a set of basic properties starting with a reference to the real entity it models (*reference*). In addition, it enables the definition of a set of role templates (*roleTemplSet*) and one role template responsible for the project component (*responsibleRoleTempl*). That way, a CPM framework can determine which human to inform (e.g., when there is a problem with an artifact). It further allows for content-related categorization by referring to an area template (*areaTempl*) and type (*compTemplType*). The latter might be for example 'PDF file' or 'Java artifact'. Based on this type, the CPM framework can issue activities matching the project component (cf. Chapter 10). As opposed to the other concepts, the project component template allows defining a set of states the *project components* based on it may have during execution. This option has been introduced since many different types of artifacts in projects with a myriad of different states exist. Note that these states are not controlled by the CPM framework, but must be set by humans during enactment.

Another feature of the project component template is the possibility to add various relations to other project component templates. Such relations can be used to model various dependencies of artifacts as required by SE process models like the OpenUP [EcFo15]. On one hand, this enables a hierarchy of project component templates with the properties *subCompTemplSet* and *superCompTemplSet.* On the other, content-related connections can be established using the *relatedCompTemplSet.* Finally, the property *reqCompTemplSet* allows one project component template to require the presence of others.

Dynamic Processes

The concepts for defining dynamic events and reactions to them are discussed in this section. The most important concepts are, in this context, the extension point and the extension (cf. Chapter 7). For both of these, we present the template concepts in Definition B.5.

Definition B.5 (Extension Point Template)
*An **extension point template** models templates for extension points to the work unit in a project. It is represented as a tuple extensionPointTempl = (type, name, extensionType, extensionSubType, abstractionLevel, parallelInsertion) where*

- *extensionType ∈ ExtensionTempls marks the type of extension template applicable to extensionPointTempl.*
- *extensionSubType ∈ STRING marks the sub-type of extension template applicable to extensionPointTempl.*
- *abstractionLevel ∈ STRING is the level of abstraction of extension templates applicable to this point.*
- *parallelInstertion ∈ BOOLEAN marks whether the extension template shall be inserted in parallel to the work unit template the former is attached to or sequentially after it.*

ExtensionPointTempls describes the set of all definable extension point templates.

The extension point template features content- and process-related information: the type of extension can be specified using the two properties (*extensionType, extensionSubType*). Further, there is property *abstractionLevel*, which defines the abstraction level of the workflow in the entire process (e.g., operational development workflow vs. a workflow representing a phase of the process) to distinguish which extensions can be feasible. The extension point template corresponds to a marking of a change to a potentially running workflow instance. As discussed in Chapter 7 we apply a simple insertion into the workflow instance (i.e., Pattern AP1 from [WRR08]). For this pattern, three options for insertion exist: serial insert, parallel insert and conditional insert. The third option is redundant, as the added activity would be contemporarily inserted into the workflow instance matching the properties of the

situation. In such a case, no further condition is necessary. To distinguish between option one and two, the property *parallelInstertion* is applied.

To classify the extensions made to the process, we further introduce the concept of the *extension template* in (cf. Definition B.6).

Definition B.6 (Extension Template)
*An **extension template** models templates for extensions to process enactment. It is represented as a tuple extensionTempl = (type, name, assignmentTempl, extensionPointTemplSet, extensionSubType, abstractionLevel, skillLevelSet) where*

- *assignmentTempl ∈ AssignTempls defines the concrete human assignment that marks the content of the extension based on extensionTempl.*
- *extensionPointTemplSet is the set of extension point templates, to which extensionTempl is applicable with extensionPointTempl ∈ ExtensionPointTempls.*
- *extensionSubType ∈ STRING marks the sub-type of extension applicable to extensionPointTempl.*
- *abstractionLevel ∈ STRING is the level of abstraction for which extensionTempl is applicable.*
- *skillLevelSet is a finite set of skill levels, one of which a human executing the extension shall possess with skillLevel ∈ SiklLevels.*

The extension template corresponds to an abstract concept that generalizes more concrete sub concepts. Therefore, ExtensionTempls is defined as follows:

ExtensionTempls := FollwowActTempls ∩ MeasureTempls ∩ ExcHandTempls. FollwowActTempls is the set of all definable Follow-up Activity Templates (used for activity coordination and detailed in Chapter 10), MeasureTempls is the set of all definable Quality Measure Templates (used for software quality management and detailed in Chapter 9), and ExcHandTempls is the set of all definable Exception Handling Templates (used for exception handling and detailed in Chapter 11). These three are disjoint subsets of ExtensionTempls.

The extension template features, same as the extension point template, a sub-type (*extensionSubType*) and abstraction level (*abstractionLevel*). Furthermore, it features a set of extension point templates for which it is applicable (*extensionPointTemplSet*) and a relation to an assignment template (*assignmentTempl*) that captures the human activity to be used to extend the process. In addition, it can also be specified, what skill level the human executing the extension should have (*skillLevelSet*). For more information on these properties and a detailed discussion of their application for integrating software quality measures into the process, we refer to Chapter 9.

The extension of a process can become necessary in many cases. We have discussed different cases for that in Chapter 4: task coordination (requirement *R:Coord*), process exception handling (requirement *R:Exc*), and software quality management (requirement *R:Qual*). In alignment with these cases and requirements, we have introduced three concrete sub-types of the abstract extension concept. These have been discussed in detail in the Chapters 9, 10, and 11.

B.2. Consistency Checks

This section discusses the consistency checks and conditions we created for the CPM concepts. It is split up regarding the different areas the CPM framework covers. These checks are extensible and do not claim to be complete. They are a starting point influenced partly by sources from literature and experiences from practical settings.

B.2.1. Basic Concepts

This section discusses consistency checks for the basic concepts applied for extending workflows.

Template and Individual Concepts

This check deals with the relation of template and individual concepts. Both concept sets share similar properties and the former set is used to pre-define the relations between concepts of the latter one. Therefore, individual concepts must not ignore these definitions. Figure B-4 illustrates a concrete case prohibited with this check. In this case, atomic task template 'Coding' is connected to the tool template 'IDE'. However, a concrete individual has a connection to the static code analysis tool PMD instead.

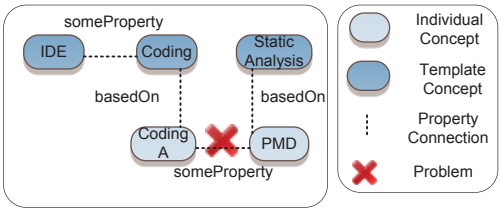

Figure B-4: Consistency check: template properties

Work Unit Containers

For the work unit containers we apply consistency checks for various problems. Figure B-5 illustrates cases where properties of work unit containers have been set erroneously. Case a) deals with a work unit container without any work unit. In turn, case b) shows a work unit container requiring another one not contained in the same project. Such a container is out of control of the current project and hence does not contribute any results to it. Cases c) and d) concern work units that read or write project components not read or written by its container. The CPM framework's definition implies that such components are exchanged with the container and distributed to its work units. Therefore, cases c) and d) should be prevented. In case e), a work unit container is defined to have no workflow instance but still has a connection to one. This collides with the definition of the 'noWorkflow' property and the workflow instance is redundant.

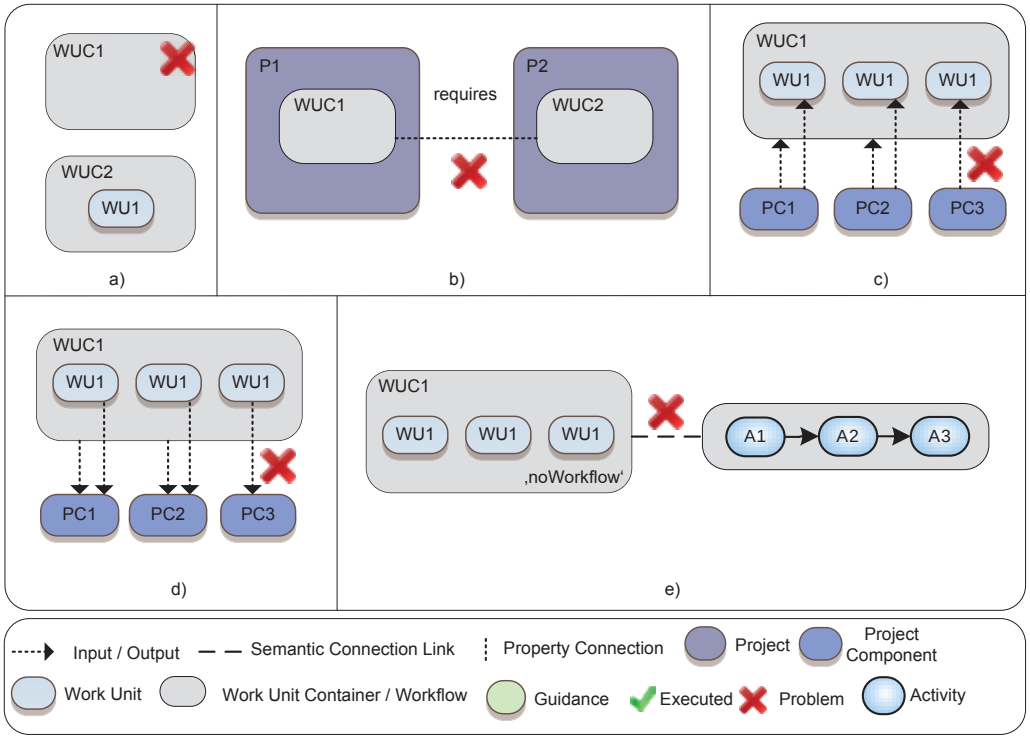

Figure B-5: Consistency check: work unit containers

Work Units

The definition of work units might contain certain erroneously set properties. Figure B-6 illustrates an undesired case for it. It concerns the usage of the work unit: It should be connected to a human-centric activity (assignment activity) or to a sub work unit container or work unit. If none of them is applied, the work unit will terminate right after its activation and would thus be useless.

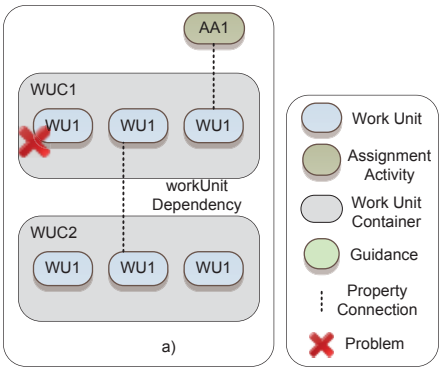

Figure B-6: Consistency check: work units

Dependencies

The dependencies between work units and containers may imply erroneously specified properties interfering with correct execution. Figure B-7 illustrates three undesired cases. Cases a) and b) show different examples of circular dependencies with work unit dependencies (a) and work unit container dependencies. Such cases might produce deadlocks and should thus be prevented. A special case for the work unit dependency is shown in case c): If such a dependency is set to a work unit that is omittable, the dependency will not be satisfied if the work unit is be omitted. Therefore, we also prevent the setting of such dependency.

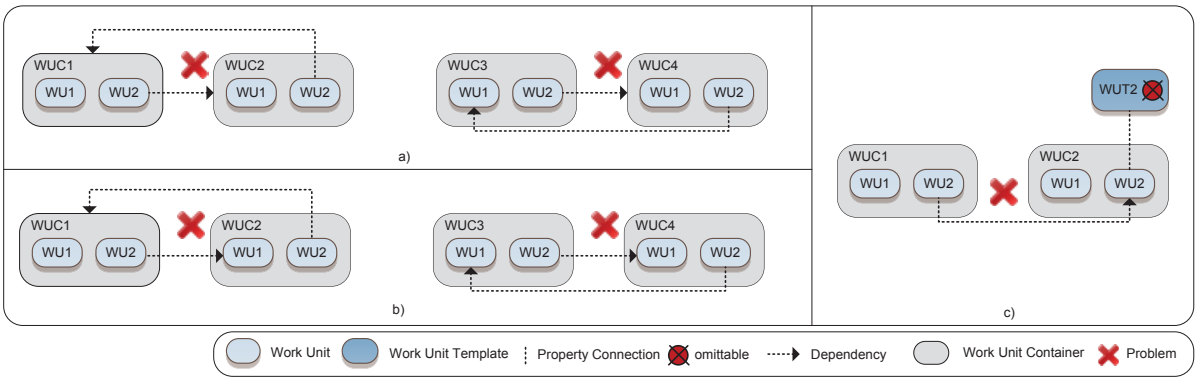

Figure B-7: Consistency check: dependencies

Variables

The variables used for governing the execution trace of the workflow instances are modeled in the context management component. This includes our concept for abstraction of internal workflow logic (cf. Chapter 7). The involved concepts might also imply erroneously set properties interfering with correct execution. Figure B-8 illustrates various cases for that. The connection to the variables in the WfMS can only be established if all variables are correctly mapped. Therefore, incorrect naming (case a) or incomplete mapping (case b) should be prevented. As the CPM framework does not monitor the correctness of all read and write operations on the variables and we have want to enable a standard trace for each executed workflow instance, each work unit container template must supply initial values for all variable templates (violated in case c). Similarly, each modeled human decision must have at least one decision alternative, otherwise the human might not make the decision (violated in

case d). For each of these decisions, a standard alternative may be defined to unburden the human from the decision. To prevent ambiguities, for each decision, there must be exactly one standard alternative (violated in case e). The decision alternatives are modeled as abstraction of the workflow variables. Therefore, each alternative must set at least one of the variables (violated in case f). Otherwise, the alternative will have no effect at all.

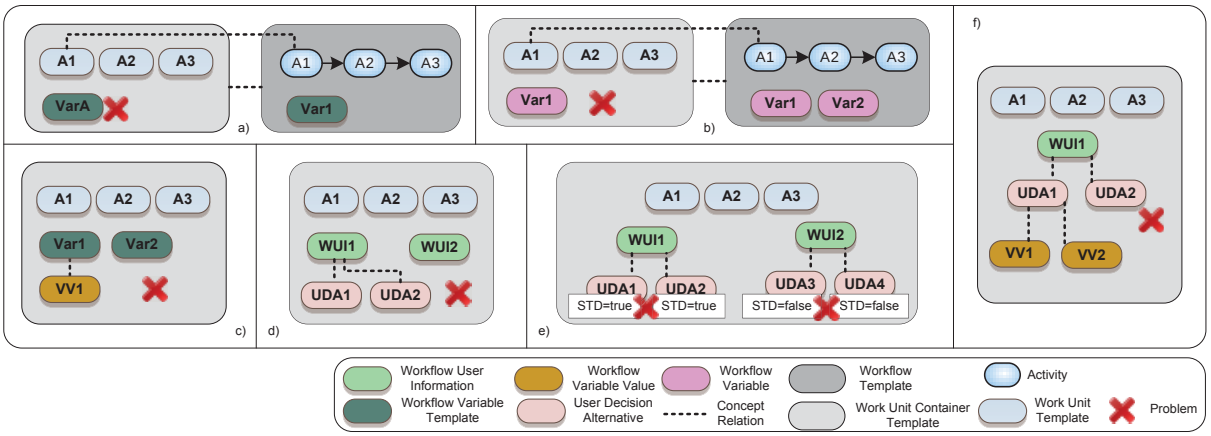

Figure B-8: Consistency check: variables

B.2.2. Extrinsic Workflows

This section discusses modeling conditions and checks for the concepts realizing extrinsic workflows for SE issue processing.

Modeling Conditions

This section presents the modeling conditions enforcing properties on the building blocks that enable the creation of block-structured workflows from them.

Condition C1: Each workflow shall not have multiple start or end points. This promotes simple and understandable models as suggested in [MRv10]. Such a start or end point can by a single building block template or multiple building block templates that are connected in parallel.
Condition C2: Each activity shall have at least one connection to other activities. This condition ensures that workflows are buildable, as a workflow cannot be built from unconnected activities since it cannot be determined when to execute this activity. The exception from this condition are containers with only one contained activity. The latter shall have no connection to other activities as they are outside the container.
Condition C3: No cyclic sequencing shall be specified, as this is error-prone: It might be impossible to determine start and end point of a cyclic workflow. Furthermore, if a cycle were integrated in a workflow, there will be no clear exit condition for that cycle making execution nondeterministic. If activities are to be executed more than once, this shall be specified using the *loop template*.
Condition C4: The activity structure shall be simple. An activity shall have only one successor and one predecessor. If multiple successors are needed, one can be defined as successor and the other shall be specified as parallel to that successor. This limitation is introduced to support simplicity and understandability of the models. Furthermore, the specification of multiple successors of an activity without specifying how they should be executed (in parallel? conditional?) results in nondeterministic models. However, complex workflow modeling is enabled in a defined way using the specialized building block templates.
Condition C5: A building block template shall not be sequentially connected to another building block template to which it is also connected in parallel. Such a connection is inconsistent, specifying that it should be executed after (before) and parallel to the other building block template at the same time.

Condition C6: The different specialized building block template concepts (*sequence template*, *parallel template*, *loop template*, and *conditional template*) enable hierarchical specification of declarative workflows. The constraints utilized to structure the building block templates (hasSuccessor and hasParallel) shall be defined in a way that does not violate this hierarchical specification as this would make the structure more complicated and may even introduce inconsistencies. This implies that a building block template is not contained in two different other building block templates and that it has no connections to other building block templates that are not contained in the same building block template. Figure B-9 shows inconsistently specified examples. The inconsistent *loop template* shows a constraint (between activity 3 and 4) that violates hierarchical specification. Generation of a block-structured workflow is not possible, as the system would generate LOOP nodes around the activities 2 and 3, and activity 3 would have a connection with activity 4 that violates the block structure.

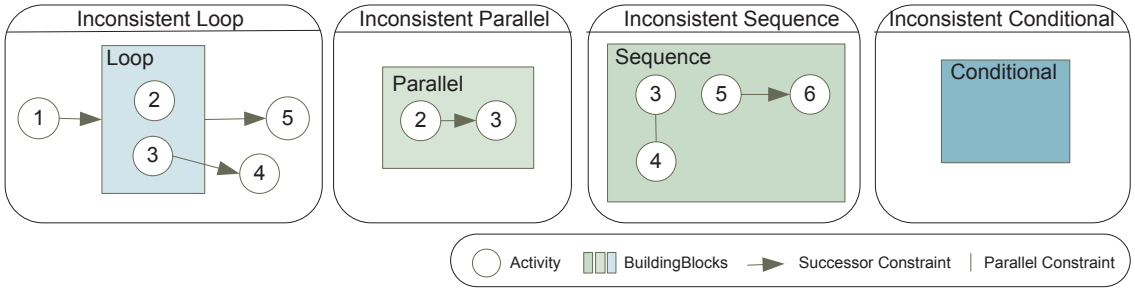

Figure B-9: Inconsistent concept examples

Condition C7: A *loop template* shall only contain one building block template. This can be a simple activity or any other building block template, enabling the looping of any structures. This constraint prohibits inconsistent specification as shown with the inconsistent *loop template* in Figure B-9. That specification lacks a connection between activity 2 and 3. Simple modeling is again supported by defining that the *loop template* is for repetitive execution of a contained activity or a structure that is represented by another building block template.

Condition C8: A *parallel template* shall contain at least two building blocks. This condition is introduced to support simple and readable process models. A *parallel template* with only one contained building block template does not endanger workflow correctness. However, it would add unnecessary AND-splits and joins to the workflow.

Condition C9: A *parallel template* shall contain only building blocks that are connected in parallel. This constraint again supports simple hierarchical modeling, prohibiting confusing and error-prone structures as shown by the inconsistent *parallel template* shown in Figure B-9.

Condition C10: A *sequence template* shall contain at least two building blocks. This condition avoids specification of unnecessary building block templates, since a *sequence template* containing only one activity is similar to only specifying that contained activity without the *sequence template.*

Condition C11: A *sequence template* shall contain only sequentially connected building blocks. As with Condition C9, this condition supports a clear definition of the building block templates. A structure as shown by the inconsistent *sequence template* in Figure B-9 is thus prohibited as it also contains the parallel activities 3 and 4. On the other hand, it also has no specified connection between the parallel activities 3 and 4 and the sequential activities 5 and 6.

Condition C12: A *sequence template* shall contain a clear start and end point. This condition avoids cyclic dependencies of the activities in the *sequence template*.

Condition C13: A *conditional template* shall only contain unconnected activities or building block templates. This condition is applied because there will be only one or none of the contained building block templates selected for execution, and connections between them would thus produce inconsistencies.

Condition C14: A *conditional template* shall contain a minimal number of activities / building block templates: If the *conditional template* is defined as optional, it must contain at least one activity, else it would only add complexity to a workflow generating XOR-splits and joins with no contained activities as shown in Figure B-9. If the *conditional template* is not defined as optional, it must contain

at least two activities since it would otherwise produce an inconsistent XOR pattern in the workflow containing only one branch.

Besides the sequencing constraints that are always only checked locally for the container or current building block template, there are also the existence constraints. These are in place for checking the soundness of a subset of activities that has been chosen due to contextual properties and are checked recursively for one container. However, to prevent modeling of containers that are inconsistent or foster inconsistent activity subsets, two conditions regarding the existence constraints are added to the build time checks:

Condition C15: One activity shall not both require and mutually exclude the same activity.
Condition C16: If an activity requires another activity, the latter must also be part of that container. If this is not the case, every activity subset containing the first activity will necessarily be inconsistent. An additional constraint for the mutual exclusion constraint is not needed, as it is possible to integrate two mutually exclusive activities in the candidate set of one container. All activity subsets not containing both of them will then be consistent.

In the following, we describe a mapping of these conditions to concrete checks applied on the different concepts. These checks have been implemented as exemplarily shown in Chapter 13 for the *sequence template*.

The conditions for the *sequence template* realize the following subset of the aforementioned modeling conditions (cf. Figure B-10): hierarchically separated modeling (cf. C6) is checked (cf. case c). The other checks deal with the conditions that directly apply to the *sequence template*: The correct number of contained building block templates (cf. C10) is enforced (cf. case b) and the correct connections between these (cf. C11) is governed (cf. case a). Finally, the presence of a single start and end point within the *sequence template* (cf. case d and e) is enforced (cf. C12).

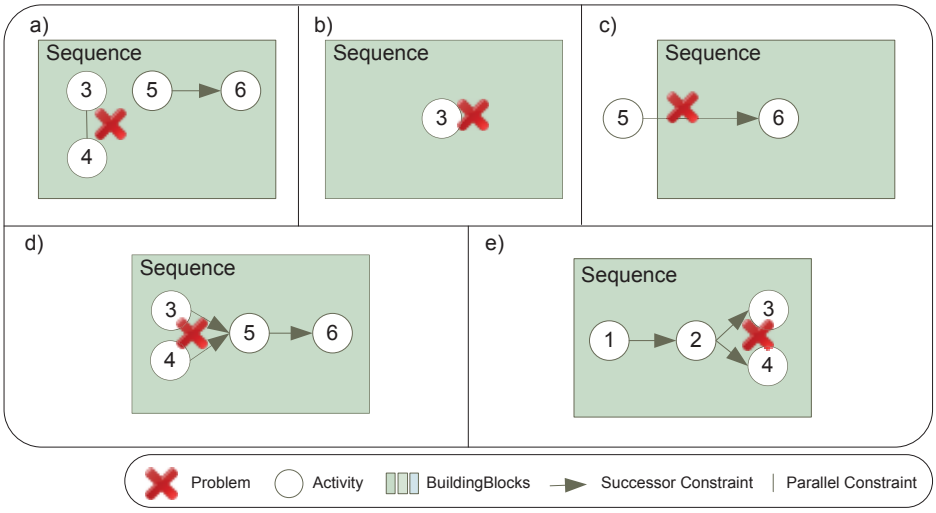

Figure B-10: Consistency check: sequence template

Similar checks are applied for the *parallel template* (cf. Figure B-11). Again, hierarchically separated modeling (cf. C6) is enforced (cf. case c). In addition, the correct connections between contained building block templates (cf. C9), and their correct number (cf. C8) is also checked (cf. case a and b).

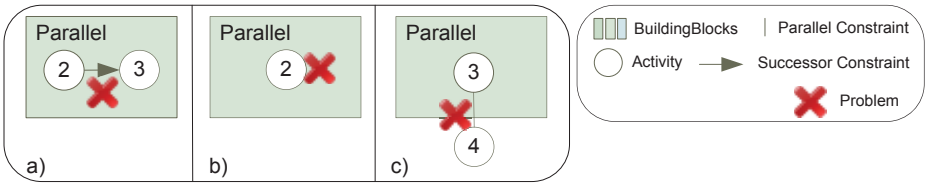

Figure B-11: Consistency check: parallel template

The checks applied to the *loop template* also implement C6, enforcing hierarchically separated modeling (cf. Figure B-12 case b). In addition the correct number of contained building block templates (cf. C7) is also checked (cf. case a).

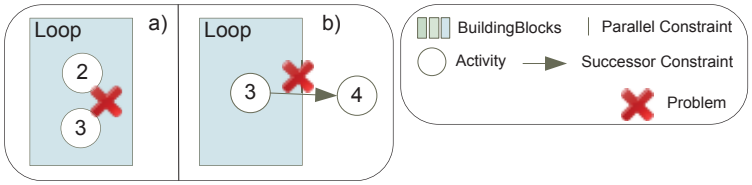

Figure B-12: Consistency check: loop template

Regarding the *conditional template*, no separate check is required for implementing hierarchically separated modeling. A *conditional template* shall only contain unconnected building block templates (cf. C13 and Figure B-13 case b). The correct number of contained building block templates (cf. C14) is also checked (cf. case a).

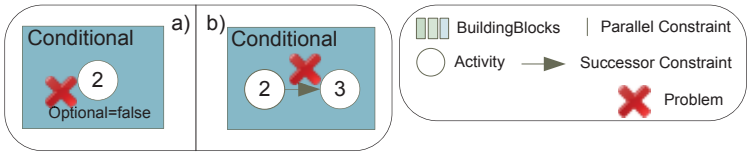

Figure B-13: Consistency check: conditional template

Concerning the declarative container template, the presence of a single start or end point (cf. C1) is checked (cf. Figure B-14 case a and b). As defined in C1, both start and end point may contain multiple building block templates if they are connected in parallel. The presence of an unconnected building block template within a declarative work unit container is prohibited as well. This will only be permitted if the container contains exactly one building block template. In that case, the building block template will have to be unconnected (cf. C2 and Figure B-14 case c). Another check prohibits cyclic dependencies between contained building block templates (cf. C3 and case d). Furthermore, a consistent container must only contain consistent building block templates (cf. case e). Two other checks deal with the existence constraint. It is ensured that no building block template in a container requires and excludes the same activity (cf. C15 and case f). Finally, no building block template in a container shall require another building block template that is not part of the container or one of its contained building block templates (cf. C16 and case g).

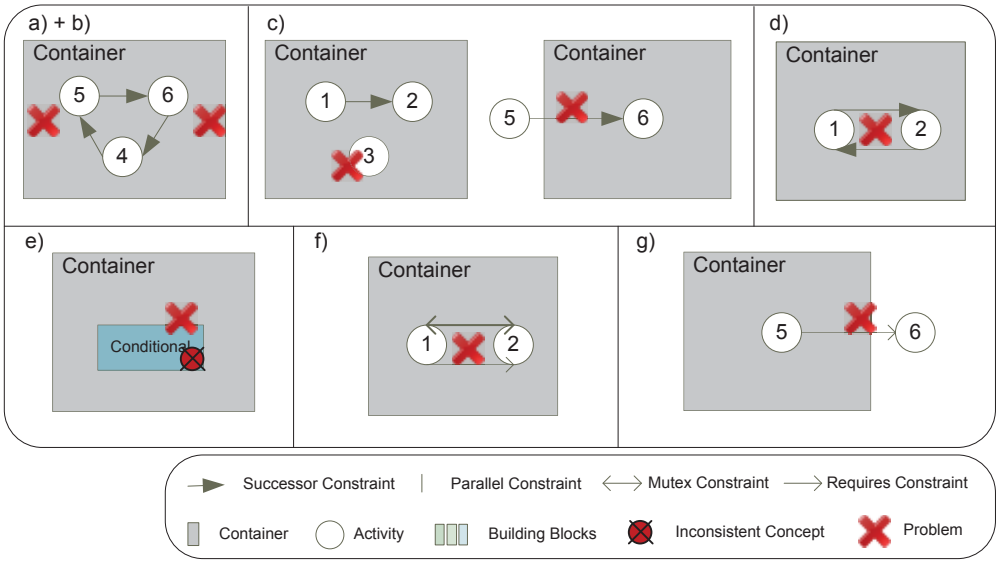

Figure B-14: Consistency check: declarative container template

B.2.3. Quality Management

This section discusses the realization of the agent structure utilized in Chapter 9 for automatic software quality measure prioritizing. The agent structure must be capable of both realizing the bidding process for the proactive measures and the voting process for the reactive measures. The bidding process shall favor agents whose goals are not in a good state. If this is the case, an agent takes place in the bidding process. If this applies for none of them, all can take place. If an agent wins one round, it may place one of his proactive measures in the list from which, at a later time point, measures for application will be selected. The voting process is different. Here, different agents vote on all measures in the reactive measure list that are attributed to their goal. That way, measures supporting multiple goals will have a higher probability to come to execution.

To be able to realize these two prioritizing processes, the agent structure is defined as depicted in Figure B-15. The *AGQM agent* is responsible for managing the multi-agent system component. It instantiates the other agents and determines whether a reactive or proactive measure will be proposed. For each defined goal, a *goal agent* is instantiated. In the proactive section, the goal agents communicate with the *session agent* to realize the bidding process. Thereby, the session agent takes the role of the "buyer" and thus selects the proactive measure from the goal agent with the highest bid. Each goal agent places bids according to its strategy. Initially, we have included three basic strategies. The strategies 'offensive', 'balanced' and 'defensive' influence the starting bid of the agents as well as win-or-lose adaptation based on the last bidding session. If insufficient points are left for the intended bid, the agent bids all points it has left. If an agent has no points left, it cannot place bids anymore until all agents have no points left, whereupon all points are reset to their initial value. Each agent has a list of proactive measures it could offer. Goals known to be at risk due to GKPI deviation are elevated to participation status in the bidding. If no report containing GKPI violations is received, all agents participate.

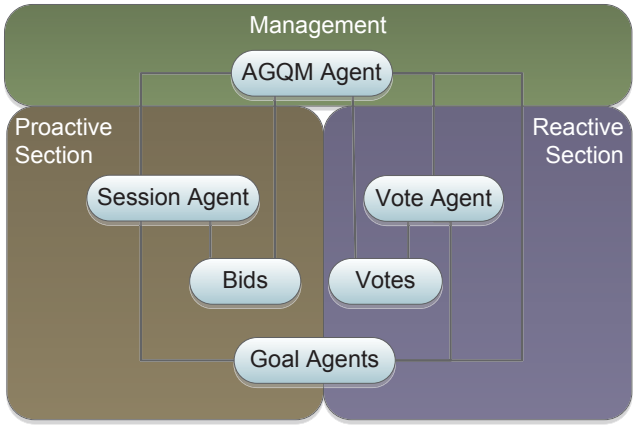

Figure B-15: Agent structure

The reactive section is realized by the *vote agent*. Each time a report is received, the vote agent creates a weighted list of reactive measures using the report. To elicit the weight of each measure, the vote agent communicates with the goal agents. For each measure, a goal agent evaluates whether that measure is associated to its goal via the aforementioned connection of measures, metrics, KPIs, and goals. In each voting process, a goal agent distributes all of its points (initially allocated at the beginning of the iteration) uniformly to all measures in the current report that are associated to its goal. If multiple agents vote on one measure, the points are aggregated. If no report has been received yet, the voting process cannot be conducted. In that case, a proactive session is substituted. That way, the multi-agent system component creates a new ordered list of measures that mirror the predefined importance of the project's quality goals.

B.3. Algorithms

This Chapter includes a set of additional algorithms not discussed in the main chapter of this work.

B.3.1. Basic Workflow Enactment

This section deals with algorithms needed for contextually extended workflow enactment as discussed in Chapter 7.

Activity Marking

This section shows algorithms for marking omittable and repeatable activities.

Omittable Activities. Activities in a workflow can be omitted due to the XOR pattern. In that case, there are points in the execution when it is clear that the execution of the respective activity will not happen in this instance of the workflow. These points correspond to the execution of other activities called *terminator activities* as described in Chapter 7. Algorithm B-1 is used to mark omittable activities and establish connections between an omittable activity and its terminator activity.

Algorithm B-1: markOmittable (Pseudo Code for marking omittable activities)

Require: Decomposed Workflow list P {Blocks, Activities}, List targetBranch, List activitiesToConnect

```
1: for all elements in targetBranch do
2:     if not activitiesToConnect.empty()
3:         connectNodes(element, activitiesToConnect, 'omittable')
4:     end if
5:     if element ∈ blocks
```

```
 6:          List childConnectActivities ← activitiesToConnect
 7:          if element ∈ xors
 8:              for all element.branches do
 9:                  if not branch.isEmpty()
10:                      List newTerminatorActivities ← Ø
11:                      getTerminatorActivities(P, branch, newTerminatorActivities, false, true)
12:                      markOmittable(P, branch, childConnectActivities ∪ newTerminatorActivities)
13:                  end if
14:              end for
15:          else
16:              for all element.branches do
17:                  markOmittable(P, branch, childConnectActivities)
18:              end for
19:          end if
20:      end if
21: end for
```

The algorithm is explained in the following and graphically illustrated in Example B-1. The algorithm takes the decomposed workflow list discussed in Chapter 7 as input as well as a decomposed workflow list representing the point in the workflow where this execution of the algorithm should operate on. For the initial execution on a workflow, this will be the whole workflow. It also expects a list of activities, whose execution triggers the deactivation of a particular activity that is empty at the beginning (called terminator activities). The algorithm iterates through the workflow list and when there are terminator activities (*activitiesToConnect*) it marks the current activities as omittable and connects it bidirectionally with its terminator activities (Line 3) (cf. connectNodes() and the activities 1-5 in the example). This is needed for each activity when a workflow is executed later on. However, the algorithm also adds the markings to the blocks. These markings will be used to facilitate the making of activities that are inserted into the workflow when it is already running (cf. Algorithm B-2). If the algorithm encounters a *block,* a new list is created (Line 5 and 6). This new list is used for new terminator activities of the encountered *block* and other *blocks* within it. This is done since the lists are passed as call-by-reference so that each level of the recursion has its own list that can also be used for further levels of the recursion but does not change the lists of upper levels of the recursion. That way, in Line 6 only the values in the list are copied (cf. e.g. in recursion Rec1 in the example). This is done because activities can be deactivated by multiple other activities. Consider e.g., multiple nested XOR patterns: An activity within an inner XOR pattern can be deactivated by activities of other branches of each of that XOR patterns. If a XOR *block* is encountered, the next step is the determination of the terminator activities for the current branch of that XOR pattern (Line 10 and 11) (cf. the initial call and the Rec2 recursion in the example). This is done by the algorithm getTerminatorActivities already described in Chapter 7. The algorithm is then called recursively for the current branch with the current childConnectActivities list and the new terminator activities for the current branch (Line 12). The same happens if another pattern as XOR is encountered (Line 16-20) (in that case without new terminator activities).

Example B-1 (markOmittable Steps):
For this example, the workflow used for Example 7-15 in Chapter 7 has been slightly adapted to contain two nested XORs to better demonstrate the XOR handling. Therefore, Figure B-16 and Figure B-17 show the adapted workflow and the concrete steps executed, both indicating the different recursion levels.

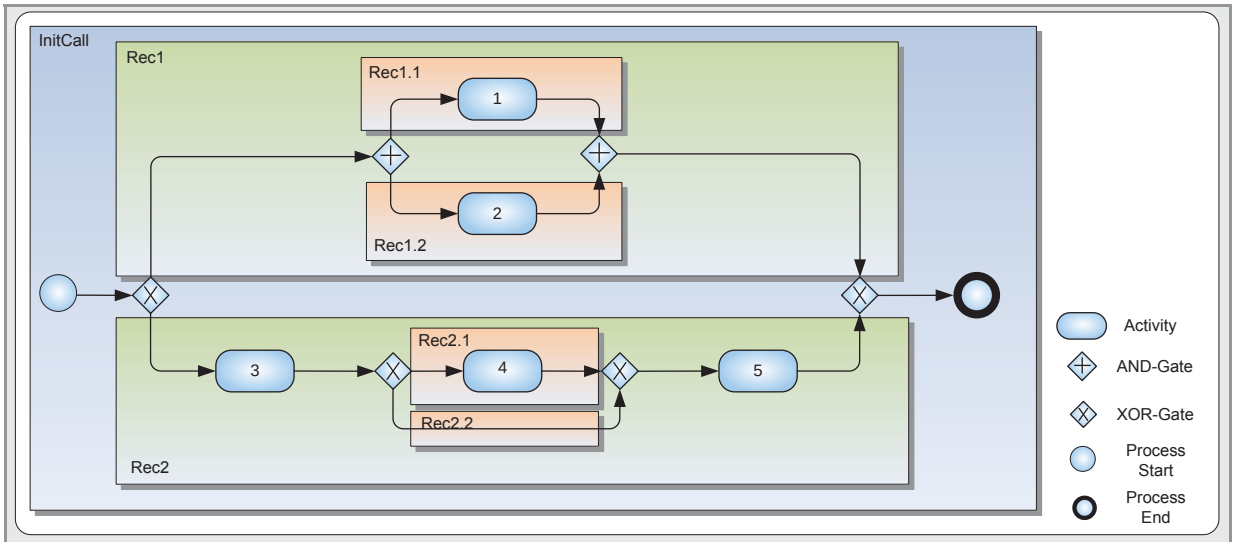

Figure B-16: markOmittable workflow

Since no activities follow the XOR1 pattern, the termination of the whole workflow is taken as terminator activity for all comprised activities. For activity 4, also the succeeding activity 5 is added.

Call	Actions	childConn	termActs
InitCall (Workflow L, targetBranch L, List activitiesToConnect AT)	element = XOR1 childConn = activitiesToConnect termActs = new List() termActs = getTA(L,L1,termActs,false,true)	--- [] [] []	--- --- [] [L]
Rec1(L, L1, childConn+termActs)	element = AND1 childConn1 = activitiesToConnect	--- [L]	--- ---
Rec1.1(L, L1.1, childConn1)	element = 1 connectNodes(1, L)	--- ---	--- ---
Rec1.2(L, L1.2, childConn1)	element = 2 connectNodes(2, L)	--- ---	--- ---
	termActs = new List() termActs = getTA(L,L2,termActs,false,true)	[] []	[] [L]
Rec2(L, L2, childConn+termActs)	element = 3 connectNodes(3, L) element = XOR2 childConn2 = activitiesToConnect termActs = new List() termActs = getTA(L,L2.1,termActs,false,true)	--- --- --- [L] [L] [L]	--- --- --- --- [] [5]
Rec2.1(L, L2.1, childConn2)	element = 4 connectNodes(4, [L, 5])	--- ---	--- ---
	termActs = new List() termActs = getTA(L,L2.2,termActs,false,true)	[L] [L]	[] [5]
Rec2.2(L, L2.2, childConn2)		---	---
	element = 5 connectNodes(5, L)	--- ---	--- ---

Figure B-17: markOmittable steps

Repeatable Activities. Due to the LOOP pattern, activities specified in a process model can be repeatable and can occur more than once in its execution. Therefore, they have to be marked so that the context management component is aware of this fact an can create new instances of the relating concepts when an activity is repeated. When activities are repeatable it may also be of interest to know when another execution of these can no more happen for a given workflow instance. This is a somehow similar case to the omittable activities and the XOR pattern: At certain points in the execution, it is clear that the respective looped activity will not be executed another time. This point is

the execution of the first activity after the LOOP. In the case of multiple nested LOOP patterns this applies to the outer LOOP pattern. Due to the similarity to the *markOmittable* Algorithm we refrain from separately discussing the *markRepeatable* algorithm.

Adaptation Markings. As discussed in Chapter 7, newly inserted activities are analyzed and marked by a separate algorithm instead of re-running all initial marking algorithms. This involves different cases. First, there are different markings: the 'repeatable' marking, the 'omittable' marking, the list of activities an activity terminates, and the list of activities that are terminated by the activity. The first three markings apply for all activities of one branch while the last one, indicating an activity as a terminator activity, only applies for the first activity in a branch or the first activity in a branch after a XOR or LOOP pattern. Different situations require that the algorithm adopts the marking in different ways. These situations are explained in the following and illustrated in Figure B-18, starting with the generic case and showing the more specific cases afterwards (where this is a refinement, i.e., the generic cases also apply to the more specific cases):

1. *Inserted into a list*, *not as first element* (i.e., the list represents the workflow instance or a branch of a pattern): In this case, only the markings (repeatable, omittable, and the connection to the terminator activities) have to be adopted from any other activity in the branch. It is assumed that in this case, the list in which the activity has been inserted cannot have been empty before because it can only be the entire workflow instance, a branch of an AND pattern, or a LOOP pattern. None of these would make sense without any contained activities.
2. *Inserted as first element into a list* (representing the workflow instance or a branch of a pattern): Being the first element in the list, the insert activity can be the terminator activity for other activities. As it is assumed (as in case 1) that the list was not empty before, the connections to activities that are terminated by the current activity can be acquired from the former first activity in the list, which is now the second activity. This activity might also be a pattern containing multiple activities. However, for this algorithm this does not matter as the marking that have been previously applied to the workflow lists treat patterns (blocks) from the outside like simple activities and apply the same markings to them.
3. *Inserted into a list after a LOOP pattern in the same branch* (i.e., the list represents the workflow instance or a branch of a pattern): The activities in the LOOP pattern are repeatable and need terminator activities to indicate that they will not be repeated again. Therefore, the LOOP and all containing activities have to get the inserted activity be added as a terminator activity. Taking a naive approach, one might assume simply taking the markings from the successor of the inserted activity suffices. However, it is possible that it is not a successor in the workflow instance.
4. *Inserted into a list after a XOR pattern in the same branch* (i.e., the list represents the workflow instance or a branch of a pattern): In principle, this is the same case as the previous one. However, XOR patterns have one special property: If they have an empty branch it is possible that no activity of the XOR pattern comes to execution. This, in turn, implies that the newly inserted activity must also be added to the activities of a XOR or LOOP pattern directly before the XOR pattern that is the predecessor of the inserted activity.
5. *Inserted into a list that represents the empty branch of a XOR pattern*: In this case, no activity is in place in the list to adopt the markings from. Therefore, the markings can be adopted from an activity in another branch and mutual terminator activity markings have to be established between the branches.

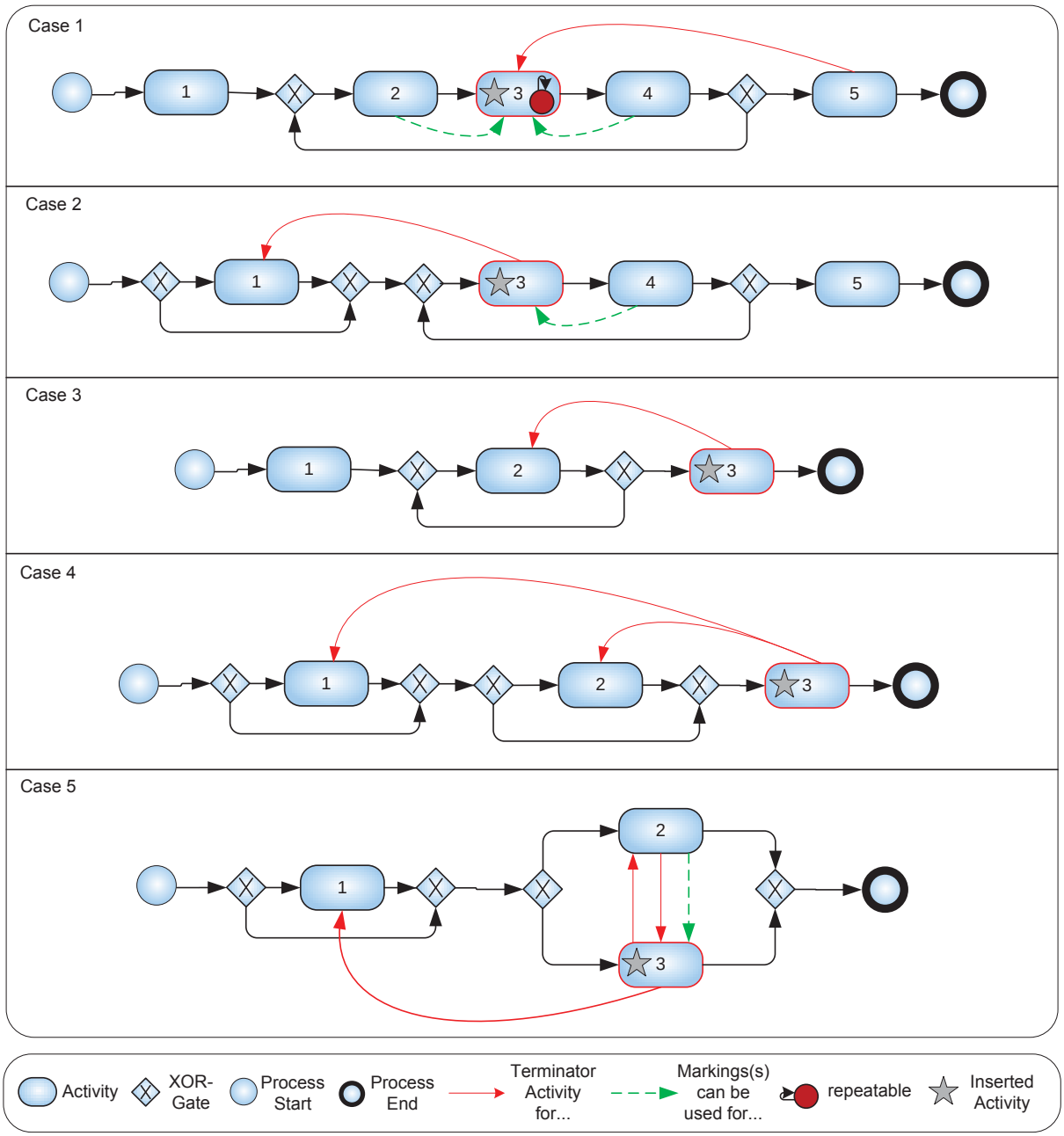

Figure B-18: Marking cases for inserted activities

In the following, Algorithm B-2 is presented that applies the markings for a newly inserted activity:

Algorithm B-2: markInsertedActivity (Pseudo Code for marking newly inserted activities)

```
Require: List targetBranch, Activity target, Block surroundingPattern
 1: if not targetBranch.size == 1
 2:     Element element ← targetBranch.getPreviousElement(target)
 3:     if element == NULL
 4:         element ← targetBranch.getNextEelement(target)
 6:     end if
 7:     adoptMarkings(target, element)
 8: end if
 9: if targetBranch.firstElement == target
10:     if not targetBranch.size == 1
11:         connectTerminatorActivity(target,targetBranch.getNextElement(target))
12:     end if
```

```
13:     if surroundingPattern == XOR and targetBranch.size == 1
14:         Boolean outerMarkings ← false
15:         for all surroundingPattern.branches do
16:             if not branch == targetBranch
17:                 if not outerMarkings
18:                     adoptMarkings(target, branch.getFirstActivity)
19:                     outerMarkings ← true
20:                 end if
21:                 target.omittableTerminators.add(branch.getFirstActivity)
22:                 branch.getFirstActivity.terminatesActivity.add(target)
23:                 for all branch.activities do
24:                     activity.omittableTerminators.add(target)
25:                     target.terminatesActivity.add(activity)
26:                 end for
27:             end if
28:         end for
29:     end if
30: else
31:     Element prevEl = targetBranch.getPreviousElement(target)
32:     while not prevEl == null
33:         if prevEl == (LOOP or XOR)
34:             addAsTerminatorActivity(prevEl , target)
35:             if prevEl == XOR and containsEmptyBranch(prevEl)
36:                 prevEl ← targetBranch.getPreviousElement(prevEl)
37:             else
38:                 prevEl ← NULL
39:             end if
40:         else
41:             prevEl ← NULL
42:         end if
43:     end while
44: end if
```

Algorithm B-2 takes as input a newly inserted activity, its branch, and the pattern surrounding that branch. First, the algorithm deals with case 1, which is the simplest case: In Line 1-8 it inherits the markings (repeatable, omittable, and potential terminator activities) from another activity in the same branch. This is only done if the new activity is not the only one in the branch, which might be the case if the surrounding pattern is a XOR pattern. Case 2 is dealt with in Lines 9 - 12: if the new activity is the first in the target branch, the list of activities it terminates is taken from the former first activity in the branch that is now the second one. The next case processed involves insertion within the empty branch of a XOR pattern (case 5). In this case, all markings are adopted from the first activity of another branch (Lines 13-20). This is done to establish connections to the other activities that are outside of the XOR pattern because the first activity of each branch of a XOR pattern has equivalent relations to activities that are outside of the XOR pattern. The mutual marking of the activities of the different branches in the XOR pattern are then applied in Lines 21-26: First, the first activity of each branch is added to the terminator activities of the newly inserted one. The latter is then added to the terminator activities of all activities in the other branches in the XOR. The final part of the algorithm (Lines 30-44) deals with cases 3 and 4. It takes the predecessor of the inserted activity and, if it is a LOOP or XOR, it adds the new activity to its terminator activities. In case of an XOR, this action is repeated. The function used to add the terminator activity (in this case *addAsTerminatorActivity()*) also applies the marking recursively to all activities contained in the LOOP / XOR.

Computational Complexity of the Algorithms

To conclude this section regarding algorithms, we will elaborate briefly on their computational complexity. For most of them, however, this is not a critical issue as they are applied during build time. Furthermore, their complexity depends on the elements in the modeled workflows and the

number of these elements is recommended to be kept rather small for various reasons. For example, [MRv10] recommends to keep the number of nodes in a workflow below 50. Our practical experiences show that it is very uncommon that a huge number of workflows or workflows with a huge number of elements are created in a modeling session. Also, only one of the algorithms (markInsertedActivity) is to be executed during runtime and this might impact operational performance. Therefore, we have put emphasis on a low complexity for this algorithm. In Table B-2 the complexity of the different algorithms is shown.

Table B-2: Complexity of the Algorithms

Algorithm	Complexity
decomposeWorkflow	O(#nodes in workflow)
markOmittable	O(#elements in list x #XORs.branches)
markRepeatable	O(#elements in list x #LOOPS)
getTerminatorActivities	O(#elements in list x #element.branches)
markInsertedActivity	O(#surroundingPattern.branches x #branch.activities +#preceding LOOPS or XORS)

The algorithm 'decomposeWorkflow' directly depends on the number of nodes in the analyzed workflow. The other three build time algorithms depend on the number of elements in the output list of the first workflow, as well as on the number of branches of the workflow patterns. However, as the build time algorithms are executed in a row or respectively call each other, an overall computational complexity for analyzing one modeled workflow can be expressed as follows:

O(#nodes + (#elements in list x #XORs.branches) x (#elements in list x #element.branches) + (#elements in list x #XORs.branches) x (#elements in list x #element.branches))
= O(#nodes + (#elements in list)² x #XORs.branches x #element.branches + (#elements in list)² x #LOOPS x #element.branches)
= **O(#nodes + (#XORs.branches + #LOOP) x (#elements in list)² x #element.branches)**

Having the properties of the workflows just discussed in mind, this complexity seems quite adequate and should not hamper modeling. Nevertheless, we have managed to realize 'markInsertedActivity' with a much smaller complexity, as it is to be executed during runtime for every activity inserted into a potentially running workflow instance. For brevity, we omit a separate discussion of the algorithms for extrinsic workflow generation as they operate on similar structures. Furthermore, extrinsic workflows are mostly smaller than intrinsic ones as they are enacted.

B.3.2. Extrinsic Workflow Generation

This section presents algorithms related to the generation of workflows from the declarative specification we have introduced in Chapter 8.

The algorithm *BBtreatment()* is utilized to convert building blocks into parts of an executable workflow. The conversion is abstracted (from the creation of context and process management concepts) using simple functions as, e.g., *insertNode()* for the insertion of one activity into a workflow.

Algorithm B-3: BBtreatment
(Pseudo Code for inserting building blocks into workflow)

Require: Building Block BB, Work Unit Container skeleton, Arc marker
Return: String errorCode
1: String *errorCode* ← empty String
2: **if** *BB* ∈ *Activities*
3: *insertNode(BB, skeleton, marker)*
4: **else if** *BB* ∈ *Sequences*
5: *errorCode* ← *sequenceTreatment(BB, skeleton, marker)*

```
 6: else if BB ∈ Parallels
 7:     errorCode ← parallelTreatment(BB, skeleton, marker)
 8: else if BB ∈ Loops
 9:     errorCode ← loopTreatment(BB, skeleton, marker)
10: else if BB ∈ Conditionals
11:     errorCode ← conditionalTreatment(BB, skeleton, marker)
12: end if
13: return errorCode
```

The algorithm expects a building block as well as the workflow skeleton to be extended including a position marker as input. If that building block is a simple activity, it is inserted into the workflow. If it is of another type, the insertion is handled by specialized algorithms. One of these, *parallelTreatment()* is exemplarily discussed in Algorithm B-4.

Algorithm B-4: parallelTreatment
(Pseudo Code for inserting a parallel into a workflow)

```
Require: Building Block BB, Work Unit Container skeleton, Arc marker
Return: String errorCode
 1: String errorCode ← empty String
 2: if BB.parallelBBset is empty
 3:     return "emptyParallel"
 4: else if BB.parallelBBset.size < 2
 5:     errorCode ← BBtreatment(parBB, skeleton, marker)
 6: else
 7:     AndSplit split ← insertParSplit(marker, skeleton)
 8:     List branches ← new List()
 9:     for all BB.parallelBBset do
10:         insertBranch(split, marker)
11:         errorCode ← BBtreatment(parBB, skeleton, marker)
12:         branches.add(marker)
13:     end for
14:     insertParJoin (marker, split, skeleton, branches)
15: end if
16: return errorCode
```

parallelTreatment inserts no pattern if no building block is contained in the parallel. If it contains only one building block, no pattern is needed either but only the building block is inserted. If multiple building blocks are contained, each is added in a separate branch of an AND pattern.

For brevity, we will refrain from discussing the computational complexity also for the extrinsic workflow generation. As stated in Chapter 8, the modeled workflows can contain many activities. However, for the workflow generation algorithms, the number of activities that are really in place for a specific situation is important. Usually, this is a rather small subset of the modeled activities.

C. Basic Actions for Process Enactment

This appendix discusses concrete actions to be performed in order to enact processes with the CPM framework.

Create Project

When a project and its process realized by a structure of workflows have been created with the template concepts, that structure can be used for concrete project executions. Therefore, a concrete project including its process must be created in the CPM framework. The action for this is shown in the following. Note that we assume that a project only has one defined process. This is an abstraction that may not hold for all projects, however, multiple processes for one project can be added with low effort within the CPM framework.

This action is applied to create a new project with its associated work unit container.
Preconditions: -
Input: project template ∧ roles, project components, and tools required by the assigned work unit container template.
Actions:

- Create project concept.
- Create contained areas as defined in template
- Apply: **Create Work Unit Container**

Output: project with work unit container in state 'Created'.

The 'Create Project' action implies the creation of its associated process that is captured by one basic work unit container (and its potential sub work unit containers). The latter is created by the following action 'Create work unit container.

Create Work Unit Container

This action is applied to create a new work unit container from a work unit container template. As opposed to WfMS where workflow instances are directly started from their templates, the containers in the CPM framework are created without starting them (or the relating WfMS workflow instances). Thus, a workflow structure for the complete process of a project can be created without having to start each of the future workflow instances.
Preconditions: -
Input: work unit container template ∧ values for roles, project components, and tools
Actions:

- Create work units as defined in template and assign to work unit container.
- Create assignment as defined in template and assign to work unit container.
- Create assignment activities as defined in template and assign to assignment.
- Create atomic tasks as defined in template and assign to assignment activities.
- Assign concrete tools to atomic tasks as defined in template.
- Set process variables as defined in the template.
- Assign concrete humans for the container roles.
- Assign concrete inputs/outputs for container (including structure of project components as defined in super/subCompsSet properties).

- Assign main human with main role also to the assignment. Distribute the humans filling the roles of the container to the work units. Add responsible party of each work unit to the relating assignment activity.
- For all defined dependencies defined by the template for work units, apply the action 'Create work unit container' to create the containers (and work units) that are the targets of the dependencies and then connect them via the 'Add work unit to work unit dependency' and 'Add work unit to container dependency' actions.

Output: work unit container in state 'Created'.

After a concrete work unit container has been created, it remains in the state created and also does not automatically initiate the start of its relating workflow instance. This has the advantage that for a project, its whole process can be prepared with a workflow structure without having to start one or more of the involved work unit containers or workflow instances. When all concepts and information is in place, a work unit container can be explicitly started including the creation / start of its relating workflow instance. To start a project, its top-level container thus has to be started. The action to execute such a start is shown in the following.

Start Work Unit Container

This action is applied to start a work unit container.
Preconditions: work unit container must be in status 'Created'.
Input: work unit container
Actions:

- Instantiate a new workflow instance from a workflow template that is connected to the template of the current work unit container.
- Connect the workflow instance and the work unit container.
- Set work unit container status to 'Started'.

Output: work unit container in state 'Started'.

When a container and its associated workflow instance is running, its progress is governed by the work units that are the mappings of the activities in the workflow instance. With these, connections to sub containers or human tasks (assignment activities) are managed as discussed in Chapter 7. Thus a sound management of their states and especially their termination is crucial as the workflow instance can only continue when one or more active work units terminate. Therefore, the action for checking if a work unit may terminate is explicitly defined in the following.

Check Work Unit Termination

This action is applied to check if a work unit can terminate.
Preconditions: work unit must be in state 'Started'.
Input: work unit
Actions:

- Check if associated assignment activity is finished.
- Check if required guidance has been used, i.e. guidance in 'guidanceSet' (of related assignment activities or project components or, if it is the final work unit of the assignment, also the assignment) are satisfied.
- Check dependencies of work unit are satisfied, i.e. if they are 'finalized' (or, in case of a work unit dependency with 'oneShot' behavior 'executed').

Output: work unit in state 'Started or 'Finished'.

As discussed in this chapter, the CPM concept applies multiple instances of the concepts relating to a looped activity in the WfMS. That means, if a WfMS activity is executed repeatedly due to a loop, the relating work unit and related concepts are already finished. So a new work unit instance has to be created, supplied with the values for markings and human activities from the prior work unit instance,

and has to be linked with the latter as well as with other containers and work units the prior work unit instance had dependency connections to. This is managed explicitly by the following action.

Create new Work Unit Instance

This action is applied to create a new instance of a work unit if the relating workflow activity is executed multiple times in a loop.
Preconditions: work unit must be in state 'Finished' and relating WfMS activity comes to execution again.
Input: work unit
Actions:

- Create new work unit and relating activity concepts and adopt values from the prior work unit.
- If the work unit is defined for single execution by the *singleExec* property only create the work unit without any other concepts and start it. That way, the new instance will terminate immediately like a 'blind activity' that will have no effect on the container and be invisible to the human.
- Check if work unit has a dependency. If yes, create the same dependencies for the new work unit. If the target container (or the container containing the target work unit) has a planned successor iteration (via the *futureExec* property), take that container as the target and link it with the new work unit. If there is no future iteration, create a new container with the values in place and link it with the new work unit.
- Check for a dependency who had the prior work unit instance as a target. If there are one or more of these having the *lastShot* behavior, link the dependency to the new work unit instance.
- Start the work unit.

Output: new work unit in state 'Started'.

As discussed many times in this work, SE process enactment is rather dynamic and thus changes to the workflow structure of a project might often be necessary. For example, a new workflow instance / work unit container may have to be added due to a changed or new customer requirement that needs to be realized. Such a new container has to be integrated into the workflow structure by adding new dependencies between that new container and a container that is part of the workflow structure. Therefore, in the following, concrete actions for adding such dependencies are shown starting with the dependency of a work unit to another container.

Add Work Unit to Container Dependency

This action is applied to create a new dependency between a source work unit and a target work unit container.
Preconditions: Source work unit and target work unit container must be in state 'Created' or 'Started'.
Input: source work unit ∧ target work unit container
Actions:

- Create dependency.

Output: newly connected work unit and container as illustrated in Figure C-1.

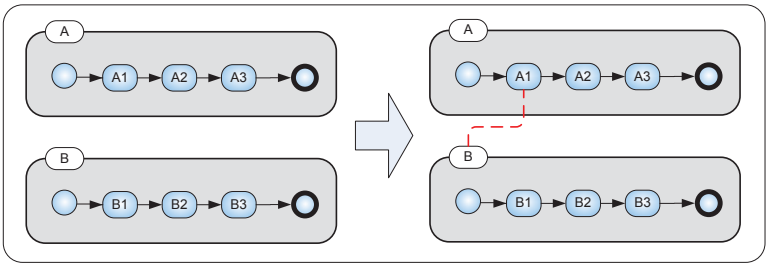

Figure C-1: Add work unit to container dependency

The addition of a dependency from a work unit to another work unit in another container is also possible as shown in the following.

Add Work Unit to Work Unit Dependency
This action is applied to create a new dependency between a source work unit and a target work unit.
Preconditions: source and target work units in state 'Created' or 'Started'.
Input: source work unit ∧ target work unit ∧ definition of behavior of new dependency
Actions:

- Create dependency.

Output: newly connected work units as illustrated in Figure C-2.

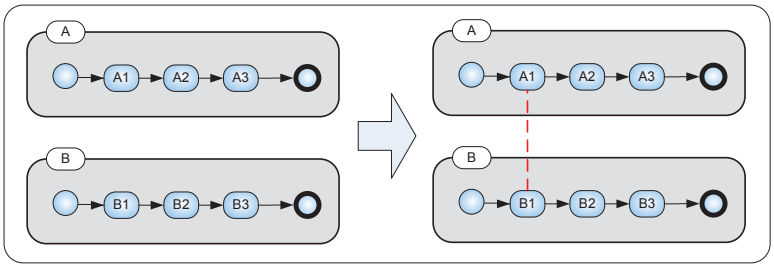

Figure C-2: Add work unit to work unit dependency

Changes to the workflow structure of a project may not only imply adding new requirements and additional workflow instances. It is also possible that, for example, a requirement can be canceled because its realization turns out to be unfeasible or too expensive. In such a case one or more containers might have to be excluded from the workflow structure. This implies removing dependencies between containers and / or work units. The actions for this are shown in the following starting with the removing of a dependency to a container.

Remove Container Dependency

This action is applied to remove mutual dependencies between work units in a source work unit container and a target work unit container.
Preconditions: source and target in state 'Created' or 'Started'.
Input: source work unit container ∧ target work unit container
Actions:

- For all dependencies of work units in the source container to work units in the target container where the source and target work units are in state 'created' or 'running', apply: **Remove work unit dependency**.

Output: -.

Same as for a container, a work unit can also have a dependency to another work unit that also might need to be removed. The relating action is shown in the following.

Remove Work Unit Dependency

This action is applied to remove a dependency between a source work unit and a target work unit or a container.
Preconditions: source and target in state 'Created' or 'Started'.
Input: source work unit ∧ (target work unit container ∨ target work unit)
Actions:

- Delete dependency
- For the source work unit apply: **Check Work Unit Termination**.

Output: -.

Another frequent change we have perceived in the projects of our industry partners is the moving of an activity / requirement / workflow instance from one point in the process to another. This happens in iterative development, when an activity is to be executed within one iteration but cannot be finished therein. Iteration deadlines are mostly firm and thus the activity (and its workflow instance) is transferred to another iteration. To facilitate this, the following action shows the moving of dependencies.

Move Work Unit Dependency

This action is applied to move a dependency between an old source work unit and a target work unit or a container to a new source work unit.
Preconditions: old and new source and target in state 'Created' or 'Started'.
Input: old source work unit ∧ new source work unit ∧ (target work unit container ∨ target work unit)
Actions:

- If target is a work unit apply for new source and target: **Add Work Unit to Work Unit Dependency**, else apply **Add Work Unit to Container Dependency**.
- If target is a work unit apply for old source and target: **Remove Work Unit Dependency**, else apply **Remove Container Dependency**.

Output: newly connected concepts as illustrated in Figure C-3 (upper half for container dependency and lower half for work unit dependency).

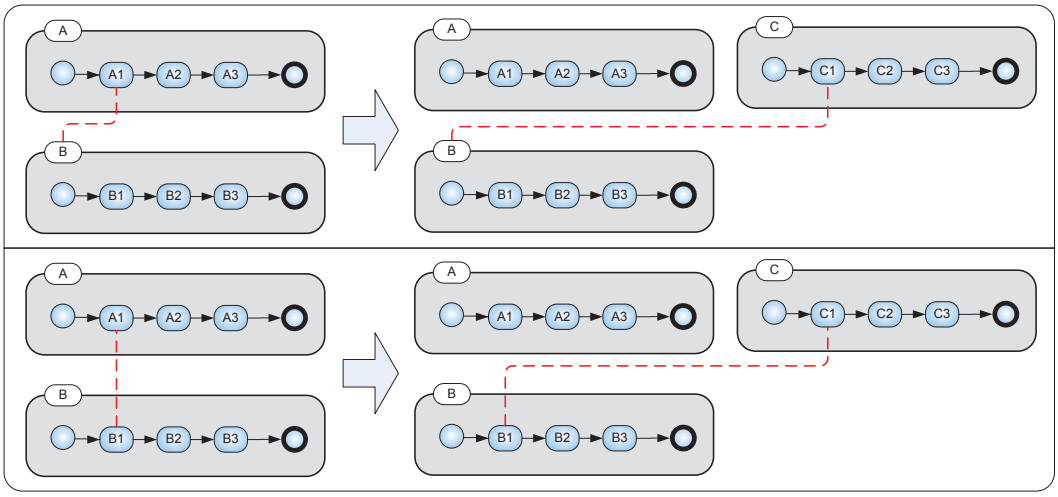

Figure C-3: Move work unit dependency

One thing that also happens frequently in projects is the situation that activities must be handed over from one human to another. The cause for this might be for example the unavailability of one human or that an activity has been assigned to a whole team and the team leader then passes it on to a concrete human best suitable for the activity. For such cases, we have applied two different actions for distributing human activities. The first one, shown in the following, deals with the distribution of one concrete assignment activity from one human to another.

Distribute Activity

This action is applied to change the executing human of an assignment activity.
Preconditions: assignment activity must be in status 'Created', 'Active', or 'Inactive'.

Input: assignment activity ∧ human
Actions:

- Remove executing human.
- Set new human as executor.

Output: assignment activity with new executor.

Another specific case is the distribution of a more complex activity, an assignment, from one human to another. In this case, the assignment might already be started and all comprised assignment activities must still be transferred to the new executor as shown in the following.

Distribute Assignment

This action is applied to change the executing human of an assignment and all of his related assignment activities belonging to that assignment.
Preconditions: assignment in state ‘Active’ or ‘Inactive’.
Input: assignment ∧ old executor ∧ new executor
Actions:

- Remove executing human of assignment.
- Set new executor as executor of assignment.
- For all assignment activities having the old executor and that are in state ‘created’ or ‘running’ apply **Distribute activity** with the new executor.

Output: assignment with new executor.